ELSEVIER

Artificial Intelligence 82 (1996) 1-20

Artificial Intelligence

Structure-driven algorithms for truth maintenance ★

Rina Dechter [a,*], Avi Dechter [b,1]

[a] *Information and Computer Science Department, University of California, Irvine, CA 92717, USA*

[b] *Department of Management Science, School of Business Administration and Economics, California State University, Northridge, CA 91330, USA*

Received February 1990; revised August 1994

Abstract

This paper studies truth maintenance and belief revision tasks on singly-connected structures for the purpose of understanding how structural features could be exploited in such tasks. We present distributed algorithms and show that, in the JTMS framework, both belief revision and consistency maintenance are linear in the size of the knowledge-base on singly-connected structures. However, the ATMS task is exponential in the branching degree of the network. The singly-connected model, while restrictive, is useful for three reasons. First, efficient algorithms on singly-connected models can be utilized in more general structures by employing well-known clustering techniques. Second, these algorithms can serve as approximations or as heuristics in algorithms that perform truth maintenance on general problems. Finally, the analysis provides insights for understanding the sources of the computational difficulties associated with JTMS and ATMS.

1. Introduction

Reasoning about dynamic environments is a central issue in artificial intelligence. When dealing with a complex environment, only partial description of the world is known explicitly at any given time. A complete picture of the environment can only be speculated by making assumptions, which must be consistent with each other and with the available information. When new facts become known, some assumptions

★ This work was supported in part by grants from the Air Force Office of Scientific Research, AFOSR 900136, by NSF grant IRI-91573636, by grants from Toshiba of America and Xerox Palo Alto research center.

* Corresponding author. E-mail: dechter@ics.uci.edu.

[1] E-mail: avi@cs.ucla.edu.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved

SSDI 0004-3702(94)00096-4

must be changed so that the consistency of our view of the world is maintained at all times.

Truth maintenance systems (TMSs) are computational schemes aimed at handling such situations. In its generic form, a TMS manipulates propositional formulas built from propositional symbols and standard Boolean connectives. A selected subset of the propositional symbols are called premises. That is, statements about the environment that do not require proof either because they are known to be true (observational facts) or because they may be assumed to be true as long as there is no evidence to the contrary (assumptions). In addition to premises, the system contains a set of Boolean *constraints*, sentences which describe invariant properties of the environment.

The main functionality of a TMS is to determine whether the truth of a given proposition follows logically from a given set of premises and from the set of constraints, and to keep this information current.

Two primary approaches to TMS implementation have been proposed: the JTMS (Justification-based TMS) [14,21,23] and the ATMS (Assumption-based TMS) [11]. Each design implements the main functionality in a different way. A JTMS starts with the given premise set and attempts to identify all provable propositions, hoping that a proof will be derived for the target proposition. An ATMS, on the other hand, maintains for each proposition a collection of consistent premise subsets (called environments) any of which is sufficient for proving the proposition. To prove that the target proposition follows from the given premise set, all that is needed is to verify that this premise set contains at least one of the proposition's environments.

A JTMS must also be able to check whether its current premise set is consistent with the constraints, and, in case it is not, point to a part of the premise set which can be shown to be a source of the inconsistency. This functionality of JTMS is closely related to the framework of agent's belief revision. The basic idea in this more recent work is to enforce minimal change in an agent's belief necessary to account for a new contradiction in his knowledge [1]. Thus far, research in belief revision focused on the task of finding a minimal revision or on finding what holds under all minimal revisions. Here we focus on the identification of a *minimum number* of changes to represent minimal change.

When originally introduced, algorithms for truth maintenance were not accompanied by complexity analysis, or performance guarantees [12,14,21]. Nevertheless, experimental work with these tools, and more recent complexity analysis, have shown that both JTMS and ATMS functionalities are very inefficient, with ATMS exhibiting higher complexity than JTMS in both time and space.

A common strategy for reducing computational complexity has been to use efficient algorithms [22] which are complete only for restricted languages (e.g., unit resolution for Horn theories), but may be incomplete in general. This paper examine another type of restriction, one based on the *structure of the knowledge-base* as reflected in the graphical properties of the TMS constraint set. We will present algorithms that are tractable for tree-like knowledge-bases and whose complexity for general theories can be bounded as a function of the "distance" of the knowledge-base from a tree.

The algorithms are based on recent work in *constraint satisfaction problems* (CSPs), which resulted in many efficient algorithms and tractable cases tied specifically to

the structure of the problem [9, 10, 16, 20]. A constraint network consists of a set of variables, each associated with a finite set of possible values, and a set of constraints, specifying joint assignments to the variables "allowed" by the constraints. A solution is an assignment of values to all the variables such that all of the constraints are satisfied.

For the purpose of using CSP techniques, we assume that the TMS knowledge-base is represented as a constraint network. In this network, each proposition (e.g., assumptions or facts) is represented by the assignment of a specific value to a particular variable. A proposition is entailed if it is the only consistent assignment of a value to that variable. To model change we introduce the notion of *assumption variables*. The TMS algorithms will be allowed to manipulate those assumptions in response to changes to the network (possibly imposed by observations from the outside world). Since the language of constraints has the same expressive power as propositional logic [21], all the algorithms presented here are applicable to propositional languages. The connection between truth maintenance systems and constraint satisfaction problems was already pointed out by several authors e.g., [12, 22, 24]. The main thrust of these efforts has been to show that search reduction techniques developed in one area may be used to the benefit of the other. Our work here take this idea one step further.

We present the algorithms in a distributed fashion, that is, assigning a processor to each variable and letting the processors communicate with each other through a uniform protocol. We emphasize the distributed nature of our algorithms for two reasons. First, in order to allow their implementation on a real physical distributed network of processors. Second, a distributed algorithm, if *self-stabilized*, is guaranteed to converge to a solution from any initial configuration (for a precise definition see [7]). The algorithms we present are self-stabilized and, consequently, the updating process of such knowledge, in response to a local change, is already encoded in the protocol.

Following preliminaries in Section 2 whereby the tasks of TMSs is defined within the constraint network model, we present (Section 3) a distributed algorithm for computing and maintaining the *entailment* status of each proposition. In Section 4 we present a distributed belief revision algorithm for restoring consistency to an inconsistent network in a way that minimizes the *number of assumption changes*. We demonstrate the applicability of this algorithm for diagnosis (Section 5). We show that JTMS's tasks of entailment and belief revision are linear time on tree-like knowledge-bases. Finally, in Section 6 we address the ATMS *labeling task* of compiling and maintaining all the labels. We show that, although in this case the algorithm is not tractable, not even for trees, its complexity is exponentially bounded by the branching degree of the tree. Therefore, chain-like networks can be processed relatively efficiently for this task as well.

2. Definitions and preliminaries

In this section we define the notions of *constraint networks* and *relations*, relate them to propositional theories, and define in their context the TMS tasks discussed in the introduction.

Definition 1 (*Relations, networks, schemes*). Given a set of variables $X = \{X_1, \ldots, X_n\}$ associated, respectively, with domains of discrete values $D_1, \ldots, D_n$, a *relation* (or, alternatively, a *constraint*) $\rho = \rho(X_1, \ldots, X_n)$ is any subset

$$\rho \subseteq D_1 \times D_2 \times \cdots \times D_n.$$

The *projection* of ρ onto a subset of variables Q, denoted $\Pi_Q(\rho)$ or ρ_Q, is the set of all tuples defined on the variables in Q that can be extended into tuples in ρ. A *constraint network* R over X is a set $\rho_1, \ldots, \rho_t$ of such relations. Each relation ρ_i is defined on a subset of variables $S_i \subseteq X$. The set of subsets $S = \{S_1, \ldots, S_t\}$ is called the *scheme* of R. The network R represents a unique relation $rel(R)$ defined over X, which stands for all consistent assignments (or all solutions), namely,

$$rel(R) = \{x = (x_1, \ldots, x_n) \mid \forall S_i \in S, \Pi_{S_i}(x) \in \rho_i\}.$$

A partial assignment $T = t$ is a value assignment to a subset of variables $T \subseteq X$. A value x in the domain of X is said to be consistent if it is part of at least one solution of R. A tuple t is consistent if it participate in at least one solution.

The scheme of a constraint network can be associated with a *constraint graph* where each subset in the scheme is represented by a node in the graph and two nodes are connected if the corresponding relations have at least one common variable. The arcs are labeled by the common variables. A network whose constraint graph is a tree is said to be *acyclic* and its corresponding constraint graph is called a *join-tree*. When all the constraints involve exactly two variables, the network is called *binary*. In such a case, another graphical representation of the network, called the primal constraint graph, is useful. In this graph each variable is represented by a node and two nodes are connected by an arc if the variables they represent are joined by a constraint.

Any propositional theory can be viewed as a special kind of constraint network, where the domain of each variable is $\{0, 1\}$ (corresponding to $\{\textbf{false}, \textbf{true}\}$) and where each clause (a disjunction of propositional symbols or their negations) specifies a constraint (in other words, a relation) on its propositional symbols. The set of all models of the theory corresponds exactly to the set of all solutions of its corresponding constraint network.

A proposition $X = x$ is *entailed* by the network if x is the only consistent value of X, namely if it participates in all solutions. Similarly, a tuple t is entailed if it participates in all solutions. We distinguish a subset of variables $A \subseteq X$ called assumption variables.

Example 2. Consider the theory $\Phi = \{\neg A \vee \neg B, \neg B \vee \neg C, C \vee D\}$. This theory can be viewed as a constraint network over the variables $\{A, B, C, D\}$, where the corresponding relations are the truth tables of each clause, that is, $\rho(AB) = \{00, 01, 10\}$, $\rho(BC) = \{00, 01, 10\}$, and $\rho(CD) = \{01, 10, 11\}$. The scheme of the theory Φ is $\{AB, BC, CD\}$. The set of all solutions to this network (and hence the set of models of Φ) is

$$\rho(ABCD) = \{0001, 0010, 0011, 0101, 1001, 1010, 1011\}.$$

It is evident that none of the propositions or their negations is entailed. Theory Φ is acyclic as demonstrated by its constraint graph:

```
        BC
     /\
  B /   \ C
   /     \
  AB      CD
```

We next define formally the TMS tasks that we consider in this paper:

Definition 3 (*TMS functionalities defined for constraint networks*). Given a network of constraints R, over a set of variables $X = X_1, \ldots, X_n$, with a subset A, $A \subseteq X$ of assumption variables, and a set of constraints $\rho_1, \ldots, \rho_m$ over X we define:

- *JTMS main functionality*: Given an instantiation of a subset of the assumption variables, determine for each $X = x$ if it is entailed.
- *JTMS belief revision*: Given a set of assumptions for which the network is inconsistent, determine a minimal size set of assumption changes that restore consistency.
- *ATMS main functionality*: For each value x of each variable X, determine the set of all consistent minimal (in the sense of set inclusion) instantiations of assumption variables that entail $X = x$.

The algorithms we discuss in this paper assume, initially, that the theories are *acyclic*. These algorithms are extensible to arbitrary theories via a procedure known as *tree-clustering* [10], which compiles any theory into a tree of relations. Consequently, given a general theory, the algorithms presented in the sequel work in two steps: A join-tree is computed by tree-clustering, and then a specialized tree algorithm for the particular TMS function is applied. The complexity of tree-clustering is bounded exponentially by the size of the maximal arity of the generated relations, and hence our algorithms are efficient for theories that can be compiled into tree networks of low-arity relations only. Examples of such theories are discussed in Section 5. An alternative approach for extending the algorithm to cyclic theories is via a method known as *cycle-cutset* [6], which is exponential in the cycle-cutset size of the theory. The size of the minimal cycle-cutset is normally higher than the cluster's sizes of a tree-embedding of the same theory. The virtue of the cycle-cutset however is that, unlike tree-clustering, it does not require exponential space.

3. Identifying all entailed propositions

A proposition of the type $X = x$ is entailed in a network of constraints R if the network has at least one solution and if x is the only value of X in all solutions. The approach we propose for identifying all entailed propositions involves computing for each value in the domain of each variable the number of solutions it participates in. We will refer to this number as the *numerical support* (*n-support* in short) of the value. Once this information is available for every value of a variable, entailment is easy to

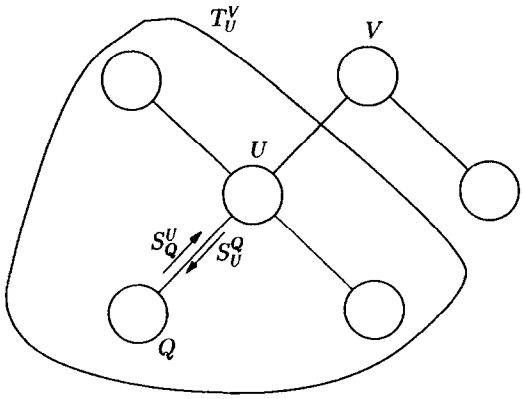


Fig. 1. A fragment of a tree network.

determine locally: *a proposition $X = x$ is entailed if x has a positive n-support while all other values of X have zero n-supports.*

The notion of numerical support may be extended to encompass not just the instantiation of a single variable but that of a tuple of variables. This is captured in the following definition.

Definition 4. Given a network of constraints over variables $X_1, \ldots, X_n$, with constraints $C_1, \ldots, C_r$, the n-support of a value x of X, denoted $s_X(x)$, is the number of solutions in which X is assigned the value x. The n-support of a tuple $C_i = c_i$, denoted $s_{C_i}(c_i)$, is the number of solutions in which $C_i = c_i$. Let R be a join-tree, and let (U, V) be an arc in the tree (see Fig. 1). The subtree rooted at U, which does not contain V, is denoted T_U^V, and $s_U^V(u)$ is the number of solutions of T_U^V which are consistent with $U = u$.

In the following we present a *distributed* algorithm that compiles all n-supports for an acyclic network. The algorithm is a distributed adaptation of a known tree algorithm that computes the number of solutions [9]. Following [9], it is easy to show that:

Theorem 5. *Let T be a join-tree, and let U be one of its relations, the overall n-support for $U = u$ can be expressed as a function of the n-supports of its neighbors, namely:*

$$s_U(u) = \prod_{(Q,U)\in T} \sum_{q:q_{U\cap Q}=u_{U\cap Q}} s_Q^U(q). \tag{1}$$

Eq. (1) lends itself to a distributed propagation scheme. If constraint U gets from each neighboring node, Q, the n-support vector, s_Q^U, it can calculate its own support vector using Eq. (1), and, at the same time, it can generate an appropriate message for each of its own neighbors. The message which U sends to Q, s_U^Q (i.e., the support vector reflecting the subtree T_U^Q) can be computed by:

$$s_U^Q(u) = \prod_{(C,U)\in T, C\neq Q} \sum_{c:c_{C\cap U}=u_{C\cap U}} s_C^U(c). \tag{2}$$

support-propagation(U)
Input: A join-tree T, having variables $X = \{X_1, \ldots, X_n\}$. A relation $U \in T$, the n-support vector s_Q^U, for all neighbors Q of U.
Output: The support vector $s_U(u)$, and for each neighbor Q, the vector $s_U^Q(u)$.

(1) Compute n-supports for each tuple:

$$s_U(u) = \prod_{(U,Q)\in T} \sum_{q;q_{U\cap Q}=u_{U\cap Q}} s_Q^U(q).$$

(2) Update single-valued n-supports: For every X in the constraint U,

$$s_X(x) = \sum_{u\in U;u_X=x} S_U(u).$$

(3) Compute n-supports messages for each neighbor Q:

$$s_U^Q(u) = \prod_{(C,U)\in T, C\neq Q} \sum_{c;c_{C\cap U}=u_{C\cap U}} s_C^U(c).$$

Fig. 2. Algorithm support-propagation.

The message generated by a leaf constraint is a vector consisting of 1's representing the tuples allowed by that constraint. The computation consists of nodes sending to their neighbors the partial n-support vectors whenever they are readily computed. When all nodes have received all the n-supports, the overall n-supports for each tuple in their constraint can be computed using (1).

Having the n-supports for each *tuple* in each relation, the single-valued n-supports can be derived by picking a relation in the tree containing the variable in question, and then summing the corresponding n-supports of all tuples in the relation that has that value. The algorithm for node U is summarized in Fig. 2.

If the algorithm is executed in a real distributed environment, its convergence is guaranteed after at most $n \cdot d$ messages when d is the maximal distance between two leaf nodes. Under some synchronization the number of messages can be reduced to $2n$. It can, similarly, be shown that the cost of updating the n-supports following a single change in one relation will cause at most $2n$ messages until convergence. The following theorem focuses only on the sequential complexity of the algorithm.

Theorem 6. *The sequential complexity of algorithm support-propagation is* $O(n \cdot r \cdot \log r)$, *where r is the maximal number of tuples of any relation in the join-tree.*

Proof. The summation operation between any two relations U and Q can be accomplished in $O(r \cdot \log r)$ steps as follows. Relation Q is projected on the variables in the intersection of Q and U. Each projected tuple is associated with a new n-support computed by summing the corresponding n-supports in the message that Q sends to U. This operation takes $O(r)$ steps. Then, the projected relation can be sorted in $O(r \cdot \log r)$ steps and each tuple can be retrieved in $\log r$ steps by relation U. □

In conclusion, once all n-supports are computed, each tuple of each relation "knows" if it is entailed, consistent, or inconsistent in constant time, while for each singleton this information can be obtained in $O(r)$ steps.

If one is interested in entailment only, *flat* support vectors, consisting of zeros and ones, can be propagated in exactly the same manner, except that the summation operation in (1) should be replaced by the logic operator OR, and the multiplication can be replaced by *AND*. This results in a distributed arc-consistency algorithm that minimizes the number of message passing along the links.

Computing n-supports is not easier than determining entailment directly (namely, by calculating all the solutions of the network). In fact, it is generally very complex, as it is #P-complete [25]. However, for acyclic networks the complexity of computing n-supports is the same as that of computing one solution. We choose to compute n-supports since this can be accomplished at no additional cost on trees, while it is more informative and might prove useful in applications.

4. Belief revision

When, as a result of a new input, the network enters a contradictory state (i.e., no solution exists), it often means that the new input is inconsistent with the *current set of assumptions*, and that some of these assumptions must be modified in order to restore consistency.

It is widely agreed that the subset of assumptions that are modified should be minimal, namely, it must not contain any proper subset of assumptions whose simultaneous modification is sufficient for that purpose. A sufficient (but not necessary) condition for this set to be minimal is for it to be as small as possible. In this section we show how to find a minimum cardinality set of assumptions that need to change in order to restore consistency.

Assume that a constraint which detects an inconsistency (i.e., all its n-supports are zero) sends this information to the entire tree, creating in the process a directed tree rooted at itself. Given this rooted join-tree, belief revision proceeds as follows.

With each tuple v of each relation V in the join-tree T, we associate a weight $w(v)$, denoting the minimum number of assumption values that must be changed in the subtree rooted at V in order to make v consistent relative to this subtree. For each tuple q of a relation Q we denote by $\delta(q)$, the number of its assumptions that differ from the *current assumptions*. We denote by $child(V)$ the set of child relation of V in the join-tree.

The weights obey the following recursion (see Fig. 3):

$$w(v) = \delta(v) + \sum_{Q;Q \in child(V)} \min_{q \in Q; q_{V \cap Q} = v_{V \cap Q}} w(q). \tag{3}$$

The computation of the weights is performed distributedly from the leaves of the directed tree to the root. A node waits to get the weights of all its child nodes, computes its own weights according to (3), and makes them available to its parent. During this *bottom-up propagation* a pointer is kept from each tuple of V to the tuples in each of its child nodes where a minimum is achieved. When the root receives all the weights, it

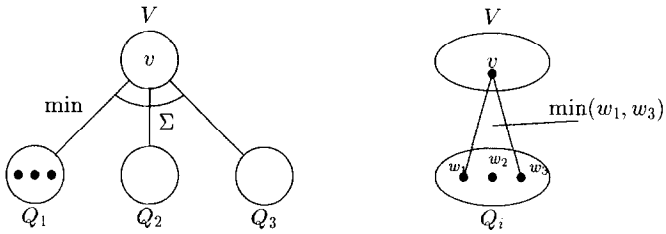


Fig. 3. Weight calculation for relation V.

belief-revision(V)
Input: A rooted join-tree T, having variables $X = \{X_1, \ldots, X_n\}$, and relations $V_1, \ldots, V_m$. A set of assumption variables $A \subseteq X$ and its current assignment $A = a$.
Output: The weights of each tuple and an indication to minimal changes in assumption assignment.

(1) For each tuple $v_i \in V_i$ compute $\delta(v_i)$.
(2) (*Bottom-up*) Compute weights of V (given the weights for its child nodes)

$$w(v) = \delta(v) + \sum_{Q;Q \in child(V)} \min_{q \in Q; q_{V \cap Q} = v_{V \cap Q}} w(q).$$

(3) (*Top-down*) Given a tuple t selected by parent, change assumption value accordingly, select that tuple of each child relation pointed to by t.

Fig. 4. Algorithm belief-revision.

computes its own weights and selects one of its minimizing tuples. It then initiates (with this tuple) a *top-down propagation* down the tree, following the pointers marked in the bottom-up propagation. At termination this process marks the assumption variables that need to be changed and the appropriate changes required. The algorithm is summarized in Fig. 4.

Theorem 7. *Given a join-tree T whose largest relation size is at most r, algorithm belief-revision can find one minimal revision in $O(n \cdot r \cdot \log r)$. The set of all minimal cardinality revisions can be generated in $O(l + n \cdot r \cdot \log r)$, where l is the size of the output.*

Proof. The minimization operation between any two relations of size r can be accomplished in $O(r \cdot \log r)$ steps as follows. A child node C, with its weighted relation, projects its relation on the intersection of its own variables and that of its parent relation P. The weight associated with each projected tuple is the minimum weight among the weights of all the corresponding tuples in C. This operation can be accomplished in linear time $O(r)$. Then the projected relation can be sorted in $O(r \cdot \log r)$ steps and can be retrieved in $\log r$ steps by the parent node P for the summation operation. If all the minimum cardinality revisions are needed they can all be retrieved in output linear time by following the pointers in the top-down step of the algorithm. □

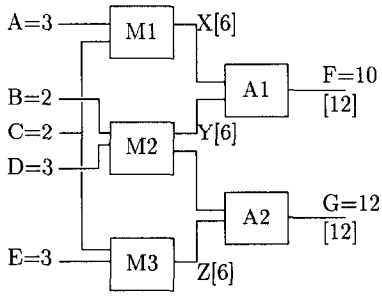


Fig. 5. A circuit example.

Once belief revision has been terminated, all assumptions can be changed accordingly, and the system can get into a new stable state using support propagation. There is no need, however, to activate the whole network for belief revision, because the n-support information clearly points to those subtrees where no assumption change is necessary. We will illustrate the belief revision algorithms in the next section using a circuit diagnosis example.

5. A circuit diagnosis example

An electronic circuit can be modeled in terms of a constraint network by associating a variable with each input, output, intermediate value, and device. Devices are modeled as bi-valued *assumption variables*, having the value "0" if functioning correctly (default) and the value "1" otherwise. There is a constraint associated with each device, relating the device variable with its immediate inputs and outputs. Given input data, the possible values of any intermediate variable or output variable is its "expected value", namely, the value that would have resulted if all devices worked correctly, or some "unexpected value" denoted by "e". A variable may have more than one expected value. For the purpose of this example we assume that the set of expected values for each variable was determined by some pre-processing and all the other values are marked collectively by the symbol "e".

Consider the circuit of Fig. 5 (also discussed in [4, 13, 18]), consisting of three multipliers, M_1, M_2, M_3, and two adders, A_1 and A_2. The values of the five input variables, A, B, C, D, and E, and of the two output variables, F and G, are given. The numbers in the brackets are the expected values of the three intermediate points X, Y, and Z, and of the outputs. The relation defining the constraint associated with the multiplier M_1 is given in Fig. 6 as an example, as well as the initial weights associated with the tuples of these leaf constraints ($\delta = w$ for leaf nodes). The weight of the first tuple is "0" since the assumption variable M_1 is assigned the currently assumed value, "0", while in the second tuple the assumed value is changed to "1". Given the inputs and outputs of the circuit, the objective is to identify a minimal set of devices which, if presumed to be malfunctioning, could be consistent with the observed behavior (i.e., $G = 12$ and $F = 10$).

M_1	A	C	X	
0	2	3	6	$w = 0$
1	2	3	e	$w = 1$

Fig. 6. A multiplier constraint.

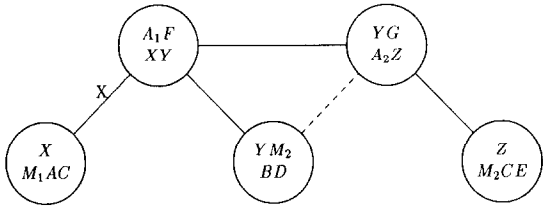


Fig. 7. An join-tree for the circuit example.

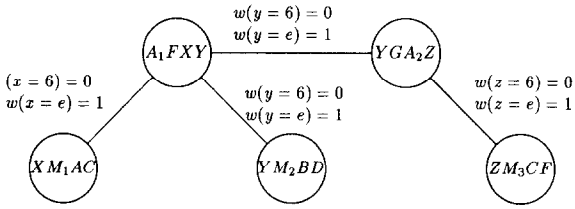


Fig. 8. Weight calculation for the circuit example.

The join-tree of the constraint network modeling this circuit is given in Fig. 7. This network is acyclic, as is evident by the fact that a join-tree can be obtained by eliminating the redundant arc (marked by a dashed line) between constraint (M_2, B, D, Y) and (A_2, Z, Y, G). For more details see [10].

Initially, when no observation of output data is available, the network propagates its n-supports assuming all device variables have their default assumption value "0". In this case only one solution exists and therefore the supports for all consistent values are "1". The diagnosis process is initiated when the value "10" is observed for variable F which is different from the expected value of 12. The value "10" is fixed as the only consistent value of F. At this point, the constraint (X, A_1, F, Y),[2] which is the only one to contain F has all its n-supports equal "0" and it induces direction on the join-tree, resulting in the directed tree (rooted at itself) of Fig. 7, and belief revision is initiated.

Each tuple will be associated with the minimum number of assumption changes in the subtree underneath it. Instead of indicating the weights associated with each tuple we indicate the weight projected on the variables on the arcs of the tree. In Fig. 8 the weights associated with the arcs of the three leaf constraints (i.e., the multipliers constraints), projected on their outgoing arcs is illustrated. These are derived from the weights associated with their incoming constraints (see the weights in Fig. 6). For instance, the weight associated with X is $w(X = 6) = 0$ since "6" is the expected value of X when

[2] For simplicity we will refere here to a constraint by the subset of variables on which it is defined.

A_2	Z	G	Y	Weights	Faulty devices
0	6	12	6	$w = 0$	none
0	e	12	e	$w = 1$	M_3
1	6	12	e	$w = 1$	A_2
1	e	12	e	$w = 2$	M_3 & A_2

Fig. 9. The weights of constraint (Y, G, A_2, Z).

	A_1	F	X	Y	Weights	Faulty devices
(1)	0	10	6	e	2	$(M_3 \vee A_2)$ & M_2
(2)	0	10	4	6	1	M_1
(3)	0	10	e	e	3	M_1 & M_2 & $(M_3 \vee A_2)$
(4)	1	10	6	6	1	A_1
(5)	1	10	6	e	3	A_1 & M_2 & $(M_3 \vee A_2)$
(6)	1	10	e	e	4	A_1 & M_2 & M_1 & $(M_3 \vee A_2)$

Fig. 10. The weights of constraint (A_1, F, X, Y) (the root).

M_1 works correctly (which is the default assumption), and $w(X = e) = 1$ since any other value can be expected only if the multiplier is faulty. Next, the weights propagate to constraint (Y, G, A_2, Z). This constraint is the only parent node of (Z, M_3, C, F) and its weights are given in Fig. 9 (note, that G's observed value is 12).

The corresponding projected Y's weights are indicated on the outgoing arc of constraint (Y, G, A_2, Z) in Fig. 8. Finally, the weights associated with the root constraint (A_1, X, Y, F) are computed by summing the minimum weights associated with each of its child nodes. The tuples associated with the root constraint and their weights are presented in Fig. 10.

We see that the minimum weight is associated with tuple (2) indicating M_1 as faulty, or tuple (4) indicating A_1 as faulty. Therefore, either A_1 or M_1 are faulty (the weights can also be used as a guide for additional measurement that should delineate between the different diagnoses).

This example illustrates the efficiency of the belief revision process when the special structure of the problem is exploited. By contrast, handling this problem using ATMS [11] may exhibit exponential behavior. A similar algorithm exploiting the framework of probabilistic networks is given [17].

6. ATMS labeling

In this section we focus on the primary ATMS functionality, namely, finding one or all minimal instantiations of assumption variables in a given network of constraints that entail the proposition $X = x$. This task is often called *label determination* in the ATMS terminology. We call each tuple representing such an instantiation a *support tuple* or t-support. The main result of this section is a bound on the complexity of finding one t-support for $X = x$ which is exponential in the branching degree of the tree. We also introduce an algorithm for performing this task which attains this complexity as a lower bound, thereby proving that the bound is tight. When computing *all* t-supports, the

complexity increases by a linear factor of the output. If only a subset of the variables is regarded as assumption variables, the complexity is still exponential in the degree of the tree unless the assumption variables are distributed in a way that reduces the *effective* degree of the tree. It should be noted that the ATMS task is equivalent to what is often referred to as *abduction*, and, therefore, the results we present extend to the abduction task as well.

To simplify the exposition we will describe an algorithm for computing minimal t-supports for trees of *binary constraints*. In this case, the *primal constraint graph*, where nodes represent variables and arcs indicate the existence of constraints between pairs of variables, is more convenient. The extension of this algorithm to general join-trees is straightforward since a join-tree can be viewed as a regular binary tree where each relation is a compound variable and its tuples are its values. We assume, without loss of generality, that all the variables are assumption variables.

Definition 8. Given a network of constraints R over a set of variables X, a partial instantiation $T = t$, $T \subseteq X$, is a support tuple (t-support) for $X_i = x_i$ iff for every solution s of R,

$$s_T = t \longrightarrow s_{X_i} = x_i. \tag{4}$$

$T = t$ is a minimal t-support of x_i is there is no subtuple of t satisfying (4). The set of all minimal t-supports for x_i relative to a network R is denoted $mst_R(x_i)$ or $mst(x_i)$ when the network's identity is clear.

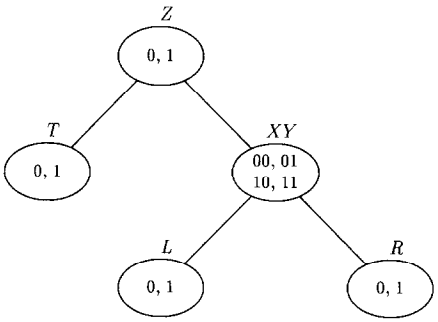
Example 9. Consider the following propositional theory

$$\varphi = \{(T \rightarrow Z), (R \rightarrow Y), (L \rightarrow X)((X \vee Y) \rightarrow Z)\}. \tag{5}$$

The theory can be modeled as binary tree network with four bi-valued variables T, Z, R, L and the compound variable XY having the domain $\{00, 01, 10, 11\}$. The explicit constraints are given by their truth tables. The constraint graph and the explicit constraints are depicted in Fig. 11.

We will illustrate the main idea of the algorithm through an example. Suppose that we want to compute all minimal t-supports for $Z = 1$ in φ. The algorithm begins by generating a directed tree rooted at Z. Then, for each variable and each of its values, it computes all its minimal support tuples *restricted to the child nodes* of the relation, called *minimal child support*. For instance, the set of minimal child supports for $XY = 11$ is $(L = 1, R = 1)$, while for $XY = 01$ the set is empty since no tuple over the child variables R and L entails $XY = 01$. The set of all minimal child supports for $X = x$, in a given directed tree, is denoted by $mcs(x)$.

Given a minimal child support, l, new supports can be generated by replacing a value in l by one of its own minimal child supports. For instance, since $(XY = 11)$ is a minimal support tuple for $Z = 1$ and since $XY = 11$ is minimally supported by $(L = 1, R = 1)$, this past set is a new support for $Z = 1$. This property seems to suggest that the set of support tuples can be generated recursively by a bottom-up process from leaves to root.



$T \rightarrow Z$

T	Z
1	1
0	1
0	0

$R \rightarrow Y$

R	XY
1	01
1	11
0	01
0	11
0	00
0	10

$L \rightarrow X$

L	XY
1	10
1	11
0	10
0	11
0	00
0	01

$X|Y \rightarrow Z$

XY	Z
01	1
10	1
11	1
00	1
0	00

Fig. 11. An example of a binary tree network.

There are, however, two problems. First, the minimality property *is not* maintained by this process. The set $(L = 1, R = 1)$, which has been generated by the substitution process, is not a minimal support for $Z = 1$ since either $L = 1$ or $R = 1$ independently support $Z = 1$. The second, and the more significant problem is that *not all support tuples* are generated. Consider again the tree in Fig. 11 restricted to variables (Z, XY, L). Each of the values $(XY = 10)$ and $(XY = 11)$ is a minimal support to $Z = 1$. However, since each one of these values is not individually supported by $L = 1$ ($L = 1$ is consistent with both of them), $L = 1$ will not be generated by the substitution process (although it is a minimal support for $Z = 1$) because it can substitute neither $XY = 01$ nor $XY = 11$.

The algorithm we present next is based on the idea discussed above: compute the minimal t-supports locally and generate the rest by a recursive substitution process. To overcome the problems mentioned, the algorithm computes local t-supports to a *subset of values* in a variable's domain, rather than to singletons. As was shown in the example, several values of a variable can play identical roles in a support tuple (e.g., $XY = 10$ and $XY = 11$ both supporting $Z = 1$) and, therefore, if a disjunction of such values has a minimal support, it can replace any of the elements of this disjunction in that label. For instance, since $L = 1$ supports the disjunction $XY = 10$ or $XY = 11$ it can replace either one of them in a t-support. Next, we formalize these notions.

Definition 10. Let V and C be two variables in a network, and let R_{CV} denote the constraint of allowed pairs between them. $M_C^V(c)$ denotes the set of values in the domain of V that are consistent with $c \in C$ (Fig. 12(a)). Namely,

$$M_C^V(c) = \{v \in V \mid (c, v) \in R_{CV}\}. \tag{6}$$

Definition 11. Given a variable V with its child nodes $C_1, \ldots, C_l$ (Fig. 12(b)) and a subset of consistent values of V, denoted A_V, a child support for A_V is an instantiation tuple $(C_1 = c_1, \ldots, C_t = c_t)$ over a subset of its child variables that entails A_V. Namely, a child support tuple, $(C_1 = c_1, \ldots, C_t = c_t)$, satisfies:

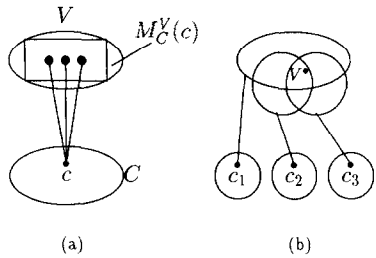


Fig. 12. Computing the minimal child support.

$$\bigcap_{c_j \in C_j} M_{C_j}^V(c_j) \subseteq A_V. \tag{7}$$

A child support tuple is minimal if no subset of it satisfies condition (7).

Example 12. It can be verified that (see Fig. 11), $Z = 1$ has four minimal child supports given by:

$$mcs_Z(\{1\}) = \{(T = 1), (XY = 01), (XY = 10)(XY = 11)\},$$

while $mcs_{XY}(\{01, 10, 11\}) = \{(L = 1), (R = 1)\}$, and it does not contain the support $(L = 1, R = 1)$ since it is not minimal.

Given a minimal child support, $l = (C_1 = c_1, \ldots, C_t = c_t)$ of A_V, we denote by $l|(C_r = c_r')$ the tuple resulting from exchanging c_r by c_r' in l. If $l|(C_r = c_r')$ is also a minimal t-support for A_V we say that c_r and c_r' play identical roles with respect to l and A_V and that they are exchangeable.

Definition 13. Given a subset A_V of V, a tuple $l \in mcs(A_V)$, and a variable–value pair $C = c$ in l we define the set of *label-dependent* values of C, denoted by $F_{A_V,l}(C)$ (or $F_l(C)$ for short), which are exchangeable in label l:

$$F_{A_V,l}(C) = \{c \in C \mid l|(C = c) \in mcs(A_V)\}. \tag{8}$$

A minimal child support for A_V is also a minimal support since we assumed that all variables are assumption variables, and it can recursively generate additional minimal support tuples. Let $l \in mcs(A_V)$ be such that

$$l = (C_1 = c_1, \ldots, C_r = c_r, \ldots, C_t = c_t)$$

and denote by $mst(A_V)$ all the minimal support tuples for A_V restricted to its rooted tree. The set of minimal supports generated via $l = (C_1, \ldots, C_t)$ in that subtree satisfies:

$$mst_l(A_V) = mst(F_l(C_1)) \times \cdots \times mst(F_l(C_r)) \times \cdots \times mst(F_l(C_t)) \tag{9}$$

and

$$mst(A_V) = \bigcup_{l \in mcs(A_V)} mst_l(A_V). \tag{10}$$

label-generation(V)
Input: A tree T rooted at V. V has a set of child variables $C_1, \ldots, C_t$. The set of label-dependent subsets of V, $L(V)$.
Output: The set of all minimal support sets to each $A \in L(V)$.
(1) (*Top-down*) For each $A_V \in L(V)$ compute the set of child support tuples $mcs(A_V)$ using Eq. (7).
(2) For each $l \in mcs(A_V)$ and for each $c_i \in l$ do
(3) Compute $F_l(C_i)$ and add it to $L(C_i)$.
(4) (*Bottom up*) (once $mst(F_l(C_i))$ are available) For each $A_V \in L(V)$ compute $mst(A_V) = \cup_{l \in mcs(A_V)} \times_{C_i \in l} mst(F_l(C_i))$.

Fig. 13. Algorithm label-generation.

Combining (9) and (10) yields a recursive equation for calculating the minimal support tuples of a subset A_V restricted to the subtree rooted at V:

$$mst(A_V) = \bigcup_{l \in mcs(A_V)} \times_{\{C_i \in l\}} mst(F_l(C_i)). \tag{11}$$

The algorithm for generating all minimal support tuples is summarized in Fig. 13. The first phase is a top-down process generating all the minimal *child supports* of all the relevant *label-dependent* subsets. The process continues top-down where the label-dependent subsets of a variable are computed only after their parents had already computed *all* their minimal child supports (or *mcs*'s for short). The second phase is a bottom-up process that starts at the leaves and continues level by level until it reaches the queried relation. Each relation computes, in its turn, for each of its label-dependent subsets, all its minimal t-supports restricted to their tree. This is accomplished by substituting a value c of C in a label l that supports A_V by one of the (already computed) minimal support sets of the subset $F_l(C)$.

The algorithm assumes that the n-supports are explicitly maintained and thus no inconsistent values participate in a support nor in a label-dependent subset.

The computation of the label-dependent subsets can be accomplished by scanning the set $mcs(A_V)$ of a given set A_V and determining for each child node the subset of its values that satisfies condition (8). Computing the minimal child supports for A_V can be implemented by a standard search algorithm that checks condition (7) for all instantiations of one child variable first, then goes to two variables, using values not selected previously, and continues to larger supports. The supports generated that way are minimal. The supports of leaf values are the empty tuples.

Theorem 14. *Algorithm label-generation generates all and only minimal support sets.*

Proof. See Appendix A.

The time complexity of the algorithm can be computed along its various steps. The computation of the minimal child supports and the label-dependent subsets is performed locally, between every node and its children. Given a parent node having d child

variables and t label-dependent subsets (already computed with respect to its parent), where $k \leqslant t \leqslant 2^k$, we can test condition (7) on subsets of child variables, in increasing order of their size. Since, in the worst case, all subsets of $d/2$ variables may need to be tested, and since for every such set all combinations of values may be involved, we get:

$$T(\textit{label-generation}) = \Theta\binom{d}{d/2} \cdot k^{d/2} = \Theta(2\sqrt{k})^d. \tag{12}$$

This performance can be attained in the worst case even when the size of the resulting *mcs* set is very small. Notice that the tree structure *does not prevent exponential computation* for finding even one *mcs*.

Once all *mcs*'s for a variable are generated, all label-dependent subsets of each child variable will be generated and this may take $O(t^2)$ steps.

Once the minimal child supports for all variables are available, the time required for generating all minimal support tuples is linear in the output (this is the bottom-up process). The total time complexity of the algorithm is dominated, therefore, by the calculation of the minimal child supports. Consequently,

Theorem 15. *The worst-case time complexity of algorithm label-generation has a lower bound* $\Omega(\exp(d) + n_s)$, *and an upper bound* $O(\exp(d) + n_s)$, *where* n_s *is the number of minimal support tuples in the output.*

The results in this section are restricted for binary tree networks. A simple way for extending them to an arbitrary join-tree is to view each relation in the tree as a meta-variable when the constraints between relations say that two partial tuples are consistent if they coincide on shared variables. Once we compute for each tuple in the root relation all its minimal supports, the minimal supports of individual values can be generated by manipulating only the tuples' support of that relations. We conclude that:

Corollary 16. *Finding all minimal support tuples of a partial tuple* t *appearing in a root relation of a join-tree can be accomplished in* $O(t^d + n_s)$ *when* t *bounds the number of tuples in each relation,* d *is the branching degree in the tree and* n_s *is the size of the output.*

7. Conclusions and related work

This paper presents truth maintenance algorithms within the framework of constraint networks. The common feature making these algorithms attractive is that their complexity is linked to the structure of the network, making their performance more predictable. Our results provide some theoretical explanations for the behavior of truth maintenance algorithms reported in the literature.

The ideal structure for all three algorithms is an acyclic network. In that case, the two JTMS algorithms (entailment and belief-revision) are time and space linear. On the other hand, the ATMS algorithm, which finds all minimal support sets for each

proposition, is time and space exponential. The exponent is reduced though from n (the number of variables) to d (the branching degree of the constraint graph).

These results support the common observation that the ATMS task is more complex than the JTMS task. It appears that, in practice, users should be advised to implement the JTMS strategy whenever possible and to use ATMS only when the task is really necessary.

When the constraint network is not acyclic, the method of tree-clustering [10] should be used as a pre-processing step. This method uses aggregation of constraints into equivalent constraints involving larger clusters of variables in such a way that the resulting network is acyclic. The clustering scheme is exponential in the *size of the largest cluster*, making the complexity of pre-processing also dependent on the structure of the network, with near-tree networks requiring less computation.

An alternative approach to dealing with cyclic theories is to use the *cycle-cutset* method [6], which is exponential in the cycle-cutset size of the theory. The size of the minimal cycle-cutset is larger than the cluster's sizes of a tree-embedding of the same theory, however, unlike tree-clustering, the cycle-cutset method does not require exponential space.

Because of the need for re-structuring the network, it is clear that the algorithms proposed in this paper are best-suited for situations where the structure of the knowledge-base is either fixed or involves only minor topological changes over time. Examples of such cases are physical or biological systems such as electronic circuits and model-based medical diagnosis systems.

Structure-exploiting algorithms for diagnosis that were inspired by algorithms on probabilistic networks appear in [3, 17, 19]. The merits of the belief revision algorithms appearing here were recently tested experimentally on various circuit diagnosis examples. The algorithm's performance was shown to be superior relative to model-based diagnosis (MBD) algorithms [15]. Another paper [2] discusses the task of finding all minimal sets of changes needed to restore consistency (as opposed to the task of all sets of minimum cardinality, discussed in the current paper). It is shown that while a propagation scheme is available in this case as well, it has greater complexity.

Appendix A. Proof of Theorem 14

Theorem 14. *Algorithm label-generation generates all and only minimal support sets.*

Proof. It is clear that the algorithm generates only t-supports. We will therefore focus on showing that any generated set is minimal and that any minimal t-support is generated by the algorithm. The proof is by induction on the distance of the variables in the support set from the queried variable.

Let $Z = z$ be the queried variable and let $s = (X_1 = x_1, \ldots, X_t = x_t)$ be a generated t-support. We will show that s is a *minimal* t-support. Let h be the furthest distance from Z of variables in s. For $h = 1$ it is known that only *minimal* t-supports are generated, i.e., the minimal child supports. Assuming that the generated t-supports having variables with distance $h - 1$ or less are all minimal, we will show that s, having longest distance

h, is also a minimal t-support. Let $s^P = (X_1{}^P = x_1, \ldots, X_j{}^P = x_j)$ be a subtuple of the t-support s having distance h such that all the participating variables have a common parent, P. Let $\bigcap_{x_i \in X_i{}^P} M^P_{X_i{}^P}(x_i) = A_P$. Since s was generated by the algorithm, s^P must be a minimal support label of a label-dependent subset of P. Therefore A_P must be a subset of a label-dependent subset of P and therefore each value of A_P can replace s^P in s, resulting in a t-support with lower distance. By performing this "reverse substitution" to all the variables in s having distance h we get a t-support whose utmost distance from Z is $h-1$ and which must have been generated by the algorithm. By the induction hypothesis this support is minimal and since each s_P is a *minimal* child support of the corresponding label-dependent subset (otherwise it would not be applied), the (nonreversed) substitution must result in a minimal t-support having distance h or less.

We will now show that if s is a minimal t-support for $Z = z$ it must be generated by the algorithm. Let $\{s^{P_i}\}$ be all the subsets of s having distance h from Z indexed by their parents, P_i. Let the corresponding ranges that each set of children determines on their parent be defined by:

$$\bigcap_{x_j \in X_j^{P_i}} M^{P_i}_{X_j{}^{P_i}}(x_j) = A_{P_i}.$$

We claim that for every parent, P_i, each value in A_{P_i} can replace the subset s^{P_i} in s to yield a minimal t-support of depth not greater than $h-1$, which we call s_{h-1}. Since, otherwise, if for some P_i and for some value in A_{P_i}, the resulting s_{h-1} is not a minimal t-support for $Z = z$, it can easily be shown that from the definition of a support set, s could not be a minimal support set either, thus resulting in a contradiction. It follows that any possible s_{h-1}, generated by exchanging a value from A_{P_i} with S^{P_i} in s, is a minimal support set. Since s_{h-1} has distance $h-1$ at the most, the induction hypothesis implies that it is generated by the algorithm. Also, since by definition A_{P_i} is a label-dependent subset of P_i, and since it is supported minimally by the label s_{P_i}, the algorithm will produce s in his substitution bottom-up process. □

Acknowledgements

We would like to thank Yousri El Fattah for commenting on a recent version of this manuscript, and Dan Frost for helping with the figures. Preliminary versions of some of the contents of this paper appeared in [8] and in [5].

References

[1] C. Alchourron, P. Gardenfors and D. Makinson, On the logic of theory change: partial meet contraction and revision functions, *J. Symbolic Logic* **50** (1985) 510–530.

[2] R. Ben-Eliyahu and R. Dechter, On computing minimal models, in: *Proceedings AAAI-93*, Washington, DC (1993) 2–8.

[3] A. Darwiche, Conditional independence in ATMSS: independence-based algorithms for computing labels and diagnosis, Tech. Rept., Rockwell (1994).

[4] R. Davis, Diagnostic reasoning based on structure and behavior, *Artif. Intell.* **24** (1984) 347–410.
[5] R. Dechter, A distributed algorithm for ATMS, in: *Proceedings Bar-Ilan Symposium on Foundation of Artificial Intelligence (BISFAI-89)* (1989).
[6] R. Dechter, Enhancement schemes for constraint processing: backjumping, learning, and cutset decomposition, *Artif. Intell.* **41** (1990) 273–312.
[7] R. Dechter, Z. Collin and S. Katz, On the feasibility of distributed constraint satisfaction, in: *Proceedings IJCAI-91*, Sydney, Australia (1991) 318–324.
[8] R. Dechter and A. Dechter, Belief maintenance in dynamic constraint networks, in: *Proceedings AAAI-88*, St. Paul, MN (1988) 37–42.
[9] R. Dechter and J. Pearl, Network-based heuristics for constraint satisfaction problems, *Artif. Intell.* **34** (1987) 1–38.
[10] R. Dechter and J. Pearl, Tree clustering for constraint networks, *Artif. Intell.* **38** (1989) 353–366.
[11] J. de Kleer, An assumption-based TMS, *Artif. Intell.* **28** (1986) 127–162.
[12] J. de Kleer, A comparison of ATMS and CSP techniques, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 290–296.
[13] J. de Kleer and B. Williams, Reasoning about multiple faults, in: *Proceedings AAAI-86*, Philadelphia, PA (1986) 132–139.
[14] J. Doyle, A truth maintenance system, *Artif. Intell.* **12** (1979) 231–272.
[15] Y. El Fattah and R. Dechter, Diagnosing tree-decomposable circuit, in: *Proceedings IJCAI-95*, Montreal, Que. (1995) 1742–1748.
[16] E.C. Freuder, A sufficient condition for backtrack-free search, *J. ACM* **29** (1982) 24–32.
[17] H. Geffner and J. Pearl, An improved constraint propagation algorithm for diagnosis, in: *Proceedings IJCAI-87*, Milan (1987) 1105–1111.
[18] M.R. Genesereth, The use of design descriptions in automated diagnosis, *Artif. Intell.* **24** (1984) 411–436.
[19] J. Kohlas, Symbolic evidence, arguments and valuation networks, in: *Proceedings Uncertainty in Artificial Intelligence (UAI-93)*, Washington, DC (1993).
[20] A.K. Mackworth and E.C. Freuder, The complexity of some polynomial network consistency algorithms for constraint satisfaction problems, *Artif. Intell.* **25** (1985) 65–74.
[21] D.A. McAllester, An outlook on truth-maintenance, Tech. Rept., AI Memo 551, MIT, Cambridge, MA (1980).
[22] D.A. McAllester, Truth maintenance, in: *Proceedings AAAI-90*, Boston, MA (1990).
[23] D. McDermott, A general framework for reason maintenance, *Artif. Intell.* **50** (1991) 289–329.
[24] G. Provan, Complexity analysis of multiple-context TMSs in scene representation, in: *Proceedings AAAI-87*, Seattle, WA (1987) 173–177.
[25] L.G. Valiant, The complexity of enumeration and reliability problems, *SIAM J. Comput.* **8**(3) (1987) 105–117.

ELSEVIER

Artificial Intelligence 82 (1996) 21–44

Artificial Intelligence

Probably approximately optimal satisficing strategies ★

Russell Greiner [a,*], Pekka Orponen [b]

[a] *Siemens Corporate Research, 755 College Road, East Princeton, NJ 08540, USA*

[b] *Department of Computer Science, University of Helsinki, P.O. Box 26, FIN-00014 Helsinki, Finland*

Received September 1990; revised December 1992

Abstract

A *satisficing search problem* consists of a set of probabilistic experiments to be performed in some order, seeking a satisfying configuration of successes and failures. The expected cost of the search depends both on the success probabilities of the individual experiments, and on the *search strategy*, which specifies the order in which the experiments are to be performed. A strategy that minimizes the expected cost is *optimal*. Earlier work has provided "optimizing functions" that compute optimal strategies for certain classes of search problems from the success probabilities of the individual experiments. We extend those results by providing a general model of such strategies, and an algorithm PAO that identifies an approximately optimal strategy when the probability values are not known. The algorithm first estimates the relevant probabilities from a number of trials of each undetermined experiment, and then uses these estimates, and the proper optimizing function, to identify a strategy whose cost is, with high probability, close to optimal. We also show that if the search problem can be formulated as an and-or tree, then the PAO algorithm can also "learn while doing", i.e. gather the necessary statistics while performing the search.

★ Some of this work was performed while the authors were at the University of Toronto, supported respectively by an operating grant from the National Science and Engineering Research Council of Canada, and by the Academy of Finland. The authors thank Dale Schuurmans, Tom Hancock and the anonymous referees for their useful comments on earlier versions of this paper. Preliminary versions of parts of the work have appeared in the conference reports [19] and [10].

* Corresponding author. E-mail: greiner@scr.siemens.com.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved

SSDI 0004-3702(95)00010-0

1. Introduction

Consider the following situation: there are two reliable tests for deciding whether an individual has hepatitis; one involves a blood test and the other a liver biopsy. Assuming there can be false negatives but no false positives, there are two "strategies" a doctor can follow to obtain a diagnosis. Using strategy $\Theta_1 = \langle \texttt{blood}, \texttt{liver} \rangle$, he would first perform the blood test and conclude the patient has hepatitis if that test is positive. If not, he would then examine the patient's liver, and conclude his diagnosis based on the result of that biopsy. The doctor's other option, strategy $\Theta_2 = \langle \texttt{liver}, \texttt{blood} \rangle$, performs these tests in the other order—first the liver test and then, only if necessary, the blood test.

Which strategy is better? Our goal is a strategy that will perform well in practice. To quantify the measurement, we assume there is a distribution of patients that the doctor will be asked to evaluate. We can then define a strategy's *expected cost* as the average cost required to perform these tests, averaged over the distribution of anticipated patients. Assuming, for now, that these tests (`blood` and `liver`) have the same cost, strategy Θ_1 is clearly better if the probability of a positive blood test (p_B) is larger than the probability of a positive liver test (p_L); otherwise, strategy Θ_2 is preferable.

Earlier research on this decision making model has produced a number of "optimizing functions" that each identify a strategy optimal for a specific testing situation, given the success probability values of the relevant experiments [6, 7, 18, 21, 22]. A limitation of these techniques, however, is that the probability values are in practice typically not known *a priori*. This paper specifies the number of trials of each experiment that are required to obtain estimates of these probability values that are good enough to identify a *nearly-optimal* strategy, with *high confidence*. It also addresses the complexities of observing this many trials.

Section 2 below first generalizes from the doctor's situation to a general class of arbitrary "decision structures" and defines strategies, and optimal strategies, for these structures. Section 3 then specifies the PAO algorithm, a general process that uses a set of observed trials of each experiment to identify a strategy whose cost is, with high probability, approximately optimal. The algorithm presumes the existence of an optimizing function for the class of search structures considered. When dealing with certain search structures, notably and-or trees, the PAO algorithm can "learn while doing", i.e. gather the necessary statistics while solving relevant performance tasks.

An extended version of this paper, available as a technical report [11], discusses several variants and applications of the basic algorithm presented here.

2. Framework

2.1. Decision structures

The doctor's task presented in Section 1 is a simple example of a *satisficing search problem* (term due to Simon and Kadane [21]), as his goal is to find a single satisfactory configuration of events: in this case, an informative combination of test results. Other

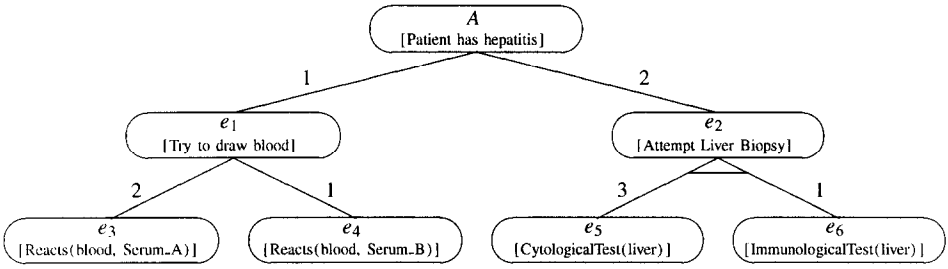


Fig. 1. An and-or tree representation of a decision structure G_1.

examples of such problems include, e.g., performing a sequence of tests to decide whether a product specimen is satisfactory [6], screening employment candidates for a position [6], competing for prizes at a quiz show [6], mining for gold buried in treasure chests [21], and performing inference in simple expert systems [10,22]. In general, such tasks may involve searching through general "decision structures", which can involve an arbitrary number of experiments, constrained by various precedence constraints.

And-or decision trees: More general versions of this diagnostic task can be represented by and-or decision trees, such as G_1 in Fig. 1. Here, the nodes $\{A, e_1, \ldots, e_6\}$ correspond to experiments, and the arcs encode the precedence relationships—e.g., the doctor cannot perform experiment e_3 (test whether the patient's blood reacts with a particular serum) until he has performed experiment e_1 (attempted to draw blood from the patient) and moreover, found that e_1 succeeds. The experiment associated with the A node is formally degenerate, i.e. it is guaranteed to succeed. The set of arcs descending from a given node can be either disjunctive, or conjunctive (here indicated by a horizontal line, e.g., connecting the e_2–e_5 arc to the e_2–e_6 arc). Hence, the graph in Fig. 1 states that the patient has hepatitis iff the condition $[e_1 \wedge (e_3 \vee e_4)] \vee [e_2 \wedge e_5 \wedge e_6]$ on the experiments holds. The number near each arc designates the cost of traversing that arc—hence it costs 1 unit to reduce the top e_0 node to the e_1 subgoal (draw blood), and 2 more units to further reduce e_1 to e_3 (test the blood against serum_A), and so forth. The incremental cost of performing each experiment is the sum of the costs of the *additional* arcs that must be traversed. (This cost specification means such trees cannot always be "collapsed" to simpler two-level trees.)

Beyond and-or trees: The general class of decision structures we shall consider is strictly more general than and-or trees. First, and-or trees can encode only simple formulae, which can include each experiment only once, and whose connections are only "and"s and "or"s. In general, we may want to express more complicated interrelationships of the experiments; e.g., the XOR of m experiments, or "at least 3 of 5 specified experiments". Second, and-or trees only permit relatively simple precedence relationships; in general, we may want to specify that an experiment can only be performed if some complicated boolean combination of other experiments has succeeded or failed. Third, and-or trees use a restricted form of cost function, in which the incremental cost of performing experiment e can depend only on which other experiments have been performed. In general, we may want the cost to depend also on whether e, and/or vari-

ous prior experiments, have been successful. There are also situations which require yet more complicated ways of computing the incremental cost of performing a particular experiment; see the extended paper [11].

To accommodate these extensions, we define a more general class of "decision structures". A decision structure can involve an arbitrary set of experiments $W = \{e_i\}_{i=1}^n$, with general precedence constraints that can prevent an experiment from being performed until after certain other specified experiments have been performed with the specified (success or failure) result. The overall test result (e.g., whether the patient has hepatitis) can correspond to an arbitrary boolean combination of the successes and failures of any subset of these experiments, and the costs of performing a sequence of experiments can be given by an arbitrary nondecreasing function. This leads to the "decision structures" defined below.

Notation: Given two sequences, $\sigma = \langle \sigma_1, \ldots, \sigma_n \rangle$ and $\tau = \langle \tau_1, \ldots, \tau_m \rangle$, let $\sigma \cdot \tau$ refer to the sequence formed by concatenating σ and τ—i.e. $\sigma \cdot \tau = \langle \sigma_1, \ldots, \sigma_n, \tau_1, \ldots, \tau_m \rangle$. The definition is extended to the case where σ or τ are single elements in the obvious manner. A sequence σ is a *subsequence* of sequence τ, denoted $\sigma \sqsubseteq \tau$, if $\sigma_i = \tau_{h(i)}$ for all $i = 1, \ldots, n$, for some monotonically increasing function h. The empty sequence $\langle\rangle$ is trivially a subsequence of any sequence σ.

Definition 1 (*Decision structures*). A *decision structure* is a four-tuple $G = \langle W, F, R, c \rangle$ where

- $W = \{e_1, \ldots, e_n\}$ is a set of *experiments*.
- $F \subseteq [W \times \{+, -\}]^* \times W$ is a *precedence relation* that specifies which further experiments can be performed given the results of a previous sequence of experiments. (E.g., $F(\langle\rangle, e_1)$ means that e_1 may be performed initially; and $F(\langle\langle e_1+\rangle, \langle e_3-\rangle\rangle, e_2)$ means that experiment e_2 may be performed after e_1 has been performed and was successful, and then e_3 has been performed but was unsuccessful.)

The following conditions use the notion of a *legal labeled experiment sequence* (abbreviated "*lles*"). This is a sequence of the form $\langle\langle e_1 \pm_1 \rangle, \ldots, \langle e_k \pm_k \rangle\rangle$, where each $e_i \in W$, each $\pm_i \in \{+, -\}$, and no $e \in W$ appears more than once. Furthermore, the sequence must satisfy the precedence constraints specified by the F relation: a sequence $\ell = \langle\langle e_1 \pm_1 \rangle, \ldots, \langle e_k \pm_k \rangle\rangle$ is a lles only if $F(\langle\langle e_1 \pm_1 \rangle, \ldots, \langle e_{m-1} \pm_{m-1} \rangle\rangle, e_m)$ holds for all $m = 1, \ldots, k$. The collection of all such sequences is denoted $\mathcal{LLES}(G)$.

- $R : \mathcal{LLES}(G) \to \{\mathcal{S}, \mathcal{F}, \mathcal{U}\}$ is the *result* function that specifies whether a given legal labeled experiment sequence renders the overall test successful or not; i.e. R maps each lles to one of $\{\mathcal{S}, \mathcal{F}, \mathcal{U}\}$ (for Success, Failure, and Undecided). We require R to be monotonic, in the sense that $R(\sigma) = \mathcal{S} \Rightarrow R(\sigma \cdot \tau) = \mathcal{S}$ and $R(\sigma) = \mathcal{F} \Rightarrow R(\sigma \cdot \tau) = \mathcal{F}$ whenever σ and $\sigma \cdot \tau$ are lles.
- $c : \mathcal{LLES}(G) \to \mathbb{R}_0^+$ is the *cost* function that maps each lles to its nonnegative real cost. It is required to be nondecreasing: $c(\sigma \cdot \tau) \geqslant c(\sigma)$ whenever σ and $\sigma \cdot \tau$ are lles.

We let $\mathcal{DS}$ refer to the class of all such decision structures.

To illustrate these definitions: the diagnostic tree of Fig. 1 can be encoded as a structure $G_1 = \langle \{e_1, \ldots, e_6\}, F_1, R_1, c_1 \rangle$, where e.g. e_3 corresponds to "a patient's blood

reacts with Serum_A" and e_5 corresponds to "a patient's liver has a positive cytological test". The F_1 relation, encoding the precedence relationships, includes $F_1(\langle\rangle, e_1)$ and $F_1(\langle\rangle, e_2)$ to mean that e_1 or e_2 can be performed initially; and $F_1(\langle\langle e_1+\rangle\rangle, e_3)$ but not $F_1(\langle\langle e_1-\rangle\rangle, e_3)$ (respectively $F_1(\langle\langle e_1+\rangle\rangle, e_4)$ but not $F_1(\langle\langle e_1-\rangle\rangle, e_4)$) to indicate that e_3 (respectively, e_4) can be performed if and only if e_1 has already succeeded. The R_1 function includes $R_1(\langle\langle e_1+\rangle, \langle e_3+\rangle\rangle) = \mathcal{S}$, $R_1(\langle\langle e_1+\rangle, \langle e_4+\rangle\rangle) = \mathcal{S}$, $R_1(\langle\langle e_2+\rangle, \langle e_5+\rangle, \langle e_6+\rangle\rangle) = \mathcal{S}$ and so forth, as well as $R_1(\langle\langle e_1-\rangle, \langle e_2-\rangle\rangle) = \mathcal{F}$, and $R_1(\langle\langle e_1-\rangle, \langle e_2+\rangle\rangle) = \mathcal{U}$, etc. The cost function c_1 encodes the incremental cost of performing any sequence of experiments: for instance, $c_1(\langle\langle e_1+\rangle, \langle e_3+\rangle, \langle e_4-\rangle\rangle) = 1 + 2 + 1 = 4$ and $c_1(\langle\langle e_1+\rangle, \langle e_3-\rangle, \langle e_2+\rangle, \langle e_6+\rangle\rangle) = 1 + 2 + 2 + 1 = 6$.

Notice that for standard graph-like decision structures, whenever an example becomes reachable, it stays so until performed, i.e. $F(\sigma, e)$ implies $F(\tau, e)$ whenever σ is a subsequence of τ and τ is a lles that does not include experiment e. For instance, both conditions $F_1(\langle\langle e_1-\rangle\rangle, e_2)$ and $F_1(\langle\langle e_1+\rangle\rangle, e_2)$ follow from the condition $F_1(\langle\rangle, e_2)$. When there is a unique minimal reachability condition for every experiment, we say that the structure is "tree-like". Formally, we define:

Definition 2 (*Tree-like decision structures*). A decision structure $G = \langle W, F, R, c\rangle$ is *tree-like* if we can identify with each experiment $e \in W$ a single minimal lles, denoted $\mathrm{path}(e)$, that encodes the necessary and sufficient conditions for reaching e; i.e.

$$\forall e \in W\ \exists \mathrm{path}(e) \in \mathcal{LLES}(G)\ \forall \sigma \in \mathcal{LLES}(G)\ F(\sigma, e) \iff [\mathrm{path}(e) \sqsubseteq \sigma].$$

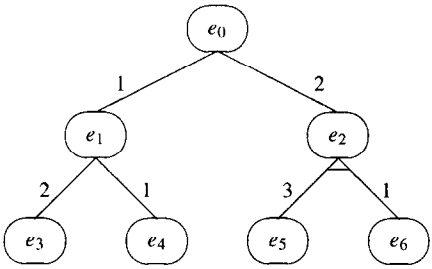
The class of all tree-like decision structures is denoted $\mathcal{TDS}$.

When the structure G represents an and-or decision tree, the lles $\mathrm{path}(e)$ corresponds to the unique path leading to the experiment e in the tree: e.g., as the path to e_4 in G_1 goes through e_0 and e_1, $\mathrm{path}(e_4) = \langle\langle e_0+\rangle, \langle e_1+\rangle\rangle$.

2.2. Satisficing search strategies

A "search strategy" for a satisficing search problem specifies the order of traversal through the associated decision structure—in the sample application G_1 of Fig. 1, it tells the doctor when to perform which tests to determine whether the patient has hepatitis.

A strategy can be represented as a binary tree; for example the tree shown on the right side of Fig. 2 represents one possible strategy Θ_1 for the decision structure G_1. Each internal node in the strategy tree is labeled with an experiment that corresponds to some node in the decision structure. The strategy specifies the sequence of experiments to be performed in any given situation. For example, the Θ_1 strategy first performs the experiment e_1 associated with Θ_1's root u_1. If e_1 succeeds, Θ_1 then follows the +-labeled arc to the strategy subtree rooted in the e_3-labeled node, and performs e_3. If that test succeeds, Θ_1 advances up to the $\mathcal{S}$-labeled node, signifying that Θ_1 terminates with success. Alternatively, if e_3 fails, Θ_1 then follows the −-labeled arc, descending to the tree rooted in the e_4-labeled node, then performs e_4, and so forth. A general definition of this process is as follows:



In each node of the strategy tree below, the experiment e_i associated to the node is indicated together with the name of the node, u_k.

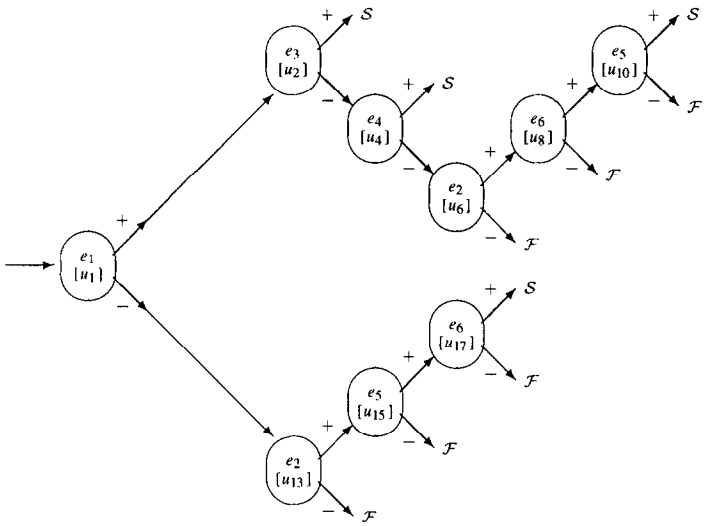


Fig. 2. Decision structure G_1 (above) and an associated strategy tree Θ_1 (below).

Definition 3 (*Search strategies*). A *strategy* for a decision structure $G = \langle W, F, R, c\rangle$ is a node- and arc-labeled binary tree $\Theta = \langle N, A, l_N, l_A\rangle$, where N is the set of nodes and $A \subseteq N \times N$ is the set of arcs connecting nodes to their descendants. The node-labeling l_N maps each internal node $n \in N$ in the tree to an experiment $e \in W$, and each leaf node to either $\mathcal{S}$ or $\mathcal{F}$. The arc-labeling l_A maps each arc $a \in A$ to either $+$ or $-$. Each internal node must have exactly two descending arcs, one labeled $+$ and the other $-$.

A *path* π in Θ is an alternating sequence of nodes and arcs leading from the root of the tree to a leaf—i.e. a sequence of the form $\pi = \langle n_1, a_{1,2}, n_2, a_{2,3}, \ldots, a_{k-1,k}, n_k\rangle$ where each $n_i \in N$, and each $a_{i,i+1} = \langle n_i, n_{i+1}\rangle \in A$. Each such path has an associated labeled experiment sequence $l(\pi) = \langle\langle l_N(n_1) l_A(a_{1,2})\rangle, \ldots, \langle l_N(n_{k-1}) l_A(a_{k-1,k})\rangle\rangle$. For Θ to be a proper strategy for G, the following conditions must be fulfilled by each path $\pi = \langle n_1, a_{1,2} \ldots, n_k\rangle$ in Θ:

(1) $l(\pi) \in \mathcal{LLES}(G)$ (i.e. $l(\pi)$ must be a legal labeled experiment sequence);

(2) $l_N(n_k) = R(l(\pi)) \in \{\mathcal{S}, \mathcal{F}\}$ (i.e. the label of the final node must be either "success" or "failure"); and
(3) $R(l_{1,\ldots,j}(\pi)) = \mathcal{U}$ for all $j < k$, where $l_{1,\ldots,j}(\pi) = \langle\langle l_N(n_1) l_A(a_{1,2})\rangle, \ldots, \langle l_N(n_j) l_A(a_{j,j+1})\rangle\rangle$ is the sequence of the first j elements of $l(\pi)$ (i.e. no proper prefix of a path can be conclusive).

We let path(Θ) refer to the set of all such proper paths in the strategy Θ. We also let $\mathcal{SS}(G)$ refer to the set of all strategies defined for the decision structure G, and $\mathcal{SS}(\mathcal{DS}) = \{\mathcal{SS}(G) \mid G \in \mathcal{DS}\}$ refer to the class of all strategies for all decision structures.

For an illustration of these notions, see the strategy tree Θ_1 shown on the right side of Fig. 2. There are 10 paths in Θ_1, one corresponding to each leaf node (indicated by the letters $\mathcal{S}$ and $\mathcal{F}$ in the figure). For instance, one such path is $\pi_{16} = \langle u_1, a_{1,13}, u_{13}, a_{13,15}, u_{15}, a_{15,16}, u_{16}\rangle$, for which the corresponding lles is $l(\pi_{16}) = \langle\langle e_1 -\rangle, \langle e_2 +\rangle, \langle e_5 -\rangle\rangle$.

2.3. Optimal strategies

We wish to identify the best strategy for traversing a given decision structure, i.e. the strategy whose expected cost is minimal. As this depends on the success probabilities of the individual experiments, different strategies will be optimal for different distributions. To state this more precisely, we define:

Definition 4 (*Expected cost of a strategy*). Let Θ be a strategy for the decision structure $G = \langle W, F, R, c\rangle$, and $p : W \to [0,1]$ be a distribution function that maps each experiment to its success probability. The (*expected*) *cost* of strategy Θ relative to the distribution p, denoted $C_p(\Theta)$, is defined as the sum of the cost of each path in the strategy, weighted by its probability, i.e.

$$C_p(\Theta) = \sum_{\pi \in \mathrm{path}(\Theta)} p(l(\pi)) \times c(l(\pi)).$$

Here the probability of a path π is defined as $p(l(\pi)) = \prod_{\langle e_i \pm_i\rangle \in l(\pi)} p^{\pm_i}(e_i)$, where $p^{\pm}(e)$ is $p(e)$ if $\pm$ equals $+$, and $1 - p(e)$ if $\pm$ equals $-$.

Definition 5 (*Optimizing functions*). An *optimizing function* for a class of decision structures $\mathcal{D} \subseteq \mathcal{DS}$ is a function OSS that maps any decision structure $G = \langle W, F, R, c\rangle \in \mathcal{D}$, together with a distribution $p \in [0,1]^W$, to a strategy in $\mathcal{SS}(G)$ whose cost is minimal. That is,

$$\forall G \in \mathcal{D}\ \forall p \in [0,1]^W\ \forall \Theta \in \mathcal{SS}(G)\ C_p(\mathrm{OSS}(G,p)) \leqslant C_p(\Theta).$$

For brevity, when the decision structure G is understood from the context, we often denote by Θ_p the optimal strategy $\mathrm{OSS}(G,p)$ provided by the optimizing function for a given distribution p.

While these definitions assume that the experiments are independent of each other, both these definitions and the theorems below could be extended to handle more complicated situations.

Since we are only dealing with finite decision structures, optimal strategies can always be found by exhaustive search. Of course, exhaustive search is in general impractical, and if we are dealing with decision structures with concise encodings, such as and-or trees, the optimal strategies may not even have polynomial-size representations.

Nevertheless, optimal strategies can be determined in polynomial time in many interesting special cases. Garey [6] provided an algorithm for finding the optimal search strategy when the constraints can be represented as a regular "or tree" (i.e. no conjunctive subgoals and no multiple predecessors are allowed; cf. also [22]). Simon and Kadane [21] later extended this algorithm to deal with directed acyclic graphs in the special case where success at any intermediate node implies global success. (In dag's where global success requires reaching a specified goal node, the problem is NP-hard [8].) It is currently not known whether optimal strategies can be found in polynomial time for and-or trees. Some partial results on this question exist: for instance, Natarajan [18] presents an efficient algorithm for finding optimal "depth-first" search strategies in this case, and Smith [22] provides an algorithm for finding optimal "serial strategies". In the more general case of and-or dag's, the problem is NP-hard even when all the success probabilities are 1 [20].

3. The PAO algorithm

Each of the above-mentioned optimization algorithms assumes that the precise success probabilities of the experiments are known, which of course is not the case in most real-life situations. The best one can do then is to estimate these probabilities by observing a set of trials of the experiments, and then use these estimates to compute a near-optimal strategy. A potential pitfall in this approach, however, is that the strategies computed by any of the above algorithms are very sensitive to errors in the probability estimates: small changes in the estimates may lead to drastically different strategies. Fortunately, even though the choice of the actual strategy is very sensitive to estimation errors, the *cost* of the strategy obtained is not. This realization is one of the main contributions of this paper, as it means that we can use our estimates to obtain a near-optimal strategy.

Below, we describe an algorithm PAO that can be used in conjunction with any optimizing function OSS. Section 3.1 first formally defines the task of finding approximately optimal strategies and outlines the algorithm. The following sections then discuss the technical issues in more detail. First, in Section 3.2 we compute the *sample complexity* of this task: how many samples of each experiment are needed to guarantee, with a high level of confidence, that a strategy based on the resulting estimates will be close to optimal.

Section 3.3 then addresses a second problem: guaranteeing that the PAO algorithm will be able to obtain a sufficient number of samples of each experiment. The main complication arises from the precedence constraints. For example, in the context of our diagnostic example (Fig. 1), the sample complexity analysis may suggest that the

```
Algorithm PAO(G : DS, ε : ℝ⁺, δ : (0,1])
  p̂ ← GS(G, ε, δ)
      /* GS may call oracle O a polynomial number of times */
  Θ̂ ← OSS(G, p̂)
  Return Θ̂
End PAO
```

Fig. 3. Outline of the PAO algorithm.

doctor needs to obtain 100 samples of the test "`CytologicalTest(liver)`". This is impossible if he is never able to perform a biopsy on any patients; i.e. if the experiment e_2 in structure G_1 never succeeds. In Section 3.3 we provide a solution to this problem for general "tree-like" decision structures, but also observe that the task is intractable in general.

3.1. The PAO task

A PAO problem instance consists of: a decision structure $G = \langle W, F, R, c \rangle \in \mathcal{DS}$; a bound on the allowed excess $\varepsilon \in \mathbb{R}^+$; and the required confidence $\delta \in (0, 1]$. The algorithm also uses an oracle $\mathcal{O}$ that produces samples drawn at random from some fixed but unknown distribution.

For each instance, the PAO algorithm returns a strategy $\Theta_{pao} \in \mathcal{SS}(G)$, whose expected cost is, with high probability, close to optimal. Stated more precisely, let $\Theta_p = \mathrm{OSS}(G, p)$ be the optimal strategy for a true, but unknown, distribution p. Then with probability at least $1 - \delta$, the cost of the strategy Θ_{pao} is no more than ε higher than the cost of this optimal strategy, i.e.

$$\Pr\left[C_p(\Theta_{pao}) \leqslant C_p(\Theta_p) + \varepsilon\right] \geqslant 1 - \delta.$$

We split the PAO task into two subtasks: subroutine GS, which gathers the relevant statistics, and OSS, which uses those statistics to produce an appropriate strategy; see Fig. 3.

The GS subroutine takes as input the decision structure G and the parameters ε and δ; it computes how many samples are required, and makes the specified number of calls to the oracle $\mathcal{O}$ (specified below) to obtain them. The subroutine produces a vector of probability estimates, $\hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_n \rangle$, where each $\hat{p}_i$ is the estimate for the success probability of the ith experiment $e_i \in W$. (To simplify our description, we are assuming that we do not know *a priori* the success probabilities of any of the experiments. If we happen to know some of the values, we can simply use those values directly, and not bother with the estimation.) The PAO algorithm then concludes by running an appropriate optimizing function OSS on these estimated probabilities $\hat{p}$, instead of the unknown true values. We concentrate here on the sample-gathering part of the PAO algorithm, GS; for the OSS functions, we rely on the ones provided by earlier researchers.

3.2. Sample complexity

We first analyze the sample complexity of the PAO task in the simple case where we can always perform the experiments whose success probabilities we need to estimate. Here we assume access to an oracle $\mathcal{O}$ that will, upon request, produce a sample κ_j from the population, together with its complete labeling $\mathcal{L}(\kappa_j) = \langle \ell_1^j, \ldots, \ell_n^j \rangle$, where ℓ_i^j is 1 if κ_j passes experiment $e_i \in W$, and 0 otherwise. The GS routine performs a number M (specified below) calls to this $\mathcal{O}$ oracle, and returns a vector of probability estimates $\hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_n \rangle$, where each $\hat{p}_i = 1/M \sum_{j=1}^{n} \ell_i^j$; the OSS optimizing function then uses these values. We prove (Corollary A.1 in Appendix A) that the cost of the strategy $\Theta_{pao} = \mathrm{OSS}(G, \hat{p})$ is within ε of the optimal, with reliability at least $1 - \delta$, whenever

$$M = \left\lceil 2 \left(\frac{nC}{\varepsilon} \right)^2 \ln \frac{2n}{\delta} \right\rceil, \tag{1}$$

where C is the worst-case cost of performing any sequence of experiments in W.

In fact, we can improve on the constant C somewhat. Let us denote by $D(e)$ the maximal cost of any concluding sequence of experiments beginning with experiment e. Formally:

Definition 6. Let $G = \langle W, F, R, c \rangle$ be a decision structure. For each experiment $e \in W$ we define

$$D(e) = \max\{c(\alpha \cdot \langle e \pm \rangle \cdot \beta) - c(\alpha) \mid \alpha \cdot \langle e \pm \rangle \cdot \beta \in \mathcal{LLES}(G)\}.$$

(The $\pm$ above indicates that the max should range over both $+$ and $-$ values.)

We can then let $C = \max_{e \in W}\{D(e)\}$ be the maximum remainder cost starting with any experiment $e \in W$. These $D(e)$ values are quite easy to compute when the underlying decision structure is an and-or tree: here, $D(e) = C_{tot} - c(\mathrm{path}(e))$, where C_{tot} is the sum of the costs of all of the tree's arcs and $c(\mathrm{path}(e))$ is the cost of $\mathrm{path}(e)$, the unique path in the and-or tree that leads from the root to experiment e; see Definition 2. (For instance, in the tree G_1 of Fig. 2 we have $C_{tot} = 10$ and $c(\mathrm{path}(e_4)) = 2$, so $D(e_4) = 10 - 2 = 8$.)

The sample complexity bound (1) is derived in Appendix A as Corollary A.1 of a more general result that also takes into account the difficulty of labeling the samples (i.e. performing the experiments; see below). To very briefly outline the proof for this simple case: we first prove that after M samples, we are at least $1 - \delta$ confident that each probability estimate $\hat{p}_i$ is within $\varepsilon/2nC$ of the correct value p_i; we then show, based directly on the definition of the cost of a strategy and independent of which optimizing function is used, that this precision of the probability estimates suffices to guarantee that the cost of the obtained strategy is within ε of the optimal, i.e. that

$$\left[\forall i \; |p_i - \hat{p}_i| \leqslant \frac{\varepsilon}{2nC}\right] \Rightarrow |C_p(\Theta_{\hat{p}}) - C_p(\Theta_p)| \leqslant \varepsilon.$$

3.3. "Learning while doing" in tree-like structures

The simple PAO algorithm presented above assumes that the oracle $\mathcal{O}$ produces a complete labeling for each sample, i.e. it returns a complete vector $\mathcal{L}(\kappa) = \langle \ell_1, \ldots, \ell_n \rangle \in \{0, 1\}^n$ on each query. In practical situations, however, such an oracle will typically not be available. Instead, the learning system must collect the statistics it needs (i.e. the individual component ℓ_i values of $\mathcal{L}(\kappa)$) while watching a performance element perform its task, over a sufficiently large set of samples. In the context of our diagnostic example, the learning module would observe the doctor as he examines patients, recording how many of these patients pass the various tests. After gathering enough information, the learner would compute the approximately optimal strategy Θ_{pao}, instruct the doctor to use this Θ_{pao} strategy, and terminate itself.[1] We view this as a "learning while doing" protocol [16], as the overall system is performing useful work during the learning phase (here, examining patients).

From now on, we assume our oracle $\mathcal{O}$, when queried, provides only an unlabeled sample (e.g., a patient κ), rather than the full labelings of that sample, $\mathcal{L}(\kappa)$. In order to determine the value of any label ℓ_i on sample κ, the GS subroutine must then actually "reach" and perform experiment e_i on κ.

Computing these ℓ_i values is problematic when there are intermediate experiments: for instance, in the case of our decision structure G_1, the doctor cannot immediately determine whether a patient's blood will react to serum_A; he must first be able to draw blood from the patient. Hence, our learning system will be unable to estimate the probability of event "`blood reacts to serum_A`" if the doctor is never able to extract blood (i.e. if Pr[Draw blood] = 0).

Fortunately, there is a way around this problem. The critical observation is the following: let $\rho(e_i)$ be the probability of "reaching" an experiment e_i during the execution of a strategy. (This notion is defined formally below.) If $\rho(e_i)$ is very small, we will be unlikely to reach e_i and hence to obtain samples of this experiment. However, the smaller the value of $\rho(e_i)$, the less sensitive the cost of the optimal strategy is to the value of the success probability $p(e_i)$, which means that we also *need* fewer samples of e_i. In the limit, if there is *no* chance of reaching e_i (i.e. $\rho(e_i) = 0$), then we will also need no samples of it (i.e. OSS can produce an optimal strategy even if $|\hat{p}_i - p_i| = 1$).

Definition 7. Let $G = \langle W, F, R, c \rangle \in \mathcal{DS}$ be a decision structure, and $p : W \to [0, 1]$ a distribution function that maps each experiment to its success probability. For any strategy $\Theta = \langle N, A, l_N, l_A \rangle \in \mathcal{SS}(G)$, and any experiment $e \in W$, let $\rho(e, \Theta)$ be the probability that Θ will reach e, i.e.

$$\rho(e, \Theta) = \sum_{n : l_N(n) = e} p(l(\pi(n))),$$

[1] We are still considering only "one-shot learning", in which the learner sets the strategy only once, after the learning phase. We are not considering ways of modifying the strategy gradually over time to become incrementally better; but see [9]. Also, this issue differs from the "Exploration-Exploitation" trade-off discussed in the context of the Bandit problem (cf. [3, 17]) as we are not concerned with minimizing the cumulative cost of the learning and performance systems together, over an infinite sequence of samples.

where the sum is over nodes n in the strategy Θ labeled with e, $\pi(n)$ is the path in Θ that leads to n, and the probability of this path $p(l(\pi(n)))$ is as defined above in Definition 4. Finally, let $\rho(e) = \max\{\rho(e,\Theta) \mid \Theta \in \mathcal{SS}(G)\}$.

The formula for $\rho(e)$ reduces to a particularly simple form when the decision structure G is tree-like. In this case $\rho(e) = \prod_{i=1}^{k} p^{\pm_i}(f_i)$, where $\text{path}(e) = \langle\langle f_1 \pm_1\rangle, \ldots, \langle f_k \pm_k\rangle\rangle$ is the unique path that leads to e in G. For instance, in the G_1 decision structure of Fig. 1, we have $\text{path}(e_4) = \langle\langle e_0, +\rangle, \langle e_1, +\rangle\rangle$, and so $\rho(e_4) = p(e_0) \times p(e_1)$.

Now let $\Theta_p = \text{OSS}(G,p)$ be the actual optimal strategy based on the unknown correct probability vector $p = \langle p_1, \ldots, p_n\rangle$, and $\Theta_{\hat{p}} = \text{OSS}(G,\hat{p})$ be the strategy that our PAO algorithm will produce, based on the estimates the GS subroutine has obtained, $\hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_n\rangle$. We wish to bound the cost difference $C_p(\Theta_{\hat{p}}) - C_p(\Theta_p)$. The following lemma shows that, in place of obtaining precise estimates of the probabilities p_i, we need only ensure that the product $\rho(e_i) \times |p_i - \hat{p}_i|$ is small for each e_i.[2]

Lemma 8. *Let $G = \langle W, F, R, c\rangle$ be a decision structure with $|W| = n$ experiments. Let $p = \langle p_1, \ldots, p_n\rangle$ be a vector of success probabilities for W, and $\hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_n\rangle$ a vector of their estimates. Let the optimal search strategy for G with respect to p be $\Theta_p = \text{OSS}(G,p)$, and let $\Theta_{\hat{p}} = \text{OSS}(G,\hat{p})$ be the strategy based on the estimated probabilities. Then*

$$C_p(\Theta_{\hat{p}}) - C_p(\Theta_p) \leqslant 2\sum_{i=1}^{n} D(e_i) \times \rho(e_i) \times |p_i - \hat{p}_i|.$$

A further complication now arises from the fact that the $\rho(e_i)$ values actually depend on the unknown true distribution p. Fortunately, we can also approximate these values as we are obtaining the estimates of the p_i. In essence, we need only "aim for e_i" a certain number of times: each time we reach e_i, we improve our estimate of p_i (i.e. reduce the $|\hat{p}_i - p_i|$ "error bars") and each time our path to e_i is blocked, we can, with confidence, reduce the value of $\rho(e_i)$.

The rest of this subsection first shows how to estimate the products $\rho(e_i) \times |p_i - \hat{p}_i|$ in tree-like decision structures, then discusses the difficulties in computing near-optimal strategies in more general structures.

Dealing with tree-like decision structures: Given an experiment e in a tree-like decision structure G, recall that $\text{path}(e) = \langle\langle e_1 \pm_1\rangle, \langle e_2 \pm_2\rangle, \ldots, \langle e_k \pm_k\rangle\rangle$ is the unique minimal lles that determines when e can be performed. We say that a strategy $\Theta \in \mathcal{SS}(G)$ is a *direct strategy* for e if it contains this lles as an initial segment, in the sense that the root of Θ is labeled with the experiment e_1, and its $\pm_1$-labeled arc (i.e. the $+$-labeled arc if $\pm_1$ equals $+$, and the $-$-labeled arc if $\pm_1$ equals $-$) descends to a node labeled with the experiment e_2, and the $\pm_2$-labeled arc from that node descends to a node labeled with e_3, and so on, down to a node labeled e_k, whose $\pm_k$-labeled arc leads to a node labeled with e. We denote the class of direct strategies for an experiment e by $\mathcal{SS}(e)$. As an example, the strategy Θ_1 shown in Fig. 2 goes directly to e_1 and hence $\Theta_1 \in \mathcal{SS}(e_1)$;

[2] Appendix A contains the proofs for all lemmata, theorems and corollaries presented in the text.

```
Algorithm GS(G: TDS, ε: ℝ⁺, δ: (0,1])
  ForEach  e ∈ W  do
    Find some Θ_e ∈ SS(e)
    tot(e) ← 0
    suc(e) ← 0
                  ⎧ ⌈2 (nD(e)/ε)² ln(2n/δ)⌉                  if  F(⟨⟩, e)
    m(e) ←        ⎨
                  ⎩ ⌈2(1 + ε/(nD(e))) (nD(e)/ε)² ln(4n/δ)⌉   if  ¬F(⟨⟩, e)
  End ForEach
  While  ∃e such that m(e) > 0  do
    Get sample  κ  from oracle  O
    Execute  strategy Θ_e  on sample κ
      After performing each experiment e_j:
        tot(e_j) ← tot(e_j) + 1
        m(e_j) ← m(e_j) − 1
        If  e_j succeeds: suc(e_j) ← suc(e_j) + 1
        If  e_j's result (success or failure) means
          e cannot be reached: m(e) ← m(e) − 1
  End While
  ForEach  e_i ∈ W  do
                  ⎧ suc(e_i)/tot(e_i)   if tot(e_i) > 0
    p̂(e_i) ←      ⎨
                  ⎩ 1/2                 otherwise
  End ForEach
  Return   p̂ = ⟨p̂(e_1), ..., p̂(e_n)⟩
End GS
```

Fig. 4. A GS algorithm for tree-like decision structures.

it also contains the direct route to e_3 as an initial segment, and hence $\Theta_1 \in \mathcal{SS}(e_3)$. On the other hand, the strategy "digresses" to consider e_3 before e_4, and so $\Theta_1 \notin \mathcal{SS}(e_4)$; similarly $\Theta_1 \notin \mathcal{SS}(e_2)$ as Θ_1 considers e_1 before e_2.

The GS algorithm shown in Fig. 4 can deal with any tree-like decision structure G. The algorithm first identifies a direct strategy $\Theta_e \in \mathcal{SS}(e)$ for each $e \in W$. (There can in general be many such strategies, performing different experiments outside their common initial path to e; this paper does not consider how to choose between the alternatives. Nor does it consider the cost of identifying any of these strategies, except to observe that for e.g. and-or trees they can be constructed quite efficiently, directly from the tree structure.) After selecting this set of strategies, GS associates three counters with each experiment, $tot(e_i)$, $suc(e_i)$ and $m(e_i)$, that will record, respectively, the number of times experiment e_i has been performed, the number of times e_i succeeded, and the number of attempts that remain to be performed. As it processes the instances, GS updates each of these counters by: incrementing $tot(e_i)$ each time GS performs

experiment e; incrementing $suc(e_i)$ each time the experiment e succeeds; and decrementing $m(e_i)$ each time GS has attempted to reach experiment e_i either by performing e_i, or by using the strategy Θ_{e_i} but failing to reach e_i.

The remaining challenge is to identify when to use which strategy. (Clearly GS will not, in general, be able to observe enough trials of the different experiments if it uses the same strategy throughout.) On each sample, GS first identifies the needy experiments, i.e. those e's for which $m(e) > 0$. If there are none, then GS has collected enough samples, and so can terminate, returning the obtained vector of estimates $\hat{p}$, which PAO will pass to the OSS algorithm. Otherwise, GS selects one of the needy experiments e, and executes the associated strategy Θ_e.

Notice that GS decrements at least one $m(e)$ counter on each sample, viz. the one associated with the experiment e to which it is currently aiming. Hence, after at most $\sum_{e \in W} m_0(e)$ samples (where the $m_0(e)$ are the initial values of the counters), all of the $m(e)$ counters will be zero and GS will terminate; it therefore requires only a polynomial number of samples. (The algorithm may of course use far fewer samples, as most Θ_e strategies will reduce the $m(e_j)$ values for several different experiments e_j. GS can also be changed to decrease the counters of all experiments e' that are deemed unreachable in the process of following Θ_e.) The following lemma characterizes the behavior of the algorithm:

Lemma 9. *Let $G = \langle W, F, R, c \rangle$ be a tree-like decision structure with $|W| = n$ experiments, and let $p = \langle p_1, \ldots, p_n \rangle$ be a vector of success probabilities for the experiments. Furthermore, let $\varepsilon, \delta > 0$ be any given constants, and let $\hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_n \rangle$ be a vector of probability estimates computed by the GS algorithm of Fig. 4. Then*

$$\forall e_i \in W \ \Pr\left[D(e_i) \times \rho(e_i) \times |p_i - \hat{p}_i| \geqslant \frac{\varepsilon}{2n}\right] \leqslant \frac{\delta}{n}.$$

(While our analysis uses the $\rho(e_i)$ values, notice that the GS algorithm never actually computes them.) Combining the results of Lemmas 8 and 9, we obtain the following theorem:

Theorem 10. *Let $G = \langle W, F, R, c \rangle$ be a tree-like decision structure with $|W| = n$ experiments, and let $p = \langle p_1, \ldots, p_n \rangle$ be a vector of success probabilities for the experiments. Furthermore, let $\varepsilon, \delta > 0$ be any given constants, and let $\Theta_{pao} = \mathrm{PAO}(G, \varepsilon, \delta)$ be the strategy produced by the* PAO *algorithm using the GS subroutine of Fig. 4. Then, with probability at least $1 - \delta$, $C_p(\Theta_{pao}) - C_p(\Theta_p) \leqslant \varepsilon$, where $\Theta_p = \mathrm{OSS}(G, p)$ is the optimal strategy for probability vector p.*

Beyond tree-like decision structures: While the specific GS algorithm presented above applies only to tree-like decision structures, there can be other related algorithms that can learn strategies for other decision structures. The main challenge is in estimating $\rho(e_i)$, as required to bound the product $\rho(e_i) \times |p_i - \hat{p}_i|$, which is complicated by the fact that there can be many distinct ways of reaching an experiment in a general decision structure.

To address this task, recall from Definition 7 that $\rho(e)$ is the *maximum* probability of reaching the experiment e, where the maximum is taken over all possible strategies. We

can always approximate this value by first estimating $\rho(e, \Theta)$ for every possible strategy Θ, and then taking the maximum of these values: if each estimate $\hat{\rho}(e, \Theta)$ is within ε of $\rho(e, \Theta)$ with probability at least $1 - \delta/|\mathcal{SS}(G)|$, then the value $\hat{\rho}(e) = \max\{\hat{\rho}(e, \Theta) \mid \Theta \in \mathcal{SS}(G)\}$ will be within ε of $\rho(e) = \max\{\rho(e, \Theta) \mid \Theta \in \mathcal{SS}(G)\}$ with probability at least $1 - \delta$. Even though the number of strategies in $\mathcal{SS}(G)$ for a given decision structure G can be exponential in the size of G, there can be ways of exploiting the structure of G, and hence of $\mathcal{SS}(G)$, to limit the number of $\hat{\rho}(e, \Theta)$ values that need to be considered. From this point of view, the GS algorithm for tree-like structures is based on the observation that for any tree-like structure G, the direct strategies $\Theta_e \in \mathcal{SS}(e)$ necessarily yield the largest values of $\rho(e, \Theta_e)$ for any experiment e. In fact, one can use the same GS algorithm whenever it is possible to identify each experiment e with a strategy Θ_e for which $\rho(e, \Theta_e) = \rho(e)$.

This is not always straightforward. The extended paper [11] includes an algorithm that uses dynamic programming techniques to sequentially estimate the probabilities of each "layer" of certain types of decision structures. Unfortunately, the following general result shows that the computational complexity of any such algorithm is likely to be exponential in the number of experiments.

Theorem 11. *Assume* RP $\neq$ NP. (RP *is the class of problems solvable by probabilistic polynomial time algorithms with one-way error, cf.* [12].) *Then there is no probabilistic polynomial time algorithm, and consequently no deterministic polynomial time algorithm that, given a decision structure* $G = \langle W, F, R, c \rangle$, *an experiment* $e \in W$, *a distribution function* $p : W \to [0, 1]$, *and parameters* $\varepsilon, \delta > 0$, *can estimate the value* $\rho(e)$ *to within* ε *with probability at least* $1 - \delta$.

4. Conclusion

The results presented in this paper have been motivated by, and extend, various other lines of research. The underlying objective of finding a provably good search strategy comes from the work on optimal satisficing search strategies [2, 6, 18, 21, 22]. Each of these earlier papers considered some specifically defined class of decision structures and, moreover, required the user to supply precise success probability values for the experiments. Our work extends this body of research in three ways. First, we have defined a general framework of "decision structures" and "search strategies", which encompasses and generalizes the models used before. Second, we have analyzed, in this very general setting, the sensitivity of optimal search strategies to errors in the probability estimates. Third, we have provided an efficient algorithm for finding good estimates of the probability values in the case of tree-like decision structures, and proved that (unless RP = NP) there can be no efficient algorithm for this task for general structures.

Our approach also resembles the work on speed-up learning (including both "explanation-based learning" [5, 14, 15] and "chunking" [13]), as it uses previous solutions to suggest a way of improving the speed of a performance system. Most speed-up learning systems, however, use only a single example to suggest an improvement; we extend those works by showing how to use a *set* of samples and by describing, furthermore, the

exact number of samples required. Also, while most speed-up learning systems are based on purely heuristic considerations, we use mathematically sound techniques to guarantee that our new strategies will be close to optimal, with provably high probability.

Finally, this work derives many of its mathematical methods, as well as its title, from the field of "probably approximately correct learning" [23]. We hope to have enriched this field by providing an application of the PAC framework outside of its traditional setting of concept learning.

Appendix A. Proofs

This appendix contains the proofs of the results mentioned in the body of the paper.

Lemma 8. *Let $G = \langle W, F, R, c\rangle$ be a decision structure with $|W| = n$ experiments. Let $p = \langle p_1, \ldots, p_n\rangle$ be a vector of success probabilities for G, and $\hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_n\rangle$ a vector of their estimates. Let the optimal search strategy for G with respect to p be $\Theta_p = \mathrm{OSS}(G, p)$, and let $\Theta_{\hat{p}} = \mathrm{OSS}(G, \hat{p})$ be the strategy based on the estimated probabilities. Then*

$$C_p(\Theta_{\hat{p}}) - C_p(\Theta_p) \leqslant 2\sum_{i=1}^{n} D(e_i) \times \rho(e_i) \times |p_i - \hat{p}_i|.$$

Proof. Given the vectors p and $\hat{p}$, let $p^{(i)}$ denote the vector $\langle \hat{p}_1, \ldots, \hat{p}_i, p_{i+1}, \ldots, p_n\rangle$, and as special cases, $p^{(0)} = p$ and $p^{(n)} = \hat{p}$. We shall prove below that for any strategy Θ for G, and for every $i = 0, \ldots, n-1$,

$$\left|C_{p^{(i)}}(\Theta) - C_{p^{(i+1)}}(\Theta)\right| \leqslant D(e_i) \times \rho(e_i) \times |p_i - \hat{p}_i|, \tag{A.1}$$

which implies that

$$\begin{aligned}
&|C_p(\Theta) - C_{\hat{p}}(\Theta)| \\
&\quad \leqslant \sum_{i=1}^{n} \left|C_{p^{(i)}}(\Theta) - C_{p^{(i-1)}}(\Theta)\right| \leqslant \sum_{i=1}^{n} D(e_i) \times \rho(e_i) \times |\hat{p}_i - p_i|.
\end{aligned}$$

Applying this bound to the strategies Θ_p and $\Theta_{\hat{p}}$, and noting that by optimality $C_{\hat{p}}(\Theta_{\hat{p}}) - C_{\hat{p}}(\Theta_p) \leqslant 0$, then yields the desired result:

$$\begin{aligned}
&C_p(\Theta_{\hat{p}}) - C_p(\Theta_p) \\
&\quad = [C_p(\Theta_{\hat{p}}) - C_{\hat{p}}(\Theta_{\hat{p}})] + [C_{\hat{p}}(\Theta_{\hat{p}}) - C_{\hat{p}}(\Theta_p)] + [C_{\hat{p}}(\Theta_p) - C_p(\Theta_p)] \\
&\quad \leqslant \left[\sum_{i=1}^{n} D(e_i) \times \rho(e_i) \times |p_i - \hat{p}_i|\right] + 0 + \left[\sum_{i=1}^{n} D(e_i) \times \rho(e_i) \times |\hat{p}_i - p_i|\right] \\
&\quad = 2\sum_{i=1}^{n} D(e_i) \times \rho(e_i) \times |\hat{p}_i - p_i|.
\end{aligned}$$

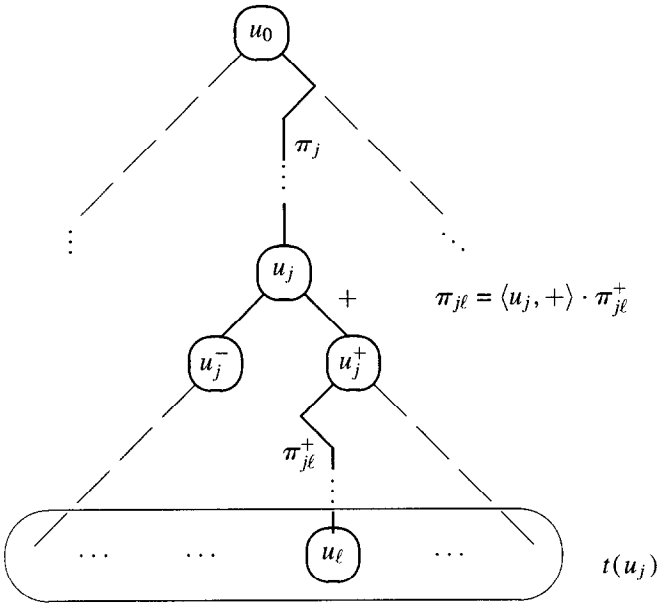


Fig. A.1. Illustration of notation.

In proving inequality (A.1), we shall make use of the following notation (cf. Fig. A.1): given any node u_j in a strategy Θ, let π_j denote the path leading from the root of Θ to u_j. For any such path π_j, we denote the associated cost $c(l(\pi_j))$ briefly by $c(\pi_j)$. (Here, we have extended the $l(\cdot)$ function to partial sequences: u_j, the final entry in π_j, does not have to be a leaf node in the strategy tree.) We also extend the cost function to incomplete paths in the strategy tree by defining

$$\begin{aligned} &c(\langle u_i, a_{i,i+1}, u_{i+1}, \ldots, u_k \rangle) \\ &\quad = c(\langle u_0, a_{0,1}, u_1, \ldots, u_i, a_{i,i+1}, u_{i+1}, \ldots, u_k \rangle) - c(\langle u_0, a_{0,1}, u_1, \ldots, u_i \rangle), \end{aligned}$$

where u_0 is the root node of the strategy tree and $\langle u_0, a_{0,1}, u_1, \ldots, a_{k-1,k}, u_k \rangle$ is any connected path through the tree; furthermore, for single nodes we define $c(u_i) = c(\langle u_i \rangle)$.

Let u_j^+ be the $+$-descendant of node u_j, and let $t(u_j)$ denote the set of leaf nodes below u_j. For each $u_l \in t(u_j)$, let π_{jl} denote the path from u_j to u_l, and if $u_l \in t(u_j^+)$, let π^+_{jl} denote the path from node u_j^+ to u_l. Let $C_p(\Theta_j^+)$ denote the "expected cost of the $+$-subtree of u_j":

$$C_p(\Theta_j^+) = c(u_j^+) + \sum_{u_l \in t(u_j^+)} p(\pi^+_{jl}) c(\pi^+_{jl}).$$

Analogous definitions hold for u_j^-, $t(u_j^-)$, $p(\pi^-_{jl})$, and $C_p(\Theta_j^-)$.

For a given experiment $e_i \in W$, let $N(e_i) = \{u_j\}_{j=1,\ldots,k}$ be the set nodes in Θ labeled with e_i. Let $\Theta \mid e_i$ be the subtree within Θ consisting of the paths from the root of Θ through a node in $N(e_i)$ down to a leaf; and let $\Theta \mid \bar{e}_i$ be the subtree consisting of

the other paths. Notice that $C_p(\Theta) = C_p(\Theta \mid e_i) + C_p(\Theta \mid \bar{e}_i)$. We may partition the paths in Θ^+ according to which of the u_j nodes each passes through (recall that in any strategy Θ, an experiment e can occur at most once on any path from the root to a leaf node), and use this representation to obtain a very simple formula expressing the influences of e_i's success probability p_i on the function $C_p(\Theta \mid e_i)$ (and hence on $C_p(\Theta)$):

$$
\begin{aligned}
&C_p(\Theta \mid e_i) \\
&= \sum_{\pi \in \mathrm{path}(\Theta^+)} p(\pi)c(\pi) = \sum_{j=1}^{k} \sum_{u_l \in t(u_j)} p(\pi_l)c(\pi_l) \\
&= \sum_{j=1}^{k} \left[\sum_{u_l \in t(u_j^+)} p(\pi_l)c(\pi_l) + \sum_{u_l \in t(u_j^-)} p(\pi_l)c(\pi_l) \right] \\
&= \sum_{j=1}^{k} \left[\sum_{u_l \in t(u_j^+)} p(\pi_j)p_i p(\pi_{jl}^+)\Big(c(\pi_j) + c(u_j^+) + c(\pi_{jl}^+) \Big) \right. \\
&\qquad \left. + \sum_{u_l \in t(u_j^-)} p(\pi_j)(1-p_i)p(\pi_{jl}^-)\Big(c(\pi_j) + c(u_j^-) + c(\pi_{jl}^-) \Big) \right] \\
&= \sum_{j=1}^{k} p(\pi_j) \left[p_i \sum_{u_l \in t(u_j^+)} p(\pi_{jl}^+)\Big(c(\pi_j) + c(u_j^+) + c(\pi_{jl}^+) \Big) \right. \\
&\qquad \left. + (1-p_i) \sum_{u_l \in t(u_j^-)} p(\pi_{jl}^-)(c(\pi_j) + c(u_j^-) + c(\pi_{jl}^-)) \right] \\
&= \sum_{j=1}^{k} p(\pi_j) \left[p_i \Bigg((c(\pi_j) + c(u_j^+)) \sum_{u_l \in t(u_j^+)} p(\pi_{jl}^+) + \sum_{j=1}^{k} p(\pi_{jl}^+)c(\pi_{jl}^+) \Bigg) \right. \\
&\qquad + (1-p_i) \Bigg((c(\pi_j) + c(u_j^-)) \sum_{u_l \in t(u_j^-)} p(\pi_{jl}^-) \\
&\qquad \left. + \sum_{j=1}^{k} p(\pi_{jl}^-)c(\pi_{jl}^-) \Bigg) \right] \\
&= \sum_{j=1}^{k} p(\pi_j) \left[p_i \Bigg(c(\pi_j) + c(u_j^+) + \sum_{j=1}^{k} p(\pi_{jl}^+)c(\pi_{jl}^+) \Bigg) \right. \\
&\qquad \left. + (1-p_i) \Bigg(c(\pi_j) + c(u_j^-) + \sum_{j=1}^{k} p(\pi_{jl}^-)c(\pi_{jl}^-) \Bigg) \right]
\end{aligned}
$$

$$= \sum_{j=1}^{k} p(\pi_j) \left[p_i(c(\pi_j) + C_p(\Theta_j^+)) + (1 - p_i)(c(\pi_j) + C_p(\Theta_j^-)) \right]$$

$$= \sum_{j=1}^{k} p(\pi_j) \left(c(\pi_j) + p_i C_p(\Theta_j^+) + (1 - p_i) C_p(\Theta_j^-) \right).$$

(Near the end of the proof we have simplified the formulas using the fact that a complete system of elementary probabilities sums to 1: in this case

$$\sum_{u_l \in t(u_j^+)} p(\pi_{jl}^+) = \sum_{u_l \in t(u_j^-)} p(\pi_{jl}^-) = 1.)$$

An analogous formula can be derived for the cost function $C_{\hat{p}_i}(\Theta \mid e_i)$, where $\hat{p}_i = \langle p_1, \ldots, p_{i-1}, \hat{p}_i, p_{i+1}, \ldots, p_n \rangle$ is the probability vector that differs from p only in the ith value. Using now the facts that $C_p(\Theta \mid \bar{e}_i) = C_{\hat{p}_i}(\Theta \mid \bar{e}_i)$ and $C_p(\Theta_j^+) = C_{\hat{p}_i}(\Theta_j^+)$, as none of the substrategies $\Theta \mid \bar{e}_i$, Θ_j^+, $j \neq i$, involve the experiment e_i, we obtain the following bound:

$$\begin{aligned}
|C_p(\Theta) - C_{\hat{p}_i}(\Theta)| &= \left| \left(C_p(\Theta \mid e_i) + C_p(\Theta \mid \bar{e}_i)\right) - \left(C_{\hat{p}_i}(\Theta \mid e_i) + C_{\hat{p}_i}(\Theta \mid \bar{e}_i)\right) \right| \\
&= \left| \left(C_p(\Theta \mid e_i) - C_{\hat{p}_i}(\Theta \mid e_i)\right) + \left(C_p(\Theta \mid \bar{e}_i) - C_{\hat{p}_i}(\Theta \mid \bar{e}_i)\right) \right| \\
&= \left| \sum_{j=1}^{k} p(\pi_j) \left((p_i - \hat{p}_i) C_p(\Theta_j^+) + (\hat{p}_i - p_i) C_p(\Theta_j^-) \right) + 0 \right| \\
&= \left| \sum_{j=1}^{k} p(\pi_j)(p_i - \hat{p}_i) \left(C_p(\Theta_j^+) - C_p(\Theta_j^-) \right) \right| \\
&\leqslant \sum_{j=1}^{k} p(\pi_j) \times |p_i - \hat{p}_i| \times \max_{l(u_j)=e_i} \left| C_p(\Theta_j^+) - C_p(\Theta_j^-) \right| \\
&\leqslant \rho(e_i) \times |p_i - \hat{p}_i| \times \max_{l(u_j)=e_i} \max \left\{ C_p(\Theta_j^+), C_p(\Theta_j^-) \right\}.
\end{aligned}$$

The last line of the calculation uses the facts that $C_p(\cdot) \geqslant 0$ and that $\sum_{j=1}^{k} p(\pi_j) = \rho(e_i, \Theta)$ is the probability that this strategy Θ will reach e_i, and hence is bounded by $\rho(e_i) = \max\{\rho(e_i, \Theta) \mid \Theta \in \mathcal{SS}(G)\}$.

All that remains is to show that the value $\max_{l(u_j)=e_i}\{C_p(\Theta_j^+), C_p(\Theta_j^-)\}$ is bounded by $D(e_i)$. To see this, consider any u_j such that $l(u_j) = e_i$. Then

$$\begin{aligned}
C_p(\Theta_j^+) &= c(u_j^+) + \sum_{u_l \in t(u_j^+)} p(\pi_{jl}^+) c(\pi_{jl}^+) \\
&\leqslant c(u_j^+) + \left(\max_{u_l \in t(u_j^+)} c(\pi_{jl}^+) \right) \sum_{u_l \in t(u_j^+)} p(\pi_{jl}^+)
\end{aligned}$$

$$
\begin{aligned}
&= c(u_j^+) + \max_{u_l \in t(u_j^+)} c(\pi_{jl}^+) \\
&= \max_{u_l \in t(u_j^+)} c(\pi_{jl}) \\
&\leqslant D(e_i).
\end{aligned}
$$

Similarly, $C_p(\Theta_j^-) \leqslant \max_{u_l \in t(u_j^-)} c(\pi_{jl}) \leqslant D(e_i)$. □

Lemma 9. *Let $G = \langle W, F, R, c\rangle$ be a tree-like decision structure with $|W| = n$ experiments, and let $p = \langle p_1, \ldots, p_n\rangle$ be a vector of success probabilities. Furthermore, let $\varepsilon, \delta > 0$ be any given constants, and let $\hat{p} = (\hat{p}_1, \ldots, \hat{p}_n)$ be a vector of probability estimates computed by the* GS *algorithm of Fig.* 4. *Then*

$$
\forall e_i \in W \ \Pr\left[D(e_i) \times \rho(e_i) \times |p_i - \hat{p}_i| \geqslant \frac{\varepsilon}{2n}\right] \leqslant \frac{\delta}{n}. \tag{A.2}
$$

Proof. We use Hoeffding's inequality, which is a simple form of Chernoff bounds [1,4]: let $\{X_i\}$ be a set of independent, identically distributed random Bernoulli variables, whose common mean is μ. Let $\bar{S}^{(M)} = 1/M \sum_{i=1}^{M} X_i$ be the sample mean after taking M samples. Then

$$
\Pr[\bar{S}^{(M)} > \mu + \lambda] < \mathrm{e}^{-2M\lambda^2}, \quad \Pr[\bar{S}^{(M)} > \mu - \lambda] < \mathrm{e}^{-2M\lambda^2}. \tag{A.3}
$$

To prove that inequality (A.2) holds for each experiment $e_i \in W$, we consider two cases, depending on whether the experiment can be performed initially or not. If so (i.e. if $F(\langle\rangle, e_i)$ and consequently $\rho(e_i) = 1$), then the GS algorithm will perform at least

$$
m(e_i) = \left\lceil 2\left(\frac{nD(e_i)}{\varepsilon}\right)^2 \ln\frac{2n}{\delta}\right\rceil \tag{A.4}
$$

trials of e_i. As the samples are drawn at random from a fixed distribution, we can use inequality (A.3): after $m(e_i)$ samples,

$$
\begin{aligned}
\Pr\left[\rho(e_i) \times |\hat{p}_i - p_i| \geqslant \frac{\varepsilon}{2nD(e_i)}\right] &= \Pr\left[|\hat{p}_i - p_i| \geqslant \frac{\varepsilon}{2nD(e_i)}\right] \\
&\leqslant 2\exp\left(-2m(e_i)\left(\frac{\varepsilon}{2nD(e_i)}\right)^2\right) \\
&\leqslant \frac{\delta}{n}.
\end{aligned}
$$

Now consider an experiment e_i that is not immediately reachable (i.e., such that $F(\langle\rangle, e_i)$ does not hold). Here, the GS algorithm will attempt to reach e_i along a direct path a total of M times, where

$$
M \geqslant \left\lceil 2\left(1 + \frac{\varepsilon}{nD(e_i)}\right)\left(\frac{nD(e_i)}{\varepsilon}\right)^2 \ln\frac{4n}{\delta}\right\rceil. \tag{A.5}
$$

Of these, GS reaches and performs e_i some number k times; and does not reach e_i the remaining $M-k$ times. Denote by $j(k)$ the value of the product $\rho(e_i) \times |\hat{p}_i - p_i|$ assuming that GS succeeds exactly k times. It suffices to show that for any value of k, $j(k) \leqslant \varepsilon/(2nD(e_i))$ with probability at least $1-\delta/n$.

Using the fact that $\hat{\rho}(e_i) = k/M$ is the estimated value of $\rho(e_i)$ here (as GS has succeeded k times out of M) together with inequality (A.3), we know that with probability at least $1-\delta/2n$,

$$\rho(e_i) \leqslant \hat{\rho}(e_i) + \sqrt{\frac{1}{2M}\ln\frac{2n}{\delta}},$$

and with probability at least $1-\delta/2n$,

$$|\hat{p}_i - p_i| \leqslant \sqrt{\frac{1}{2k}\ln\frac{4n}{\delta}}.$$

Of course, each of these terms tops off at 1 (since $\rho(e_i) \leqslant 1$ and $|\hat{p}_i - p_i| \leqslant 1$). We therefore define

$$g(k) = \min\left\{1, \frac{k}{M} + \sqrt{\frac{1}{2M}\ln\frac{2n}{\delta}}\right\} \times \min\left\{1, \sqrt{\frac{1}{2k}\ln\frac{4n}{\delta}}\right\},$$

and observe that $j(k) \leqslant g(k)$ with probability at least $(1-\delta/2n)^2 \geqslant 1-\delta/n$.

We now need to bound the largest possible value of $g(k)$. First, note that $g(k)$ can be bounded by

$$g(k) \leqslant \begin{cases} \dfrac{k}{M} + \sqrt{\dfrac{1}{2M}\ln\dfrac{2n}{\delta}} & \text{when } 0 \leqslant k \leqslant k_0, \\ \left(\dfrac{k}{M} + \sqrt{\dfrac{1}{2M}\ln\dfrac{2n}{\delta}}\right) \times \sqrt{\dfrac{1}{2k}\ln\dfrac{4n}{\delta}} & \text{when } k_0 \leqslant k \leqslant M, \end{cases}$$

where $k_0 = 1/2\ln(4n/\delta)$. (The two expressions for $g(k)$ have the same value at $k = k_0$.) As the first expression has a positive first derivative with respect to k, it is largest at its largest allowed value of k, viz. k_0. Using first and second derivatives, we see that the second expression is upwards concave on the interval $k = [k_0, M]$, and thus its value is maximal at either $k = k_0$ or $k = M$.

Hence, the largest value of $g(k)$ is bounded by the value of the second expression at either $k = k_0$ or $k = M$, i.e. by the larger of the values

$$g(k_0) = \frac{1}{2M}\ln\frac{4n}{\delta} + \sqrt{\frac{1}{2M}\ln\frac{2n}{\delta}},$$

$$g(M) = \frac{1}{2M}\sqrt{\ln\frac{4n}{\delta}\ln\frac{2n}{\delta}} + \sqrt{\frac{1}{2M}\ln\frac{4n}{\delta}}.$$

By inspection, both of these values, and hence all values of $g(k)$, are below

$$g_M = \frac{1}{2M}\ln\frac{4n}{\delta} + \sqrt{\frac{1}{2M}\ln\frac{4n}{\delta}} = \left(\frac{1}{2} + \sqrt{\frac{1}{2M}\ln\frac{4n}{\delta}}\right)^2 - \frac{1}{4}.$$

As $\rho(e_i) \times |\hat{p}_i - p_i| = j(k) \leqslant g_M$, we need only find an M sufficiently large so that $g_M \leqslant \varepsilon/(2nD(e_i))$. Solving $g_M = \varepsilon/(2nD(e_i))$ for M yields

$$M = \left(\sqrt{\frac{2\varepsilon}{nD(e_i)} + 1} - 1\right)^{-2} \ln \frac{4n}{\delta}.$$

To see that the $m(e_i)$ value from Eq. (A.5) is larger than this M (and hence the corresponding g_M value will be yet smaller), just observe that $(\sqrt{2\gamma+1} - 1)^{-2} \leqslant (1+\gamma)/(\gamma^2)$ holds for any $\gamma > 0$, so in particular for $\gamma = \varepsilon/(nD(e_i))$. Hence, after taking at least $m(e_i)$ samples, we can be confident that the product $\rho(e_i) \times |\hat{p}_i - p_i|$ is sufficiently small, as desired.

Notice that in the typical situation where ε is small relative to $D(e)$, the $m(e_i)$ value obtained here is only slightly larger than the value obtained from Eq. (A.4). □

Theorem 10. *Let $G = \langle W, F, R, c\rangle$ be a tree-like decision structure with $|W| = n$ experiments, and let $p = \langle p_1, \ldots, p_n\rangle$ be a vector of success probabilities for the experiments. Furthermore, let $\varepsilon, \delta > 0$ be any given constants, and let $\Theta_{pao} = \mathrm{PAO}(G, \varepsilon, \delta)$ be the strategy produced by the* PAO *algorithm using the GS subroutine of Fig. 4. Then, with probability at least $1 - \delta$, $C_p(\Theta_{pao}) - C_p(\Theta_p) \leqslant \varepsilon$, where $\Theta_p = \mathrm{OSS}(G, p)$ is the optimal strategy for probability vector p.*

Proof. Let $\hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_n\rangle$ be the vector of probability estimates produced by the GS subroutine. By Lemma 9 each of the products $D(e_i) \times \rho(e_i) \times |p_i - \hat{p}_i|$ is upper bounded by the value $\varepsilon/2n$ with probability at least $1 - \delta/n$. Hence, the probability that they are *all* less than $\varepsilon/2n$ is at least $1 - \delta$. The theorem's claim follows from this by Lemma 8. □

Corollary A.1. *Let $\Theta_{pao} = \mathrm{PAO}(G, \varepsilon, \delta)$ be the result of the* PAO *algorithm, where $G = \langle W, F, R, c\rangle$ is any decision structure in $\mathcal{DS}$, and $\varepsilon, \delta > 0$ are given constants. Then, with probability at least $1 - \delta$, $C_p(\Theta_{pao}) - C_p(\Theta_p) \leqslant \varepsilon$ where $\Theta_p = \mathrm{OSS}(G, p)$ is the optimal strategy, based on the correct probability vector p.*

Proof. This result follows immediately from the proof of Theorem 10, using only the easy first part of Lemma 9, and the observation that $C \geqslant D(e_i)$ guarantees that the value M in Eq. (1) is larger than the value $m(e_i)$ in Eq. (A.4). □

Theorem 11. *Assume* RP $\neq$ NP. *Then there is no probabilistic polynomial time algorithm that, given a decision structure $G = \langle W, F, R, c\rangle$, an experiment $e \in W$, a distribution function $p : W \to [0, 1]$, and parameters $\varepsilon, \delta > 0$, can estimate the value $\rho(e)$ to within ε with probability at least $1 - \delta$.*

Proof. Assume to the contrary that such an algorithm exists for some fixed values of $\varepsilon, \delta > 0$; say $\varepsilon = 1/3 = \delta$. We show that this algorithm could also be used to decide the

satisfiability of boolean formulas (SAT) with reliability $1-\delta$.[3] As the SAT problem is NP-complete, it would follow by standard arguments that RP = NP.

Let φ be a boolean formula over the variables $x_1, \ldots, x_n$. We show how to construct a corresponding decision structure $G_\varphi = \langle W_\varphi, F_\varphi, R_\varphi, c_\varphi \rangle$, such that the formula φ has (respectively does not have) a satisfying assignment to its variables if and only if the value $\rho(g)$ for a specific experiment g in W_φ is 1 (respectively 0). We could thus decide the satisfiability of formula φ, with reliability $1-\delta$, by running our hypothetical algorithm on structure G_φ and experiment g, and checking whether the estimate it provides for $\rho(g)$ is greater than $1-\varepsilon$ or less than ε.

The structure G_φ has $2n+1$ experiments $W_\varphi = \{e_1, \bar{e}_1, \ldots, e_n, \bar{e}_n, g\}$. The precedence relation F_φ permits exactly one of e_1 or $\bar{e}_1$ to be performed initially, then exactly one of e_2 or $\bar{e}_2$, and so on; in general permitting exactly one of e_k or $\bar{e}_k$ as the kth experiment. (That is, $F_\varphi(\alpha, e_{k+1})$ and $F_\varphi(\alpha, \bar{e}_{k+1})$ both hold iff α is of the form $\langle\langle \tilde{e}_1+\rangle, \langle \tilde{e}_2+\rangle, \ldots, \langle \tilde{e}_k+\rangle\rangle$, where each $\tilde{e}_i$ is either e_i or $\bar{e}_i$.) Now each complete sequence of the $\tilde{e}$-type experiments, $\alpha = \langle\langle \tilde{e}_1+\rangle, \langle \tilde{e}_2+\rangle, \ldots, \langle \tilde{e}_n+\rangle\rangle$, can be identified with a truth assignment to the variables $x_1, \ldots, x_n$, whereby x_i is true (respectively false) if and only if $\tilde{e}_i = e_i$ (respectively $\tilde{e}_i = \bar{e}_i$) in α. The precedence relation F finally specifies that experiment g can be performed (i.e. $F_\varphi(\alpha, g)$ holds) if and only if this truth assignment satisfies the formula φ.

Consider then the trivial probability distribution that assigns success probability 1 to all experiments, and recall that $\rho(g)$ is the maximum probability of reaching experiment g using any strategy. Given the precedence constraints specified above, it is clear that there exists a strategy Θ for reaching g if and only if there exists a satisfying truth assignment for φ, and any such strategy will have $\rho(g, \Theta) = 1$. Hence $\rho(g) = 1$ if and only if φ is satisfiable, and otherwise $\rho(g) = 0$. □

References

[1] N. Alon, J.H. Spencer and P. Erdös, *The Probabilistic Method* (Wiley, New York, 1992).

[2] J.A. Barnett, How much is control knowledge worth?: a primitive example, *Artif. Intell.* **22** (1984) 77–89.

[3] D.A. Berry and B. Fristedt, *Bandit Problems: Sequential Allocation of Experiments* (Chapman & Hall, London, 1985).

[4] H. Chernoff, A measure of asymptotic efficiency for tests of a hypothesis based on the sums of observations, *Ann. Math. Stat.* **23** (1952) 493–507.

[5] G. DeJong and R. Mooney, Explanation-based learning: an alternative view, *Mach. Learn.* **1** (1986) 145–176.

[6] M.R. Garey, Optimal task sequencing with precedence constraints, *Discrete Math.* **4** (1973).

[7] D. Geiger and J.A. Barnett, Optimal satisficing tree searches, in: *Proceedings AAAI-91*, Anaheim, CA (1991).

[8] R. Greiner, Finding the optimal derivation strategy in a redundant knowledge base, *Artif. Intell.* **50** (1991) 95–116.

[9] R. Greiner and I. Jurišica, A statistical approach to solving the EBL utility problem, in: *Proceedings AAAI-92*, San Jose, CA (1992).

[3] We are indebted to Tom Hancock for suggesting this reduction.

[10] R. Greiner and P. Orponen, Probably approximately optimal derivation strategies, in: J.A. Allen, R. Fikes and E. Sandewall, eds., *Proceedings KR-91*, Cambridge, MA (Morgan Kaufmann, San Mateo, CA, 1991).

[11] R. Greiner and P. Orponen, Probably approximately optimal satisficing strategies, Tech. Rept., Siemens Corporate Research (1993).

[12] D.S. Johnson, A catalog of complexity classes, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science A: Algorithms and Complexity* (Elsevier, Amsterdam, 1990) 67–186.

[13] J.E. Laird, P.S. Rosenbloom and A. Newell, *Universal Subgoaling and Chunking: The Automatic Generation and Learning of Goal Hierarchies* (Kluwer Academic Publishers, Hingham, MA, 1986).

[14] S. Minton, J. Carbonell, C.A. Knoblock, D.R. Kuokka, O. Etzioni and Y. Gil, Explanation-based learning: a problem solving perspective, *Artif. Intell.* **40** (1989) 63–119.

[15] T.M. Mitchell, R.M. Keller and S.T. Kedar-Cabelli, Example-based generalization: a unifying view, *Mach. Learn.* **1** (1986) 47–80.

[16] T.M. Mitchell, S. Mahadevan and L.I. Steinberg, LEAP: a learning apprentice for VLSI design, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985) 573–580.

[17] K.S. Narendra and M.A.L. Thathachar, *Learning Automata: An Introduction* (Prentice-Hall, Englewood Cliffs, NJ, 1989).

[18] K.S. Natarajan, Optimizing depth-first search of AND-OR trees, Research Report RC-11842, IBM T.J. Watson Research Center (1986).

[19] P. Orponen and R. Greiner, On the sample complexity of finding good search strategies, in: *Proceedings COLT-90*, Rochester (1990) 352–58.

[20] S. Sahni, Computationally related problems, *SIAM J. Comput.* **3** (1974) 262–279.

[21] H.A. Simon and J.B. Kadane, Optimal problem-solving search: all-or-none solutions, *Artif. Intell.* **6** (1975) 235–247.

[22] D.E. Smith, Controlling backward inference, *Artif. Intell.* **39** (1989) 145–208.

[23] L.G. Valiant, A theory of the learnable, *Commun. ACM* **27** (1984) 1134–1142.

ELSEVIER

Artificial Intelligence 82 (1996) 45–74

Artificial Intelligence

Knowledge representation and inference in similarity networks and Bayesian multinets

Dan Geiger [a,*], David Heckerman [b]

[a] *Department of Computer Science, Technion Israel Institute of Technology, Haifa 32000, Israel*

[b] *Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399, USA*

Received February 1993

Abstract

We examine two representation schemes for uncertain knowledge: the similarity network (Heckerman, 1991) and the Bayesian multinet. These schemes are extensions of the Bayesian network model in that they represent asymmetric independence assertions. We explicate the notion of relevance upon which similarity networks are based and present an efficient inference algorithm that works under the assumption that every event has a nonzero probability. Another inference algorithm is developed that works under no restriction albeit less efficiently. We show that similarity networks are not *inferentially complete*—namely—not every query can be answered. Nonetheless, we show that a similarity network can always answer any query of the form: "What is the posterior probability of an hypothesis given evidence?" We call this property *diagnostic completeness*. Finally, we describe a generalization of similarity networks that can encode more types of asymmetric conditional independence assertions than can ordinary similarity networks.

1. Introduction

Traditional probabilistic approaches to knowledge acquisition and inference for diagnostic, classification, and pattern-recognition systems face a critical choice: either specify precise relationships between all relevant variables or make uniform independence assumptions throughout the model. The first choice is computationally infeasible except in very small domains, whereas the second choice is rarely justified and often yields inaccurate conclusions. *Bayesian networks* offer a compromise between the two extremes by encoding independence when possible and dependence when necessary.

* Corresponding author. E-mail: dang@cs.technion.ac.il.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0004-3702(95)00014-3

They allow a wide spectrum of independence assertions to be considered by the model builder, so that a practical balance can be established between computational needs and the accuracy of conclusions.

Although Bayesian networks considerably extend traditional approaches, they are not sufficiently expressive to encode every independence assertion that may facilitate knowledge acquisition or speed up inference. One deficiency is their inability to represent naturally *asymmetric independence* assertions. Such assertions state that variables are independent for some but not necessarily for all of their values.

The *similarity network* is a probabilistic representation that addresses this deficiency for problems of diagnosis. The representation employs multiple Bayesian networks, and thereby allows the representation of asymmetric independence assertions. In practice, the representation has proved to be extremely useful, facilitating the construction of expert systems for the diagnosis of breast, lymph-node, intestine, ovary, skin, soft-tissue, testis, and thymus pathology [16, 27], sleep disorders [28], eye diseases [13], and efficiency problems in gas turbines that generate electricity [2].

Heckerman [14] introduces the similarity network representation. In his work, he describes the representation from the perspective of the user, emphasizing the benefits of the representation for knowledge acquisition. The development does not concentrate on issues of probabilistic inference. In particular, Heckerman describes how a similarity network can be converted to a Bayesian network, and proposes that probabilistic inference be performed using the Bayesian network rather than using the similarity network. The disadvantage of this approach is that, in the process of generating a Bayesian network from a similarity network, one encodes asymmetric independence in the numbers rather than in the topology of the Bayesian network. Consequently, these asymmetric assertions are not available to the inference algorithm to speed up computations. In addition, Heckerman's developments are limited to models where (1) there exists an ordering over all variables that is consistent with every Bayesian network within the similarity network, and (2) no relationship among variables is deterministic. We overcome these constrains and also discuss more fully the situation when diagnostic hypotheses are not mutually exclusive or when the hypothesis variable is not a root node.

Moreover, in this paper, we offer several enhancements to the similarity network representation. We present an efficient inference algorithm that works under the assumption that every event has a nonzero probability. Another inference algorithm is developed that works under no restriction albeit less efficiently. In the processes of developing the later algorithm we introduce another representation of asymmetric independence called a *Bayesian multinet*, describe an algorithm that converts a similarity network into a Bayesian multinet, and show how to perform inference using the Bayesian multinet obtained.

We show that similarity networks are not *inferentially complete*—namely—not every query can be answered. Nonetheless, we show that every similarity network can answer any query of the form: "What is the posterior probability of an hypothesis given evidence?" We call this property *diagnostic completeness*. Finally, we describe a generalization of similarity networks that can encode more types of asymmetric conditional independence assertions than can ordinary similarity networks.

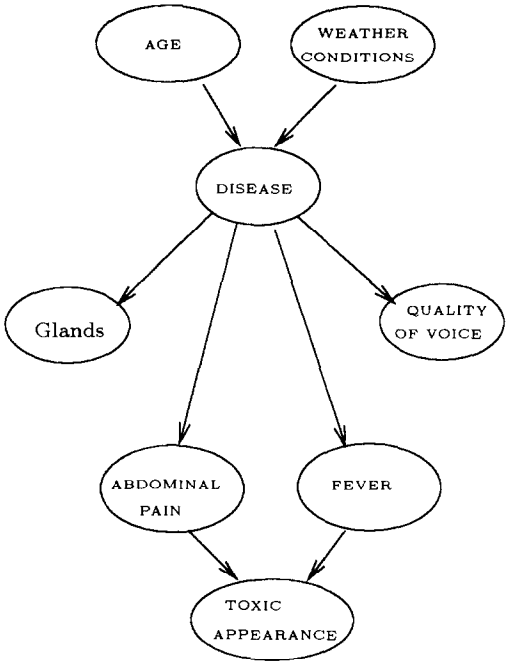


Fig. 1. A Bayesian network for the diagnosis of sore throat.

2. Bayesian networks: an overview [1]

2.1. Informal description

The Bayesian network paradigm was introduced to the AI community by Pearl [31, 32]. It is best explained via a simple example:

> Age and weather influence whether a child gets a sore throat. There are five mutually exclusive and exhaustive types of a sore throat: *viral pharyngitis*, *tonsillar cellulitis*, *mononucleosis*, *strep throat*, and *peritinsillar abscess*. Several symptoms are associated with a sore throat, including fever, toxic appearance, abdominal pain, swollen glands, and voice quality. Most symptoms occur independently of each other in patients having a sore throat, except toxic appearance, which depends upon having fever or abdominal pain. [2]

A Bayesian network representing this description is shown in Fig. 1. The network is constructed from cause-and-effect relationships by placing links from each cause to its direct consequences. For example, *fever* and *pain* are causes for *toxic appearance*, and *disease* is their common cause.

[1] This section is based on Geiger [8].
[2] A modified example of Heckerman [14].

Each node represents a variable having a finite set of values. Continuous variables such as *age* and *fever* are made discrete. For example, the values of *age* can be partitioned into: *infant*, *toddler*, and *school-age child*, and the values of *fever* can be partitioned into: *normal*, *moderately elevated*, and *markedly elevated*. (Work on continuous variables without discretization can be found in [10,32,36].)

Each variable u is associated with a conditional distribution $P(u \mid \pi(u))$, where $\pi(u)$ is the set of parents of u in the network. For example, $P(fever \mid disease)$ is specified by fifteen numbers: one for each value combination of the variables *disease* and *fever*. When u has no parents, that is, when $\pi(u) = \emptyset$, then u is associated with the marginal distribution $P(u)$. For example, $P(age)$ is specified by three numbers depending upon the relative proportion of infants, toddlers, and school-age children among the intended patients. Such distributions are associated with each node in the network.

From the chaining rule of probability, we know that

$$P(u_1, \ldots, u_n) = \prod_i P(u_i \mid u_1, \ldots, u_{i-1}). \tag{1}$$

If P satisfies

$$P(u_i \mid \pi(u_i)) = P(u_i \mid u_1, \ldots, u_{i-1}) \tag{2}$$

for $i = 1, \ldots, n$, then

$$P(u_1, \ldots, u_n) = \prod_i P(u_i \mid \pi(u_i)). \tag{3}$$

Each of the n equalities in Eq. (2) corresponds to an *independence assertion* stating that, given $\pi(u_i)$, u_i is independent of $\{u_1, \ldots, u_{i-1}\} \setminus \pi(u_i)$. (The symbol $\setminus$ stands for set difference.) The structure of a Bayesian network represents these independence assertions, as well as all those assertions implied by them. Thus, according to Eq. (3), a Bayesian network and its associated distributions $P(u_i \mid \pi(u_i))$ determine the joint distribution over $u_1, \ldots, u_n$.

According to Eq. (2), the network of Fig. 1 represents the assertion that *age* and *weather conditions* are independent—that is, $P(age) = P(age \mid weather)$. This assertion appears reasonable. Nonetheless, if this assertion—or any other independence assertion encoded in the network—does not accurately reflect the beliefs of the constructor of the network, then additional nodes or links are drawn until a sufficiently accurate model is realized. For example, one may argue that the life span on the North pole is generally shorter than that in California where weather conditions are more benign. This dependency between age and weather conditions could be modeled by adding a new node called climate and making it the parent of both weather conditions and age.

Another independence assertion implied by the network of Fig. 1 is that toxic appearance is independent of disease, given the values for fever and abdominal pain. That is,

$$\begin{aligned} &P(toxic\ appearance \mid fever,\ pain,\ disease) \\ &\quad = P(toxic\ appearance \mid fever,\ pain). \end{aligned}$$

This assertion reflects the view that fever and pain are the only intervening mechanisms by which a disease related to a sore throat causes toxic appearance. If there are other intervening mechanisms beside pain and fever, then these mechanisms can be represented in the network by either adding new nodes or by adding a direct link between disease and toxic appearance that summarizes the effect of these mechanisms, while keeping them implicit. There are other assertions of independence implied by this graph. All of these assertions can be read directly from the graph (see Section 2.3).

The network and the probability distribution that result from this judgemental process provide a model of a domain as conceived by an expert. Philosophical justifications for the use of probabilities to represent expert's judgments are given in [7,24,34]. In addition, Bayesian networks can be constructed directly from data or from a combination of expert knowledge and data [3,5,6,15,25,33,37].

2.2. Notations and basic definitions

Before we state the definition of Bayesian networks, we provide some notational conventions. Let $\{u_1, \ldots, u_n\}$ be a finite set of variables each having a finite set of values, P be a probability distribution having the Cartesian product of these sets of values as its sample space, and $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ be arbitrary values for $u_1, \ldots, u_n$, respectively. Throughout this article, we often say that P is a probability distribution over U, keeping the remaining details implicit.

We use capital letters from the end of the alphabet (e.g., X, Y, Z) to denote sets of variables and the respective bold characters to denote their values. For example, if $X = \{u_1, u_2\}$, then $\boldsymbol{X} = \{\boldsymbol{u}_1, \boldsymbol{u}_2\}$ is a value of X. We use the notation $P(\boldsymbol{X} \mid \boldsymbol{Y})$ to denote the conditional probability $P(X = \boldsymbol{X} \mid Y = \boldsymbol{Y})$, and $P(X \mid Y) = P(X)$ to denote $P(\boldsymbol{X} \mid \boldsymbol{Y}) = P(\boldsymbol{X})$ for all values of X and Y such that $P(\boldsymbol{Y}) \neq 0$. Also, we use $P(X)$ to denote $P(X \mid \emptyset)$.

Definition. Let P be a probability distribution over U. A directed acyclic graph D (i.e., a directed graph with no directed cycles) is a *Bayesian network of* P, if D is constructed from P by the following steps. Assign a *construction order* $u_1, u_2, \ldots, u_n$ to the variables of U and designate a node u_i for each variable u_i.[3] For each variable u_i in U, identify a set $\pi(u_i) \subseteq \{u_1, \ldots, u_{i-1}\}$ such that

$$P(u_i \mid \pi(u_i)) = P(u_i \mid u_1, \ldots, u_{i-1}) \tag{4}$$

(for all values of $u_1, \ldots, u_i$). Assign a direct link from every node in $\pi(u_i)$ to node u_i. This network is *minimal* if for each $u_i \in U$, no proper subset of $\pi(u_i)$ satisfies Eq. (4).

For example, if $u_1, \ldots, u_5$ is a construction order, and if P satisfies, $P(u_3 \mid u_1) = P(u_3 \mid u_1, u_2)$, $P(u_4 \mid u_2, u_3) = P(u_4 \mid u_1, u_2, u_3)$, and $P(u_5 \mid u_4) = P(u_5 \mid u_1, u_2, u_3, u_4)$, then the network of Fig. 2 is a Bayesian network of P.

[3] We deliberately denote with u_i the node that corresponds to variable u_i. It will be immaterial or clear from the context whether we talk about a variable or its associated node.

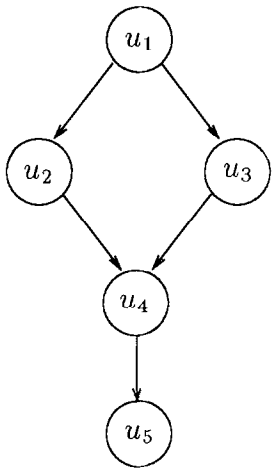


Fig. 2. An abstract example of a Bayesian network.

The number of parameters that a Bayesian network requires and the complexity of its topology depends on the construction order, which is not dictated by its definition. In practice, cause-and-effect and time-order relationships often suggest construction orders that yield simple networks.

Definition. Let P be a probability distribution over U. Let X, Y, and Z be three disjoint subsets of U, and $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ be arbitrary respective values. Then $\boldsymbol{X}$ is *conditionally independent* of $\boldsymbol{Y}$ given $\boldsymbol{Z}$, denoted by $I(\boldsymbol{X},\boldsymbol{Y} \mid \boldsymbol{Z})$, if

$$P(\boldsymbol{X} \mid \boldsymbol{Z},\boldsymbol{Y}) = P(\boldsymbol{X} \mid \boldsymbol{Z}) \quad \text{or} \quad P(\boldsymbol{Z}) = 0.$$

$I(\boldsymbol{X},\boldsymbol{Y} \mid \boldsymbol{Z})$ is called an *independence assertion.*

Definition. A set X is *conditionally independent* of Y given Z, denoted by $I(X,Y \mid Z)$, if $I(\boldsymbol{X},\boldsymbol{Y} \mid \boldsymbol{Z})$ holds for every respective value of X, Y and Z. $I(X,Y \mid Z)$ is called a *symmetric* independence assertion.

Using the above notation, the Bayesian network depicted in Fig. 2 satisfies $I(u_3,u_2 \mid u_1)$, $I(u_4,u_1 \mid \{u_2,u_3\})$, and $I(u_5,\{u_1,u_2,u_3\} \mid u_4)$. (For simplicity, we use u_i to denote the singleton $\{u_i\}$.)

When $I(\boldsymbol{X},\boldsymbol{Y} \mid \boldsymbol{Z})$ holds for some but not all the values of the variables involved, then this independence assertion is called *asymmetric*. This term is adapted from the literature on decision analysis, where asymmetric independence corresponds to asymmetries in decision trees [19]. Asymmetric independence assertions are not represented in the topology of Bayesian networks, whereas the representations described in this paper do explicitly encode such assertions. As we will see, this difference makes our new representations better suited for the representation and solving of diagnostic problems.

2.3. Semantics of Bayesian networks

The criteria of d-separation, defined below, characterizes all independence assertions implied by the topology of a Bayesian network.

Definition. The *underlying graph* of a Bayesian network is an undirected graph obtained from the network by replacing every link with an undirected edge.

Definition. A *trail* in a Bayesian network is a sequence of links that form a cycle-free path in the underlying graph.

Definition (*Pearl* [32]). A node b is called a *head-to-head* node w.r.t. (with respect to) a trail t if there are two consecutive links $a \rightarrow b$ and $b \leftarrow c$ on t.

For example, $u_2 \rightarrow u_4 \leftarrow u_3$ is a trail in Fig. 2 and u_4 is a head-to-head node with respect to this trail.

Definition (*Pearl* [32]). A trail t is *active* w.r.t. a set of nodes Z if (1) every head-to-head node w.r.t. t either is in Z or has a descendant in Z, and (2) every other node along t is outside Z. Otherwise, the trail is said to be *blocked* (or *d-separated*) by Z.

In Fig. 2, for example, both trails between $\{u_2\}$ and $\{u_3\}$ are d-separated by $Z = \{u_1\}$; the trail $u_2 \leftarrow u_1 \rightarrow u_3$ is d-separated by Z because node u_1, which is not a head-to-head node w.r.t. this trail, is in Z whereas the trail $u_2 \rightarrow u_4 \leftarrow u_3$ is d-separated by Z, because node u_4 and its descendant u_5 are outside Z. The trail, $u_2 \rightarrow u_4 \leftarrow u_3$, however, is not d-separated by $Z' = \{u_1, u_5\}$, because u_5 is in Z'.

Theorem 1. *Let D be a Bayesian network of a probability distribution P over U and let X, Y, and Z be three disjoint subsets of U. Then:*

Soundness (Verma and Pearl [38]): *If all trails between a node in X and a node in Y are d-separated by Z, then X and Y are conditionally independent given Z in P.*

Completeness (Geiger and Pearl [11]): *The criterion above lists all independence assertions holding in P that can be identified from the topology of D.*

This theorem is extremely useful. It implies, for example, that each variable is independent of all its non-descendants, conditioned on its parents, because the parents of each node d-separates all trails between a node and its non-descendants.[4] It also implies that each variable is independent, given its parents, children, and its children's parents, of all other variables in the network [31]. Such independence assertions are the cornerstone of efficient computations. A generalization of this theorem is given in [12].

[4] This observation was first made by Howard and Matheson [19], and then proven by Olmsted [29].

2.4. Bayesian networks and inference

Several algorithms exist to compute posterior distributions given that the values of some variables are observed. These algorithms are collectively called *inference algorithms* and they all rely on independence relationships encoded in the network. For example, each of these inference algorithms can compute the posterior distribution for sore-throat diseases given that glands swollen and high fever are observed, based on the prior distribution, and the independence relationships encoded in Fig. 1.

Pearl [30] and Kim and Pearl [23] developed inference algorithms for networks in which every two nodes are connected with at most one trail. Pearl [31] extended the algorithm to general networks.

Another inference algorithm is that of Lauritzen and Spiegelhalter [26], which initially compiles a given network into a *clique tree*. Each node in a clique tree represents a cluster of variables that are collapsed into a single variable whose domain is the Cartesian product of its constituents. The algorithm minimizes as much as possible the size of the largest cluster. Computations of posterior probabilities are done using Kim and Pearl's algorithm on the clique tree. Improvements to this algorithm are described in [21,22].

Shachter [35] developed an inference algorithm based on two types of transformations: *node removal* and *arc reversal*. Node removal is the elimination of a node that has no descendants from the network. This operation corresponds to summing over its possible values. Arc reversal refers to the reversal of a particular arc after the addition of other arcs. The parameters in the transformed network are computed via a simple closed-form formula based on Bayes rule. Both transformations preserve the joint distribution over the remaining variables and can therefore be applied repeatedly for inference [29].

The time complexity of the above three algorithms is exponential, a fact that is not surprising since inference in Bayesian networks is NP-hard [4]. Nevertheless, these algorithms are efficient enough for many real-world applications [1,2,16].

3. Bayesian multinets

3.1. Definition and representational advantages

Although Bayesian networks considerably extend traditional approaches, they are still not expressive enough to encode every piece of information that may reduce computations. The following example demonstrates the inadequacy of Bayesian networks for representing asymmetric independence:

> A guard of a secured building expects three types of persons to approach the building's entrance: workers in the building, approved visitors, and spies. As a person approaches the building, the guard can note its gender and whether or not the person wears a badge. Spies are mostly men. Spies always wear badges in an attempt to fool the guard. Visitors don't wear badges because they don't have one. Female workers tend to wear badges more often than do male workers. The task of the guard is to identify the type of person approaching the building.

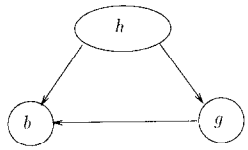


Fig. 3. A Bayesian network for the secured-building example.

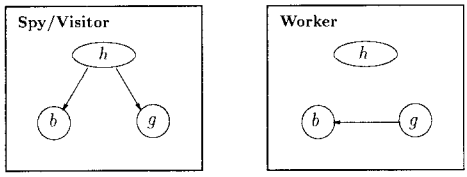


Fig. 4. A Bayesian multinet representation of the secured-building story.

A Bayesian network that represents this story is shown in Fig. 3. Variable h in the figure represents the correct identification. It has three values *worker*, *visitor*, and *spy*. Variables g and b are binary variables representing, respectively, the person's gender and whether or not the person wears a badge. The links from h to g and from h to b reflect the fact that both gender and badge worn are clues for correct identification. The link from g to b encodes the relationship between gender and badge worn.

Unfortunately, the topology of this network hides the fact that gender and badge worn are conditionally independent, given that the person is a spy or a visitor (this assertion holds because, independent of gender, spies always wear badges, and visitors never do). The link between g and b is drawn only because gender and badge worn are related variables when the person is a worker.

We can represent the independence assertions in this story more explicitly using the two Bayesian networks shown in Fig. 4. The first network represents the cases where the person approaching the entrance is either a spy or a visitor. In these two cases, badge worn depends only on the type of person approaching, and not on his or her gender. Consequently, nodes b and g are shown to be conditionally independent (node h blocks the trail between them). The links from h to b and from h to g in this network reflect the fact that badges and gender are relevant clues that help to discriminate spies from visitors. The second network represents the situation where the person is a worker, in which case gender and badge worn are related.

Fig. 4 is a better representation of our story than is Fig. 3, because it shows the dependence of badge worn on gender only in the context in which such a relationship exists—namely, for workers. Moreover, the former representation requires 11 probabilities whereas the representation of Fig. 4 requires only 9 probabilities. This gain, due to the explicit representation of asymmetric independence, can be substantially larger for real-world problems, because the number of probabilities needed can grow exponentially in the number of variables, whereas the overhead of representing multiple networks grows linearly in the number of variables.

We call the representation scheme of Fig. 4 a *Bayesian multinet*. In the remainder of this paper, we often refer to h as the *hypothesis variable*; and we refer to the values of h as *hypotheses*. Furthermore, the variable h will be the focus of construction for Bayesian multinets and for similarity networks, and thus sometimes we call h the distinguished variable. We refer to other variables in a given domain as non-distinguished variables.

Let A_i be a subset of the values of h and let the event $[\![A_i]\!]$ stand for "one of the hypotheses in A_i holds true".

Definition. Let $P(h, u_1, \ldots, u_n)$ be a probability distribution and $A_1, \ldots, A_k$ be non-empty mutually disjoint sets whose union is equal to the set of all values of h (i.e., a partition of the domain of h). A directed acyclic graph D_i is called a *comprehensive local network of P associated with* A_i if D_i is a Bayesian network of $P(h, u_1, \ldots, u_n \mid [\![A_i]\!])$. The set of k local networks is called a *Bayesian multinet* of P. When each A_i is a singleton, we say the Bayesian multinet is *hypothesis-specific*.

In the secured-building example of Fig. 4, $\{\{spy, visitor\}, \{worker\}\}$ is a partition of the values of the hypothesis node h, one local network is a Bayesian network of $P(h, b, g \mid worker)$, and the other local network is a Bayesian network of $P(h, b, g \mid [\![spy, visitor]\!])$ where the conditioning event $[\![spy, visitor]\!]$ is a short-hand notation for saying that either $h = spy$ or $h = visitor$.

The fundamental concept associated with Bayesian multinets is that of *conditioning*; each local network represents a distinct situation where hypotheses are restricted to a specified subset. As a result of such conditioning, asymmetric independence assertions are encoded in the topology of the local networks. Consequently, savings in computations and memory requirements result. In our example, conditional independence between gender and badge worn is encoded as a result of conditioning on h.

Conditioning may also destroy independence relationships rather then create them [32]. Nonetheless, if the distinguished variable is a root node (i.e., a node with no incoming links), then, according to d-separation, conditioning on its values never decreases and often increases the number of independence relationships. We address situations where the hypothesis variable is not a root node and where more than one node represents hypotheses in Sections 3.3 and 5, respectively.

The relationship between gender and badge worn is an example of *hypothesis-specific* independence, wherein two variables are independent given some hypotheses ($\{spies, visitors\}$) but dependent given others ($workers$). In general, a hypothesis-specific independence assertion is represented in a Bayesian multinet whenever a link between two non-distinguished variables exists in some local networks but does not exist in other local networks.

The following variation of the secured-building example demonstrates an additional type of asymmetric independence that can be represented by Bayesian multinets.

> The guard of the secured building now expects *four* types of persons to approach the building's entrance: executives, regular workers, approved visitors, and spies. The guard can note gender, whether or not the person is wearing a badge, and whether or not the person arrives in a limousine (l). We assume that only executives arrive

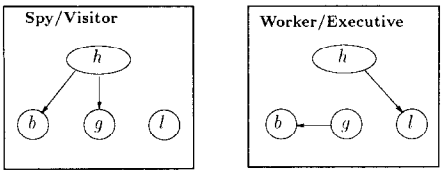


Fig. 5. A Bayesian multinet representation of the augmented secured-building story.

in limousines and that male and female executives wear badges just as do regular workers (to serve as role models).

This story is represented by the two local networks shown in Fig. 5. One network represents a situation where either a spy or a visitor approaches the building, and the other network represents a situation where either a worker or an executive approaches the building. The link from h to l in the latter network reflects the fact that arrival in a limousine is a relevant clue that helps to discriminate workers from executives. The absence of this link in the former network reflects the fact that arrival in a limousine does not help to discriminate spies from visitors.

The relationship between arrival in limousines and the hypothesis variable h is an example of *subset* independence, wherein a non-distinguished variable (l) is independent of h given h draws its values from a subset of hypotheses {*spy*, *visitor*}. In general, a subset independence assertion is represented in a Bayesian multinet whenever a link between the hypothesis node and a non-distinguished variable exists in some local networks but not in all. The relationship between the set of variables {badge worn, gender} and h, when h is restricted to {*worker*, *executive*} is another example of subset independence. [5]

The next theorem demonstrates that a Bayesian multinet always encodes the conditional distribution $P(u_1, \ldots, u_n \mid h)$. Its proof provides us with an inference algorithm.

Theorem 2. *Let M be a Bayesian multinet of $P(h, u_1, \ldots, u_n)$ based on a partition $A_1, \ldots, A_k$ of the values of h. Then, the distribution $P(u_1, \ldots, u_n \mid h)$ can be computed from the parameters encoded in M.*

Proof. The distribution $P(u_1, \ldots, u_n, h \mid [\![A_i]\!])$ is encoded in the local network associated with A_i because according to the definition of a local network,

$$P(u_1, \ldots, u_n, h \mid [\![A_i]\!]) = \prod_v P(v \mid \pi(v), [\![A_i]\!]) \tag{5}$$

where v is either h or some u_i and $\pi(v)$ are v's parents.

Let $\boldsymbol{h}$ be an hypothesis in A_i. The distribution $P(u_1, \ldots, u_n \mid \boldsymbol{h}, [\![A_i]\!])$ is computed from $P(u_1, \ldots, u_n, h \mid [\![A_i]\!])$. Moreover, the distribution $P(u_1, \ldots, u_n \mid \boldsymbol{h}, [\![A_i]\!])$ is equal to $P(u_1, \ldots, u_n \mid \boldsymbol{h})$ because the assertion "h is assigned the value $\boldsymbol{h}$" logically implies

[5] Heckerman [14] coined the terms subset independence and hypothesis-specific independence. A hypothesis-specific Bayesian multinet is similar to hypothesis-specific similarity network defined in [14] except that it contains all the variables in U.

the assertion "h draws its value from A_i" whenever A_i includes $\boldsymbol{h}$. Thus $P(u_1, \ldots, u_n \mid h)$ can be computed from the parameters of M. □

The parameters encoded in a Bayesian multinet can be used to compute the relative posterior probability between every pair of hypotheses within each A_i. In order to compute the absolute value of the posterior probability of each hypothesis, however, one must have information about the prior distribution $P(h)$ in addition to the Bayesian multinet because $P(h)$ cannot be computed from the parameters encoded in the local networks.

3.2. Bayesian multinets and inference

The proof of Theorem 2 and the comment that follows suggest an inference algorithm for computing the posterior distribution of h from a Bayesian multinet of P and from the prior distribution of h. The inference algorithm uses a procedure called *INFER* which has two parameters, one specifying a query of the form "compute $P(X \mid \boldsymbol{Y})$" and the second is a Bayesian network where X and Y are sets of variables that appear in the network and $\boldsymbol{Y}$ is a value of Y. As we have discussed in Section 2.4, there are many ways to realize *INFER* and we do not need to specify *INFER*'s operational details in order to demonstrate how this procedure is extended to operate on Bayesian multinets. The inference algorithm is described below.

Algorithm (*Bayesian multinet inference*).

Input:

- A Bayesian multinet of $P(h, u_1, \ldots, u_n)$ based on a partition $A_1, \ldots, A_k$ of h's values. The local network associated with A_i is denoted by D_i.
- A priori probability distribution $P(h)$.
- Instances $\boldsymbol{u}'_1, \ldots, \boldsymbol{u}'_m$ for a set of variables $\{u'_1, \ldots, u'_m\} \subseteq \{u_1, \ldots, u_n\}$.

Output: The posterior probability distribution $P(h \mid \boldsymbol{u}'_1, \ldots, \boldsymbol{u}'_m)$.

1 For each partition element A_j
2 For each hypothesis $\boldsymbol{h}_i \in A_j$
3 $\alpha_{i,j} = INFER(\, P(\boldsymbol{u}'_1, \ldots, \boldsymbol{u}'_m \mid \boldsymbol{h}_i, [\![A_j]\!]),\ D_j)$
4 For each $\boldsymbol{h}_i$
5 Compute $P(\boldsymbol{h}_i \mid \boldsymbol{u}'_1, \ldots, \boldsymbol{u}'_m) = P(\boldsymbol{h}_i) \cdot \alpha_{i,j} / (\sum_i P(\boldsymbol{h}_i) \cdot \alpha_{i,j})$

Line 3 is the normal computation performed by an inference algorithm for Bayesian networks. Lines 4 and 5 encode Bayes rule together with the fact that the distribution $P(u_1, \ldots, u_n \mid \boldsymbol{h}_i, [\![A_j]\!])$ is equal to $P(u_1, \ldots, u_n \mid \boldsymbol{h}_i)$ which is computed on line 3 and assigned to $\alpha_{i,j}$. This equality follows from the fact that $\boldsymbol{h}_i$ implies $[\![A_j]\!]$ whenever $\boldsymbol{h}_i \in A_j$.

The advantage of computing $P(u_1, \ldots, u_n \mid \boldsymbol{h})$ via this algorithm versus using *INFER* on a Bayesian network of P arises from the fact that independence assertions are represented in some local networks, but not in the Bayesian network. For example, suppose the guard of our secured-building problem sees a person wearing a badge ($\boldsymbol{b}$) approach the building but does not notice the person's gender. Using the Bayesian network of

Fig. 3, *INFER* computes the posterior probability of each possible identification (*worker*, *visitor*, *spy*) as follows:

$$P(h \mid \boldsymbol{b}) = k \cdot P(h) \cdot \sum_{g} P(\boldsymbol{g} \mid h) \cdot P(\boldsymbol{b} \mid \boldsymbol{g}, h) \tag{6}$$

where k is the normalizing constant that makes $P(h \mid \boldsymbol{b})$ sum to unity. Since the Bayesian network representing this problem does not encode any statement of conditional independence the above computation is done by any realization of *INFER*.

Alternatively, our inference algorithm computes the posterior probability of each hypothesis more efficiently, using the Bayesian multinet of Fig. 4, as follows:

$$P(spy \mid \boldsymbol{g}, \boldsymbol{b}) = k \cdot P(spy) \cdot P(\boldsymbol{b} \mid spy), \tag{7}$$

$$P(visitor \mid \boldsymbol{g}, \boldsymbol{b}) = k \cdot P(visitor) \cdot P(\boldsymbol{b} \mid visitor), \tag{8}$$

$$P(worker \mid \boldsymbol{g}, \boldsymbol{b}) = k \cdot P(worker) \cdot \sum_{g} P(\boldsymbol{g} \mid worker) \cdot P(\boldsymbol{b} \mid \boldsymbol{g}, worker). \tag{9}$$

Eqs. (7) and (8) take advantage of hypothesis-specific independence. In particular, the two equations incorporate the fact that g and b are conditionally independent given $h = spy$ and $h = visitor$, respectively. Thus, we do not have to sum over the variable gender as we do when using a Bayesian network (Eq. (6)). These savings are achieved by the inference algorithm for Bayesian multinets because the computations of line 3 are done on the local network that encodes this independence information. If we were to use the same inference algorithm used by line 3 on the Bayesian network of Fig. 3, where this independence assertion is not displayed, then the more costly computation done by Eq. (6) would have been performed.

3.3. Nonroot hypothesis variables

The multinet approach described thus far is especially beneficial when the hypothesis variable can be modeled as a root node because, then, no new links are ever introduced by conditioning on the different hypotheses. Nonetheless, there are situations where modeling the hypothesis node as a root node is awkward. For example, in the secured-building story, suppose there are two independent reports indicating possible spying—say, for military and economical reasons. Such a priori factors for correct identification are best modeled as parent nodes of h, called—say—*economics* and *military*. The resulting subnetwork among these variables is *economics* $\rightarrow h \leftarrow$ *military*, which represents the reasonable assertion that *economics* and *military* are marginally independent.

When h assumes the value *spy*, however, an induced dependency is introduced between its parents *economics* and *military*; For example, a military explanation for a confirmed spy makes less likely an economical explanation, because the former explains the presence of the spy. Consequently, a link must be drawn between the *economics* and *military* nodes in the local network for spies versus visitors. This link would not appear in the full Bayesian network because *economics* and *military* can reasonably be assumed independent. They only become dependent when conditioning on $h = spy$. The

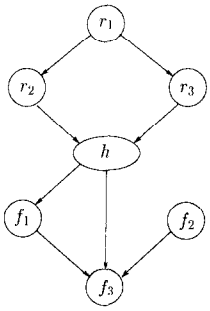


Fig. 6. A Bayesian network where all trails between a priori factors r_i and evidential clues f_i pass through h.

probability distributions associated with such induced links are difficult to assess (e.g., $P(economics \mid h, military)$). Thus, in this example, constructing a local network is harder than constructing the full network.

One approach to handle this problem is as follows. First, construct a Bayesian network that represents only a priori factors that influence the hypotheses, ignoring any evidential variables (such as gender, badge worn, and arrival in limousine). In our example, this network would be $economics \rightarrow h \leftarrow military$. Then, use this network to revise the a priori probabilities of the different hypotheses. Finally, construct local networks ignoring a priori factors (as is done in Fig. 4) and use the resulting multinet with the revised priors of h to compute the posterior probability of h as determined by the evidential clues. This decomposition technique works best if a priori factors are independent of all evidential clues conditioned on the different hypotheses. That is, in situations that can be modeled with Bayesian networks of the form shown in Fig. 6, where all trails between a priori factors r_i and evidential clues f_i pass through h.

When a network of this form cannot serve as a justifiable model, another approach can be used instead. First, compose a Bayesian multinet ignoring a priori factors, construct a Bayesian network from the local networks by taking the union of all their links (e.g., the union of all links in Fig. 4 yields the Bayesian network of Fig. 1). Then, add a priori factors to the resulting network. This approach is described in [14]. The disadvantage of this method is that in the process of generating a Bayesian network from a multinet, one encodes asymmetric independence in the parameters rather than in the topology of the Bayesian network. Consequently, these asymmetric assertions cannot speed up the computations of known inference algorithms. Nevertheless, this approach is still the best alternative for decomposing the construction of large Bayesian networks having topologies more complex than that of Fig. 6.

4. Similarity networks

4.1. Definition and representational advantages

In Bayesian multinets, we required that every variable be included in each local network. This requirement stands in contrast to the observation that in many domains

each measurement often helps to discriminate only a specialized class of hypotheses. Symptoms are often related to narrow classes of diseases, and systems' faults often isolate a specific class of potential malfunctions. Assessing the dependence between two variables under assumptions unrelated to their semantics can present an insurmountable burden on the model builder. This difficulty was realized during the construction of an expert system for surgical pathology diagnosis [14]. When the expert pathologist was asked by the model builder: Given a particular disease, does observing symptom x change your belief that you will observe symptom y? The pathologist would sometimes reply:

> I've never thought about these two symptoms at the same time before. Symptom x is relevant to only one set of diseases, while symptom y is only relevant to another set of diseases. These sets of diseases do not overlap, and I never confuse the first set of diseases with the second.

An erroneous solution to this difficulty is to include in each local network of a Bayesian multinet only those variables that help to discriminate among the hypotheses covered by that local network. In doing so, however, valuable information for correct identification may be lost.

For example in the secured-building problem gender (g) and badge worn (b) do not help to discriminate workers from executives. If these variables would not have been depicted in the local network for $\{worker, executive\}$ in the Bayesian multinet of Fig. 5 then this multinet would have failed to represent the genuine relationship between badge worn and gender.

As we will see, a correct solution to this difficulty is indeed to include in each local network only those variables that help to discriminate among the hypotheses covered by that local network, but also to construct additional local networks to compensate for lost information. The structure that results is called a *similarity network* [14]. For example, the secured-building problem can be represented by a similarity network shown in Fig. 7. Whereas the Bayesian multinet of Fig. 5 contains two local networks, the similarity network contains three local networks: one local network helps to discriminate spies from visitors, another local network helps to discriminate visitors from workers, and a third local network helps to discriminate workers from executives. In each local network, we include only those variables that help to discriminate among the hypotheses covered by that local network. For example, in Fig. 7, the dependence between badge worn and gender is not included in the local network for workers versus executives. This dependence, however, is included in the local networks for visitors versus workers, because badge worn helps to discriminate between these two hypotheses.

The main advantage of similarity networks, from the perspective of knowledge acquisition, is that a domain expert who provides the parameters of the network is not required to quantify the dependence between variables that are not related to the hypotheses under consideration. In order not to loose information needed for correct diagnosis we will see that the local networks must be based on a *connected cover* of hypotheses.

Definition. A *cover* of a set of hypotheses H is a collection $\{A_1, \ldots, A_k\}$ of nonempty subsets of H whose union is H. Each cover is a hypergraph, called a *similarity hyper-*

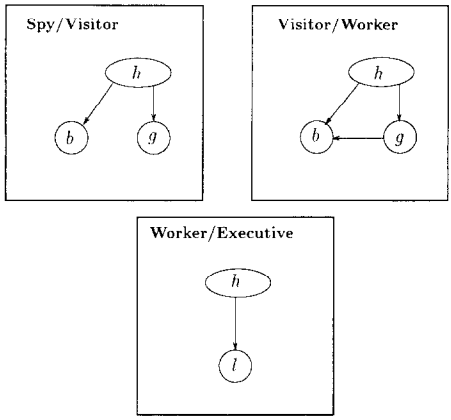


Fig. 7. A similarity network representation of the secured-building story.

graph, where the A_i are hyperedges and the hypotheses are nodes. A cover is *connected* if the similarity hypergraph is connected.

In Fig. 7, {*spy*, *visitor*}, {*visitor*, *worker*}, {*worker*, *executive*} is a cover of the hypotheses set. This cover is connected because it consists of the three links *spy–visitor–worker–executive* which form a connected hypergraph (as well as a connected graph). The set {*spy*, *visitor*}, {*worker*, *executive*} is also a cover but it is not connected. The set {*worker*, *executive*, *visitor*}, {*visitor*, *spy*} is an example of a connected cover that is a hypergraph but not a graph.

Definition. Let $P(h, u_1, \ldots, u_n)$ be a probability distribution and $A_1, \ldots, A_k$ be a connected cover of the values of h. A directed acyclic graph D_i is called a *local network of P associated with A_i* if D_i is a Bayesian network of $P(h, v_1, \ldots, v_m \mid [\![A_i]\!])$ where $\{v_1, \ldots, v_m\}$ is the set of all variables in $\{u_1, \ldots, u_n\}$ that "help to discriminate" the hypotheses in A_i. The set of k local networks is called a *similarity network* of P.

We define "help to discriminate" formally in the next section.

The definition of similarity networks does not specify how to select a connected cover of hypotheses. Although any selection of a connected cover yields a valid similarity network, some selections yield similarity networks that display more asymmetric independence assertions than do other selections. An analogous situation exists when constructing a Bayesian network where some construction orders yield Bayesian networks that display more symmetric independence assertions than do other Bayesian networks. The practical solution for constructing a Bayesian network is to choose a construction order according to cause-effect relationships.[6] This selection tends to maximize the information about symmetric independence encoded in the resulting network.

[6] Bayesian networks are often called *causal networks*.

The practical solution for constructing the similarity hypergraph is to choose a connected cover by grouping together hypotheses that are "similar" to each other by some accessible criteria (e.g., spies and visitors are outsiders). This choice tends to maximize the number of asymmetric independence assertions encoded in a similarity network. Hence the name for this representation.

Similarity networks have another important advantage not mentioned so far: This representation helps to prevent the model builder from omitting relevant information. For example, suppose workers and executives often arrive at work with a smile, whereas spies and visitors often arrive at work without a smile. This information is likely to be forgotten when constructing the local networks for spies versus visitors and for visitors versus executives because it does not help to discriminate between these pairs of hypotheses. When constructing the similarity network of Fig. 7, however, the builder of the network is likely to recall the information about smile because he must compose a local network for discriminating visitors from workers—a task for which this information is important. In general, whenever the cover is connected, every variable that is useful for discriminating some pair of hypotheses will appear in at least one local network. This property of similarity network was called *exhaustiveness* by Heckerman [14].

4.2. Relevance relations

The definition of similarity networks is not complete without attributing a precise meaning to the utterance "helps to discriminate" used in the definition of a local network. Below we give several possibilities.

Definition. Let $P(u_1, \ldots, u_n \mid \boldsymbol{e})$ be a probability distribution where $\boldsymbol{e}$ is a fixed event. Variables u_i and u_j are *unrelated given* $\boldsymbol{e}$ if u_i and u_j are disconnected in every minimal Bayesian network of $P(u_1, \ldots, u_n \mid \boldsymbol{e})$. Otherwise, u_i and u_j are *related given* $\boldsymbol{e}$, denoted $related(u_i, u_j \mid \boldsymbol{e})$.

This definition states that two variables u_i an u_j are unrelated given $\boldsymbol{e}$ if there exists no trail connecting them, i.e., there exists no sequence of variables $u_i, \ldots, u_j$ such that every two consecutive variables in this sequence are connected with a link. The requirement that u_i and u_j be disconnected in *every* minimal network is not as strong as it may seem because if u_i and u_j are disconnected in one minimal Bayesian network of P then u_i and u_j are disconnected in every minimal Bayesian network of P [9]. Furthermore, one can phrase the definition of relatedness as follows.

Theorem 3 (Geiger and Heckerman [9]). *Let $P(u_1, \ldots, u_n \mid \boldsymbol{e})$ be a probability distribution over $U = \{u_1, \ldots, u_n\}$ and $\boldsymbol{e}$ be a fixed event. Then, u_i and u_j are unrelated given $\boldsymbol{e}$ iff there exist a partition U_1, U_2 of U such that $u_i \in U_1$, $u_j \in U_2$, and $P(U_1, U_2 \mid \mathrm{e}) = P(U_1 \mid \boldsymbol{e}) P(U_2 \mid \boldsymbol{e})$.*[7]

[7] In [9], $P(U \mid \boldsymbol{e})$ is replaced with $P(U)$. Since $\boldsymbol{e}$ is a fixed event, this shift of notation does not alter this theorem's proof.

An immediate consequence of this theorem is that the relation *related* is transitive, namely, for every three variables u_i, u_j and u_k,

$$(related(u_i, u_j \mid \boldsymbol{e}) \text{ and } related(u_j, u_k \mid \boldsymbol{e})) \Rightarrow related(u_i, u_k \mid \boldsymbol{e}). \tag{10}$$

The second definition is the more appealing one from a knowledge engineering point of view. It states that two variables u_i and u_j are unrelated if, in any context, knowing the value of one variable does not change the knowledge about the values of the other.

Definition. Let $P(u_1, \ldots, u_n \mid \boldsymbol{e})$ be a probability distribution where $\boldsymbol{e}$ is a fixed event. Variables u_i and u_j are *pairwise irrelevant* given $\boldsymbol{e}$ if

$$P(u_i \mid u_j, v_1 \in V_1, \ldots, v_m \in V_m, \boldsymbol{e}) = P(u_i \mid v_1 \in V_1, \ldots, v_m \in V_m, \boldsymbol{e})$$

or

$$P(v_1 \in V_1, \ldots, v_m \in V_m \mid \boldsymbol{e}) = 0$$

for every subset of values $V_1, \ldots, V_m$ for $v_1, \ldots, v_m$, respectively, where $\{v_1, \ldots, v_m\}$ is an arbitrary subset of $\{u_1, \ldots, u_n\} \setminus \{u_i, u_j\}$.

This definition states that u_i and u_j are pairwise irrelevant if they are independent given any possible context. That is, if u_i and u_j are independent given that the value of each v_i $(i = 1, \ldots, m)$ is taken from a subset V_i of v_i's domain and that u_i and u_j remain independent under each selection of variables $v_1, \ldots, v_m$ and for each restriction of their values.

When $u_1, \ldots, u_n$ are all binary variables, then the sets V_i are singletons, namely, a single specific assignment for v_i. In [9], we have shown that pairwise irrelevance and unrelatedness are equivalent when all variables are binary and when the distribution P is strictly positive. We conjecture that the equivalence between pairwise irrelevance and relatedness holds even when these restrictions are lifted.

In the remainder of this paper we use the concept of relatedness for defining which variables are excluded from a local network. We do so by distinguishing one variable as the hypothesis variable (call it h) and defining the event $\boldsymbol{e}$ to be $[\![A_i]\!]$, namely, a disjunction over a subset of the values of h. A variable x is then excluded from a local network for A_i iff x and h are unrelated given $[\![A_i]\!]$.

4.3. Inference using similarity networks

Similarity networks are designed for the task of diagnosis or discrimination. In particular, they are designed to compute the posterior probability of each possible hypothesis given a set of observations. In this section, we show that under reasonable assumptions, the computation of the posterior probability of each hypothesis can be done in each local network and then be combined coherently according to the axioms of probability theory. We analyze the complexity of our algorithm demonstrating its superiority over inference algorithms that operate on Bayesian networks.

We assume that any instantiation of the variables in a similarity network of P has a nonzero probability to occur. Such a probability distribution is said to be *strictly positive*.

This assumption is reasonable for some domains of medical diagnosis, where given an arbitrary collection of clinical findings, the existence of each disease retains a nonzero probability. Subject to this assumption, we develop an inference algorithm that operates directly on similarity networks. We will remove this assumption later at the cost of higher complexity.

The inference problem at hand can be stated as follows: Given a similarity network of $P(h, u_1, \ldots, u_n)$ that is based on a partition $\mathcal{A} = \{A_1, \ldots, A_k\}$ of the values of h, and given a set of assignments $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ for a set $v_1, \ldots, v_m$ of variables that is a subset of $\{u_1, \ldots, u_n\}$ compute $P(\boldsymbol{h}_j \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$—the posterior probability of $\boldsymbol{h}_j$—for every $\boldsymbol{h}_j$.

In order to compute the posterior probability of each $\boldsymbol{h}_j$ we use the procedure *INFER*. As in Section 3.2, this procedure has two parameters, one specifying a query of the form "compute $P(X \mid \boldsymbol{Y})$" and the second is a Bayesian network where X and Y are sets of variables that appear in the network and $\boldsymbol{Y}$ is a value of Y. As before we do not need to specify *INFER*'s operational details in order to demonstrate how this procedure is extended to operate on similarity networks. The new inference algorithm is described below.

First, for each $\boldsymbol{h}_i$ we identify a set of hypotheses $A_j \in \mathcal{A}$ to which $\boldsymbol{h}_i$ belongs and compute the posterior probability of hypothesis $\boldsymbol{h}_i$ under the additional assumption that one of the hypotheses in A_j holds true. In other words, we compute $P(\boldsymbol{h}_i \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m, [\![A_j]\!])$. Second, we compute the posterior probabilities $P(\boldsymbol{h}_j \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ from the probabilities $P(\boldsymbol{h}_j \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m, [\![A_i]\!])$, by solving a set of linear equations:

$$P(\boldsymbol{h}_j \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m) = P(\boldsymbol{h}_j \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m, [\![A_i]\!]) \cdot \sum_{\boldsymbol{h}_j \in A_i} P(\boldsymbol{h}_j \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$$

that relate these quantities. We will see later that these equations have a unique solution.

It remains to show how to compute the query $P(\boldsymbol{h}_i \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m, [\![A_j]\!])$. It seems that one can merely call the procedure *INFER* to compute this query using the local network D_j which corresponds to A_j. The query $P(\boldsymbol{h}_i \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m, [\![A_j]\!])$, however, may include variables that do not appear in D_j in which case *INFER* is not applicable.

Fortunately, the following equality will be shown to hold:

$$P(\boldsymbol{h}_i \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_l, [\![A_j]\!]) = P(\boldsymbol{h}_i \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m, [\![A_j]\!]) \tag{11}$$

where $v_1, \ldots, v_l$ are the variables in $\{v_1, \ldots, v_m\}$ that appear in D_j and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_l$ are their values. Thus to compute $P(\boldsymbol{h}_i \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m, [\![A_j]\!])$ we use the procedure *INFER* to compute the query $P(\boldsymbol{h}_i \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_l, [\![A_j]\!])$ using the network D_j. Eq. (11) tells us that the two computations yield identical answers.

Eq. (11) states that $v_{l+1}, \ldots, v_m$ are conditionally independent of $\boldsymbol{h}_j$ given every value of the variables $v_1, \ldots, v_l$ that appear in D_j where $v_{l+1}, \ldots, v_m$ are the variables in $\{v_1, \ldots, v_m\}$ that do not appear in D_j. If Eq. (11) does not hold, some of the variables in $\{v_{l+1}, \ldots, v_m\}$ would appear in the local network D_j, contrary to our assumption that D_j contains only $v_1, \ldots, v_l$.

This algorithm is summarized below.

Algorithm (*Inference in similarity networks*).

Input: A similarity network of $P(u_1, \ldots, u_n, h)$ based on a connected cover $A_1, \ldots, A_k$ of h's values.

Output: $P(h \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ where $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ are values of variables $v_1, \ldots, v_m$ and $\{v_1, \ldots, v_m\}$ is a subset of $\{u_1, \ldots, u_n\}$.

Notation: D_j denotes the local network that corresponds to A_j and V_j are the variables that appear in D_j.

1 For each A_j
2 Let $\{v_1, \ldots, v_l\}$ be the variables in $V_j \cap \{v_1, \ldots, v_m\}$
3 For each $\boldsymbol{h}_i \in A_j$
4 $\alpha_{i,j} := INFER(P(\boldsymbol{h}_i \mid \boldsymbol{v}_1, \ldots \boldsymbol{v}_l, [\![A_j]\!]), D_j)$
5 If $\alpha_{i,j} = 0$, then Return "P is not strictly positive"
6 Solve the following set of linear equations:
7 For all i and j, $P(\boldsymbol{h}_i \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m) = \alpha_{i,j} \cdot \sum_{\boldsymbol{h}_i \in A_j} P(\boldsymbol{h}_i \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$
8 $\sum_i P(\boldsymbol{h}_i \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m) = 1$
9 Return $P(h \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$

We have argued already that the solution to the equations listed in lines 7 and 8 provide the desired posterior probability. It remains to show that there exists a unique solution. Let us examine a local network D_j that corresponds to A_j. Assume A_j consists of $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_r$. Since $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ remain fixed throughout the computations we denote $P(\boldsymbol{h}_i \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ by $Q(\boldsymbol{h}_i)$. Consider the following equations:

$$Q(\boldsymbol{h}_1) = \alpha_{1,j}\,[Q(\boldsymbol{h}_1) + Q(\boldsymbol{h}_2) + \cdots + Q(\boldsymbol{h}_r)]\,, \tag{12}$$

$$Q(\boldsymbol{h}_2) = \alpha_{2,j}\,[Q(\boldsymbol{h}_1) + Q(\boldsymbol{h}_2) + \cdots + Q(\boldsymbol{h}_r)]\,, \tag{13}$$

$$\vdots$$

$$Q(\boldsymbol{h}_r) = \alpha_{r,j}\,[Q(\boldsymbol{h}_1) + Q(\boldsymbol{h}_2) + \cdots + Q(\boldsymbol{h}_r)]\,. \tag{14}$$

These are the subset of the equations defined in line 7 which correspond to the local network D_j. By dividing every pair of consecutive equations, we obtain the following ratios:

$$Q(\boldsymbol{h}_r) = \frac{\alpha_{r,j}}{\alpha_{r-1,j}} Q(\boldsymbol{h}_{r-1}), \qquad Q(\boldsymbol{h}_{r-1}) = \frac{\alpha_{r-1,j}}{\alpha_{r-2,j}} Q(\boldsymbol{h}_{r-2}),$$
$$\ldots, Q(\boldsymbol{h}_2) = \frac{\alpha_{2,j}}{\alpha_{1,j}} Q(\boldsymbol{h}_1). \tag{15}$$

Hence, the solution of these equations provides the ratios of the posterior probabilities between every pair of hypotheses in A_j. Since we repeat this process for every A_j and since the cover defined by $A_1, \ldots A_k$ is connected, the ratio of every pair of hypotheses is established. To obtain the absolute values of each $Q(\boldsymbol{h}_i)$, it remains to normalize their sum to one, using the equation on line 8 of the algorithm.

Consequently we have proven the following theorem.

Theorem 4. *Let $P(h, u_1, \ldots, u_n)$ be a strictly positive probability distribution and $\mathcal{A} = \{A_1, \ldots, A_k\}$ be a partition of the values of h. Let S be a similarity network based on $\mathcal{A}$. Let $v_1, \ldots, v_m$ be a subset of variables whose value is given. There exists a single solution for the set of equations defined by lines 7 and 8 of the above algorithm and this solution determines uniquely the conditional probability $P(h \mid v_1, \ldots, v_m)$.*

An important observation to make is that the equations on lines 7 and 8 are derived from a given probability distribution $P(h, u_1, \ldots, u_n)$. Consequently, although some equations might be redundant, these equations are always consistent. When the set of local networks is constructed from expert judgments, as done in practice, consistency is not guaranteed. Heckerman [14] describes an algorithm that helps a user to construct a consistent set of local networks by prompting to his attention all probabilities that have already been assigned previously in another local network and verifying with him that these probabilities are acceptable.

It remains to analyze the complexity of this inference algorithm. For simplicity, we assume that all variables are binary in which case the procedure *INFER* has a worst-case complexity of $O(2^n)$. In the worst case, the proposed inference algorithm may not perform more efficiently, because all n variables may appear in each local network. In practice, however, each local network contains a small percentage, say c, of the n variables because all other variables are irrelevant given the context of a specific local network.[8] If $O(n)$ local networks are given, the worst-case complexity of applying *INFER* to these local networks is $O(n \cdot 2^{cn})$, which is smaller than $O(2^n)$ obtained by applying *INFER* on a single Bayesian network generated from these local networks. The complexity of solving the equations on lines 7 and 8 is ignored because it is linear in n. Thus from a worst-case of 2^{100} calculations, for example, we reduce the number of calculations to $100 \cdot 2^{20}$.

4.4. Inferential and diagnostic completeness

An important property of Bayesian networks is that their parameters encode the entire joint distribution through the product rule (Eq. (3)). This property guarantees that any inference task can in principle be computed from the parameters encoded in a Bayesian network. Motivated by this observation we establish the following definition.

Definition. A similarity network S for $P(u_1, \ldots, u_n, h)$ is *inferentially complete* if the distribution $P(u_1, \ldots, u_n, h)$ can be recovered from the parameters of S.

Not all similarity networks are inferentially complete. For example, if $P(u_1, \ldots, u_n, h)$ factors into the product $P(u_1)P(u_2 \ldots, u_n, h)$ then the variable u_1 will not be included in any local network. Therefore, it will be impossible to recover $P(u_1)$ from the parameters encoded in the similarity networks of P. The information about $P(u_1)$ that is lost in the process of producing a similarity network of P, however, is never needed in order to compute the posterior probability of any hypothesis. Evidently, inferential

[8] An approximate number for c in the lymph-node pathology domain is 0.2.

completeness is too strong a requirement for the purpose of computing the posterior probability of each hypothesis.

Definition. A similarity network S for $P(h, u_1, \ldots, u_n)$ is *diagnostically complete* if the conditional distribution $P(h \mid v_1, \ldots, v_m)$ can be recovered from the parameters of S for every subset $\{v_1, \ldots, v_m\}$ of $\{u_1, \ldots, u_n\}$.

In the previous section, we showed that every similarity network of a strictly positive probability distribution P is diagnostically complete (Theorem 4). The inference algorithm we presented shows how to compute $P(h \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ for every value of $v_1, \ldots, v_m$. If P is not strictly positive, then one can construct examples where the equations defined by lines 7 and 8 of our inference algorithm do not determine the probability $P(h \mid \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$. Nevertheless, we will prove that, under minor restrictions, every similarity network is diagnostically complete.

Before proving diagnostic completeness we resort to an example where our inference algorithm fails, and examine how the posterior probability can be computed in an alternative way. This computation highlights the general approach. Suppose S is a similarity network for $P(h, y)$ where h has three values $\{\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3\}$ having equal a priori probability and suppose that y has two values $+y$, $-y$. Also assume that S is based on the cover $\{\{\boldsymbol{h}_1, \boldsymbol{h}_2\}, \{\boldsymbol{h}_2, \boldsymbol{h}_3\}\}$ and that $P(+y \mid \boldsymbol{h}_2) = 0$.

When we apply our algorithm to compute $P(\boldsymbol{h}_i \mid +y)$, the algorithm generates three equations $P(\boldsymbol{h}_1 \mid +y, [\![\boldsymbol{h}_1, \boldsymbol{h}_2]\!]) = 1$, $P(\boldsymbol{h}_2 \mid +y, [\![\boldsymbol{h}_2, \boldsymbol{h}_3]\!]) = 0$, and $P(\boldsymbol{h}_3 \mid +y, [\![\boldsymbol{h}_2, \boldsymbol{h}_3]\!]) = 1$. From these three equations, we cannot compute the relative magnitude of the posterior probability of $\boldsymbol{h}_1$ versus $\boldsymbol{h}_3$. All three equations merely show that $P(\boldsymbol{h}_2 \mid +y)$ is zero.

Nonetheless, $P(\boldsymbol{h}_i \mid +y)$ can be computed from the parameters that quantify S. These parameters include the following: $P(\boldsymbol{h}_1 \mid \boldsymbol{h}_1 \vee \boldsymbol{h}_2)$, $P(\boldsymbol{h}_2 \mid \boldsymbol{h}_2 \vee \boldsymbol{h}_3)$, $P(\boldsymbol{h}_3 \mid \boldsymbol{h}_2 \vee \boldsymbol{h}_3)$, and $P(+y \mid \boldsymbol{h}_1, \boldsymbol{h}_1 \vee \boldsymbol{h}_2)$, $P(+y \mid \boldsymbol{h}_2, \boldsymbol{h}_1 \vee \boldsymbol{h}_2)$ and $P(+y \mid \boldsymbol{h}_3, \boldsymbol{h}_2 \vee \boldsymbol{h}_3)$. From the first three parameters, $P(\boldsymbol{h}_i)$, $i = 1, \ldots, 3$, can be recovered provided none of the prior probabilities is zero. The restriction that all prior probabilities are nonzero is quite reasonable. If the prior probability of some hypothesis were zero, there would be little reason to include that hypothesis in the model.

The other three parameters are equal to $P(+y \mid \boldsymbol{h}_1)$, $P(+y \mid \boldsymbol{h}_2)$, and $P(+y \mid \boldsymbol{h}_3)$, respectively, because $\boldsymbol{h}_i$ entails $\boldsymbol{h}_i \vee \boldsymbol{h}_j$. Thus $P(\boldsymbol{h}_i \mid +y)$ can be computed by Bayes rule:

$$P(\boldsymbol{h}_i \mid +y) = \frac{P(+y \mid \boldsymbol{h}_i) P(\boldsymbol{h}_i)}{\sum_{j=1}^{3} P(+y \mid \boldsymbol{h}_j) P(\boldsymbol{h}_j)}.$$

This example suggests a general methodology for computing the posterior probability of each hypothesis. The general method is based on the proof of the following two theorems.

Theorem 5 (restricted inferential completeness). *Let S be a similarity network of $P(h, u_1, \ldots, u_n)$ based on the connected cover $A_1, \ldots, A_k$ of the values of h. Let $\{v_1, \ldots, v_l\}$ be a subset of variables in $\{u_1, \ldots, u_n\}$ that satisfy related(v_i, h). Then,*

the distribution $P(h, v_1, \ldots, v_l)$ can be computed from the parameters encoded in S provided $P(\boldsymbol{h}_i) \neq 0$ for every value $\boldsymbol{h}_i$ of h.

Proof. To show that the distribution $P(h, v_1, \ldots, v_l)$ can be computed from the parameters of S, we will show how to compute $P(h)$ and then we will show how to compute $P(v_1, \ldots, v_l \mid h)$. The product of these two probability distributions is equal to $P(h, v_1, \ldots, v_l)$.

For each hypothesis $\boldsymbol{h}_i$, let $\alpha_{i,j}$ equal $INFER(P(\boldsymbol{h}_i \mid [\![A_j]\!]), D_j)$, where A_j contains $\boldsymbol{h}_i$ and D_j is the local network corresponding to A_j. The prior probability of each $\boldsymbol{h}_i$ is computed by solving the following set of linear equations:

$$P(\boldsymbol{h}_i) = \alpha_{i,j} \cdot \sum_{\boldsymbol{h}_i \in A_j} P(\boldsymbol{h}_i), \quad \sum_{1}^{n} P(\boldsymbol{h}_i) = 1.$$

In the previous section, we solved these equations and showed that the solution (Eq. (15)) is unique provided $P(\boldsymbol{h}_i) \neq 0$ for all $\boldsymbol{h}_i$.

Due to the chaining rule, $P(v_1, \ldots, v_l \mid \boldsymbol{h}_i)$ can be factored as follows:

$$P(v_1, \ldots, v_l \mid \boldsymbol{h}_i) = P(v_1 \mid \boldsymbol{h}_i) \cdot P(v_2 \mid v_1, \boldsymbol{h}_i) \cdots P(v_l \mid v_1, \ldots, v_{l-1}, \boldsymbol{h}_i). \tag{16}$$

Thus, it suffices to show that for each variable v_j, $P(v_j \mid v_1, \ldots, v_{j-1}, \boldsymbol{h}_i)$ can be computed from the parameters encoded in S. Furthermore we can assume that the conditioning event is possible, namely, $P(v_1, \ldots, v_{l-1}, \boldsymbol{h}_i) > 0$, lest the entire product is zero and the equality holds.

Let D_i denote a local network in S, A_i be the hypotheses associated with D_i, and $\boldsymbol{h}_i$ be an hypothesis in A_i. Each variable v_j is depicted in some local network because it satisfies $related(v_j, h)$. Let $A_i, A_{i+1}, \ldots, A_m$ be a path in the similarity hypergraph where A_m is the only hyperedge on this path associated with a local network that depicts v_j as a node. Such a path exists because the similarity hypergraph is connected and v_j is depicted in one of the local networks. If v_j is depicted in A_i (i.e., $m = i$) then $P(v_j \mid v_1, \ldots, v_{j-1}, \boldsymbol{h}_i)$ can be computed from the local network that corresponds to A_i.

Suppose that $m > i$. Let D_k be the local network associated with A_k for $k = i+1, \ldots, m$ and let $\boldsymbol{h}_{i+1}, \boldsymbol{h}_{i+2}, \ldots, \boldsymbol{h}_m$ be a sequence of hypotheses such that $\boldsymbol{h}_k \in A_{k-1} \cap A_k$. Due to the definition of a similarity network, since v_j is not depicted in D_k where $k < m$, v_j is unrelated to h given $[\![A_k]\!]$. Thus,

$$P(v_j \mid v_1, \ldots, v_{j-1}, \boldsymbol{h}_{k-1}, [\![A_k]\!]) = P(v_j \mid v_1, \ldots, v_{j-1}, \boldsymbol{h}_k, [\![A_k]\!]).$$

Since h implies $[\![A]\!]$ whenever $h \in A$ it follows that

$$P(v_j \mid v_1, \ldots, v_{j-1}, \boldsymbol{h}_{k-1}) = P(v_j \mid v_1, \ldots, v_{j-1}, \boldsymbol{h}_k).$$

This equation holds for every k between $i+1$ and m, thus we obtain,

$$P(v_j \mid v_1, \ldots v_{j-1}, \boldsymbol{h}_i) = P(v_j \mid v_1, \ldots v_{j-1}, \boldsymbol{h}_m).$$

Furthermore,

$$P(v_j \mid v_1, \ldots v_{j-1}, \boldsymbol{h}_m) = P(v_j \mid v'_1, \ldots v'_l, \boldsymbol{h}_m) \tag{17}$$

where $v'_1, \ldots, v'_l$ are the subset of variables of $v_1, \ldots, v_{j-1}$ which are depicted in D_m.

Eq. (17) holds lest $related(v, v_j \mid [\![A_m]\!])$ would hold, where v is some variable in $\{v_1, \ldots, v_{j-1}\}$ not appearing in D_m. However, together with $related(v_j, h \mid [\![A_m]\!])$ which holds because v_j is depicted in D_m, these two assertions would imply by transitivity that $related(v, h \mid [\![A_m]\!])$ holds too, contradicting the fact that v is assumed not to be included in D_m. Finally,

$$P(v_j \mid v'_1, \ldots v'_l, \boldsymbol{h}_m) = P(v_j \mid v'_1, \ldots v'_l, \boldsymbol{h}_m, [\![A_m]\!]), \tag{18}$$

because $\boldsymbol{h}_m$ logically implies the disjunction over all hypotheses in A_m.

The latter probability can be computed using *INFER* on the local network D_m. Thus, due to the equalities above, $P(v_j \mid v_1, \ldots, v_{j-1}, \boldsymbol{h}_i)$ can be computed as needed. □

The above theorem shows that similarity networks are inferentially complete subject to the restriction that only features that help to discriminate between some hypotheses are included in the model and that all hypotheses which are included in the model have a probability greater than zero. Consequently, diagnostic completeness is guaranteed too.

Theorem 6 (diagnostic completeness). *Let S be a similarity network of $P(h, u_1, \ldots, u_n)$. Then the conditional distribution $P(h \mid v_1, \ldots, v_m)$ can be computed from the parameters of S for every subset $\{v_1, \ldots, v_m\}$ of $\{u_1, \ldots, u_n\}$ provided $P(\boldsymbol{h}_i) \neq 0$ for every value $\boldsymbol{h}_i$ of h.*

Proof. To compute $P(h \mid v_1, \ldots, v_m)$ observe that $P(h \mid v_1, \ldots, v_m) = P(h \mid v'_1, \ldots, v'_l)$ where $v'_1, \ldots, v'_l$ is the subset of variables in $v_1, \ldots, v_m$ that are related to h. Theorem 5 states that the joint distribution $P(h, v'_1, \ldots, v'_l)$ can be computed from the parameters of S. The conditional probability $P(h \mid v'_1, \ldots, v'_l)$ can be computed from this joint distribution. □

The above two theorems provide us with a naive computation of the posterior probability of each hypothesis. This computation does not take into account the fact that $P(h, v'_1, \ldots, v'_l)$ might be too large to be explicitly computed or stored as a table. Moreover, the computation suggested by these proofs ignores the crucial observation that, in practice, all local networks are often constructed according to a common order, say $h, v'_1, \ldots, v'_l$, which usually reflects cause-effect relations or time constrains.

When such a common order exists some computations become much easier. In particular, Eq. (18) can be further developed, that is,

$$P(v_j \mid v'_1, \ldots v'_l, \boldsymbol{h}_m, [\![A_m]\!]) = P(v_j \mid \pi_{D_m}(v_j), [\![A_m]\!]) \tag{19}$$

where $v'_1, \ldots, v'_l$ are the variables depicted in the local network D_m and $\pi_{D_m}(v_j)$ are the parents of v_j in D_m. Consequently, $P(v_j \mid v'_1, \ldots v'_l, \boldsymbol{h}_m, [\![A_m]\!])$ need not be computed using *INFER* as done in the proof. It is stored explicitly at node v_j in the local network D_m.

Eq. (19) defines a Bayesian network M_i of $P(v_1, \ldots, v_m \mid \boldsymbol{h}_i)$ because, for each v_j, π_{M_i} is set to be v_j parents' set in D_m excluding h, and the parameters associated

with v_j in M_i are merely those associated with v_j in D_m. The collection of these local networks, one network for each hypothesis $\boldsymbol{h}_i$, forms an hypothesis-specific Bayesian network of $P(h, v_1, \ldots, v_m)$. We can now use our inference algorithm developed for Bayesian multinets to compute the posterior probability of each hypothesis.

The algorithm below summarizes this technique. Its first step uses arc-reversal transformations in order to reorient all local networks according to a common construction order. This step is given for the purpose of completeness, namely, to enable the algorithm to process similarity networks that are not constructed according to a common construction order. In practice, however, this step is usually not needed because similarity networks are constructed according to a common order of all relevant variables.

Algorithm (*Similarity network to Bayesian multinet conversion*).

Input: A similarity network S of $P(h, u_1, \ldots, u_n)$ based on a connected cover $A_1, \ldots, A_k$ of the values of h.

Output: A hypothesis-specific Bayesian multinet of $P(h, v_1, \ldots, v_l)$ where each v_i is depicted in some local network of S.

Notation:

- M_i is the comprehensive local network for hypothesis $\boldsymbol{h}_i$,
- D_i is the local network associated with A_i,
- $\pi_G(u)$ are the parents of u in a graph G,
- the probability associated with node u in M_i is $P_{M_i}(u \mid \pi_{M_i}(u), \boldsymbol{h}_i)$ and the probability associated with node u in D_m is $P_{D_m}(u \mid \pi_{D_m}(u), \boldsymbol{h}_m)$.

```
Reorient all local networks in S according to a common construction order
For each h_i construct M_i as follows
  For each v_j taken in order v_1, ..., v_l
    Find a path A_i, ..., A_m such that h_i ∈ A_i and v_j is depicted only in A_m
    Set π_{M_i}(v_j) to be π_{D_m}(v_j) \ {h}
    Set P_{M_i}(v_j | π_{M_i}(v_j), h_i) to be P_{D_m}(v_j | π_{D_m}(v_j), h_m)
```

As an example, let us examine how the algorithm processes the similarity network S in Fig. 7. Because the node ordering h, g, b, l is common to all local networks of S, the algorithm performs no arc reversals. Suppose the algorithm first builds the comprehensive local network M_e for the hypothesis *executive*. Because l appears in the local network for $\{worker, executive\}$ with only h as a parent, the algorithm makes l a root note in M_e, and sets $P_{M_e}(l \mid executive)$ to be $P_{w \vee e}(l \mid executive)$, where $w \vee e$ denotes the local network for $\{worker, executive\}$. The local network for $\{visitor, worker\}$ is the closest neighbor to the local network for $\{worker, executive\}$ that depicts g and b. Because the only parent of g in the local network for $\{visitor, worker\}$ is h, the algorithm makes g a root node in M_e. Because g and h are the parents of b in the local network for $\{visitor, worker\}$, the algorithm makes g a parent of b in M_e. The algorithm sets $P_{M_e}(g \mid executive)$ to be $P_{v \vee w}(g \mid worker)$ and $P_{M_e}(b \mid g, executive)$ to be $P_{v \vee w}(b \mid g, worker)$, where $v \vee w$ denotes the local network for $\{visitor, worker\}$. The algorithm constructs the comprehensive local networks for *worker*, *visitor* and *spy* similarly.

4.5. Similarity networks in the real world

The practical use of similarity networks for constructing and reasoning with probabilistic models consists of a few straightforward steps. In particular, to construct a joint probability distribution for $h, u_1, \ldots, u_n$, a user (1) constructs a similarity hypergraph for the values of h, (2) constructs a local network for each hyperedge in the hypergraph, and (3) assesses each local network. The inference algorithm described in Sections 4.2–4.4 can then compute the most likely hypothesis. The quality of a similarity network is strongly influenced by the first step, wherein the user identifies sets of similar hypotheses.

In addition to the theoretical arguments in favor of the use of similarity networks, this representation has also proven itself in practice. As mentioned in the introduction, the similarity network representation has facilitated the construction of several real-world expert systems including Pathfinder—an expert system that assists pathologists with the diagnosis of lymph-node diseases [14, 16, 18]. Pathfinder reasons about over 60 diseases (25 benign diseases, 9 Hodgkin's lymphomas, 18 non-Hodgkin's lymphomas, and 10 metastatic diseases) and over 140 features of disease, including morphologic, clinical, laboratory, immunological, and molecular biological findings. A formal evaluation of Pathfinder has demonstrated that its diagnostic accuracy is at least as good as that of the expert consulted to build it [17].

5. Generalized similarity networks

In previous sections we assume all hypotheses are mutually exclusive and are, therefore, represented as values of a single hypothesis variable h. In this section, we outline a way to relax this assumption, introducing a representation that allows several variables to represent hypotheses.

Definition. Let P be a probability distribution over $\{h_1, \ldots, h_r, u_1 \ldots u_n\}$ where $H = \{h_1, \ldots, h_r\}$ is a set of distinguished variables each representing a set of hypotheses. Denote the Cartesian product of the sets of values of the distinguished variables by $domain(H)$. Let $A_1, \ldots, A_k$ be a connected cover of $domain(H)$. A directed acyclic graph D_i is called a *local network of P associated with* A_i if D_i is a Bayesian network of $P(h_1, \ldots, h_r, v_1, \ldots, v_m \mid [\![A_i]\!])$ where $\{v_1, \ldots, v_m\}$ is the set of all variables in $\{u_1, \ldots, u_n\}$ that "help to discriminate" the values of A_i. The set of k local networks is called a *generalized similarity network* of P.

The generalized similarity network of Fig. 8, for example, represents the following problem:

> A *pair* of people approach the secured building and the guard tries to classify them as they approach. Each approaching person is either a worker (w), a visitor (v), or a spy (s). Assume that only workers converse (c) and that workers often arrive with other workers (because they car-pool).

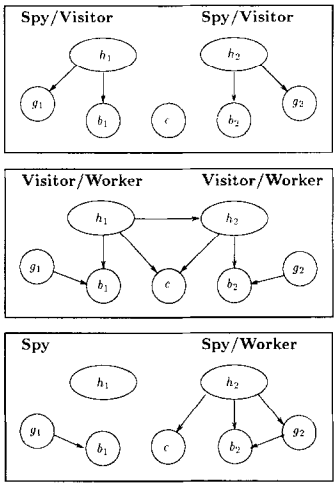


Fig. 8. A generalized similarity network with two hypothesis nodes.

Note, $H = \{h_1, h_2\}$ and $domain(H)$ consists of nine elements (x, y) where both x and y are drawn from the set $\{worker, visitor, spy\}$ whose elements we denote by $\{w, v, s\}$. The connected cover of $domain(H)$ corresponding to Fig. 8 is

$$\{(s,s),(v,s),(s,v),(v,v)\},$$
$$\{(v,v),(w,v),(v,w),(w,w)\},$$
$$\{(s,s),(s,w)\}.$$

The absence of a link between h_1 and h_2 in the top network encodes the fact that if the guard knew that one person is a spy or visitor, then this knowledge would not help him to decide whether the other person is a spy or a visitor. The existence of a link between h_1 and h_2 in the middle network encodes the fact that workers come in pairs more often than do visitors.

Node c should not have been included in the top local network, but it is drawn merely to highlight the independencies involving c. For the same reason, nodes g_1 and b_1 are drawn in the bottom local network. The remaining independence assertions encoded in Fig. 8 were described in previous sections or are obvious from the verbal description of the story.

The relationship between the hypothesis variables h_1 and h_2 in case of spies versus visitors is an example of *inter-hypothesis independence*, wherein two distinguished variables are independent given some hypotheses, but dependent given others. An inter-hypothesis independence assertion is represented in a generalized similarity network whenever a link between two distinguished variables exists in some local networks, but does not exist in other local networks. Such asymmetric independence assertions cannot be encoded in a non-generalized similarity network.

The results about similarity networks presented in the previous section can be extended in a straightforward manner to generalized similarity networks. An analogous definition for generalized Bayesian multinet is also self-evident.

6. Summary

In this article, we provide several enhancements to the similarity network representation originated by Heckerman [14].

First, we introduced the Bayesian multinet. We showed how the representation uses multiple Bayesian networks to encode asymmetric independence assertions, and how we can use these assertions to decrease storage requirements and increase the efficiency of inference. Next, we offered a definition of similarity networks which emphasizes the advantages of similarity networks compared to Bayesian multinets for knowledge acquisition. Then, we introduced an algorithm that converts a similarity network into a Bayesian multinet, thereby providing a general inference algorithm for similarity networks. In addition, we described generalized similarity networks which facilitate the representation of non-mutually exclusive hypotheses. We hope that this work will encourage a line of research that strives to devise additional graphical representation schemes of salient types of asymmetric independence that further simplify knowledge acquisition and inference.

Finally, we note that the computational advantages that result from sparse matrix manipulations, as suggested in [20], can also be combined with knowledge about asymmetric independence. In fact, we have argued that any inference algorithm for Bayesian networks can also be applied to a similarity network. One should emphasize, however, that these two sources of computational savings are disjoint since inference methods that rely on sparse matrices arise from the presence of zero probabilities whereas inference methods that rely on asymmetric independence constrains arise due to equalities between certain probabilities.

Acknowledgments

An earlier version of this article has been presented in the Seventh Conference on Uncertainty in Artificial Intelligence. Some of this work was done while the first author was at Northrop Research and Technology Center and the second author was at the University of California at Los Angeles. We thank Mark Peot for suggesting an early version of the inference algorithm for strictly positive distributions.

References

[1] S. Andreassen, M. Woldbye, B. Falck and S. Andersen, MUNIN: a causal probabilistic network for interpretation of electromyographic findings, in: *Proceedings IJCAI-87*, Milan (Morgan Kaufmann, San Mateo, CA, 1987) 366–372.

[2] J. Breese, E. Horvitz, M. Peot, R. Gay and G. Quentin, Automated decision-analytic diagnosis of thermal performance in gas turbines, in: *Proceedings International Gas Turbine and Aeroengine Congress and Exposition* (American Society of Mechanical Engineers, Cologne, 1992).

[3] W. Buntine, Theory refinement on Bayesian networks, in: *Proceedings Seventh Conference on Uncertainty in Artificial Intelligence*, Los Angeles, CA (Morgan Kaufmann, Los Altos, CA, 1991) 52–60.

[4] G.F. Cooper, Computational complexity of probabilistic inference using Bayesian belief networks, *Artif. Intell.* **42** (1990) 393–405.

[5] G.F. Cooper and E. Herskovits, A Bayesian method for constructing Bayesian belief networks from databases, in: *Proceedings Seventh Conference on Uncertainty in Artificial Intelligence*, Los Angeles, CA (Morgan Kaufmann, Los Altos, CA, 1991) 86–94.

[6] G.F. Cooper and E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, *Mach. Learn.* **9** (1992) 309–347.

[7] R. Cox, Probability, frequency and reasonable expectation, *Am. J. Phys.* **14** (1946) 1–13.

[8] D. Geiger, Probabilistic networks, in: S. Shapiro, ed., *Encyclopedia of Artificial Intelligence* (Wiley, New York, 2nd ed., 1992) 1201–1209.

[9] D. Geiger and D. Heckerman, Separable and transitive graphoids, in: *Proceedings Sixth Conference on Uncertainty in Artificial Intelligence*, Boston, MA (Association for Uncertainty in Artificial Intelligence, Mountain View, CA, 1990) 538–545.

[10] D. Geiger and D. Heckerman, Learning Gaussian networks, in: *Proceedings Tenth Conference on Uncertainty in Artificial Intelligence*, Seattle, WA (Morgan Kaufmann, Los Altos, CA, 1994) 235–243.

[11] D. Geiger and J. Pearl, On the logic of causal models, in: *Proceedings Fourth Workshop on Uncertainty in Artificial Intelligence*, Minneapolis, MN (Association for Uncertainty in Artificial Intelligence, Mountain View, CA, 1988) 136–147; also in: R. Shachter, T. Levitt, L. Kanal and J. Lemmer, eds., *Uncertainty in Artificial Intelligence* **4** (North-Holland, New York, 1990) 3–14.

[12] D. Geiger, T. Verma and J. Pearl, Identifying independence in bayesian networks, *Networks* **20** (1990) 507–534.

[13] M. Goldbaum, B. Cote, A. Listhaus, D. Heckerman, E. Horvitz and J. Breese, IntellEye, an expert system for diagnostic ophthalmologic diseases from images of the ocular fundus, in: *Investigative Ophthalmology and Visual Science*, Supplement (1991).

[14] D. Heckerman, *Probabilistic Similarity Networks* (MIT Press, Cambridge, MA, 1991).

[15] D. Heckerman, D. Geiger and D. Chickering, Learning Bayesian networks, *Mach. Learn.* **20** (1995) 197–243.

[16] D. Heckerman, E. Horvitz and B. Nathwani, Toward normative expert systems, Part I: The Pathfinder project, *Methods Inf. Medicine* **31** (1992) 90–105.

[17] D. Heckerman and B. Nathwani, An evaluation of the diagnostic accuracy of Pathfinder, *Comput. Biomed. Res.* **25** (1992) 56–74.

[18] D. Heckerman and B. Nathwani, Toward normative expert systems, Part II: Probability-based representations for efficient knowledge acquisition and inference, *Methods Inf. Medicine* **31** (1992) 106–116; also in: J. van Bemmel and A. McCray, eds., *Yearbook of Medical Informatics* (International Medical Informatics Association, Rotterdam, 1993) 430–440.

[19] R. Howard and J. Matheson, Influence diagrams, in: R. Howard and J. Matheson, eds., *Readings on the Principles and Applications of Decision Analysis*, Vol. II (Strategic Decisions Group, Menlo Park, CA, 1981) 721–762.

[20] F. Jensen and S. Andersen, Approximations in Bayesian belief universes for knowledge based systems, in: *Proceedings Sixth Conference on Uncertainty in Artificial Intelligence*, Cambridge, MA (Morgan Kaufmann, Los Altos, CA, 1990) 162–169.

[21] F. Jensen, S. Lauritzen and K. Olesen, Bayesian updating in recursive graphical models by local computations, *Comput. Stat. Quart.* **4** (1990) 269–282.

[22] F. Jensen, K. Olesen and S. Andersen, An algebra of Bayesian belief universes for knowledge-based systems, *Networks* **20** (1990) 637–660.

[23] J. Kim and J. Pearl, A computational model for causal and diagnostic reasoning in inference engines, in: *Proceedings IJCAI-83*, Karlsruhe (1983) 190–193.

[24] H. Kyburg and H. Smokler, *Studies in Subjective Probability* (Wiley, New York, 1980).

[25] W. Lam and F. Bacchus, Using causal information and local measures to learn Bayesian networks, in: *Proceedings Ninth Conference on Uncertainty in Artificial Intelligence*, Washington, DC (Morgan Kaufmann, Los altos, CA, 1993) 243–250.

[26] S. Lauritzen and D. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems, *J. Roy. Stat. Soc. B* **50** (1988) 157–224.

[27] B. Nathwani, D. Heckerman, E. Horvitz and T. Lincoln, Integrated expert systems and videodisc in surgical pathology: an overview, *Human Path.* **21** (1990) 11–27.
[28] G. Nino-Murcia and M. Shwe, An expert system for diagnosis of sleep disorders, in: M. Chase, R. Lydic and C. O'Connor, eds., *Sleep Research* **20** (Brain Information Service, Los Angeles, CA, 1991) 433.
[29] S. Olmsted, On representing and solving decision problems, Ph.D. Thesis, Department of Engineering-Economic Systems, Stanford University, Stanford, CA (1983).
[30] J. Pearl, Reverend Bayes on inference engines: a distributed hierarchical approach, in: *Proceedings AAAI-82*, Pittsburgh, PA (AAAI Press, Menlo Park, CA, 1982) 133–136.
[31] J. Pearl, Fusion, propagation, and structuring in belief networks, *Artif. Intell.* **29** (1986) 241–288.
[32] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, San Mateo, CA, 1988).
[33] J. Pearl and T. Verma, A theory of inferred causation, in: J. Allen, R. Fikes and E. Sandewall, eds., *Knowledge Representation and Reasoning: Proceedings Second International Conference* (Morgan Kaufmann, San Mateo, CA, 1991) 441–452.
[34] L. Savage, *The Foundations of Statistics* (Dover, New York, 1954).
[35] R. Shachter, Evaluating influence diagrams, *Oper. Res.* **34** (1986) 871–882.
[36] R. Shachter and C. Kenley, Gaussian influence diagrams, *Manage. Sci.* **35** (1989) 527–550.
[37] P. Spirtes, C. Glymour and R. Scheines, *Causation, Prediction, and Search* (Springer, New York, 1993).
[38] T. Verma and J. Pearl, Causal networks: semantics and expressiveness, in: *Proceedings Fourth Workshop on Uncertainty in Artificial Intelligence*, Minneapolis, MN (Association for Uncertainty in Artificial Intelligence, Mountain View, CA, 1988) 352–359; also in: R. Shachter, T. Levitt, L. Kanal and J. Lemmer, eds., *Uncertainty in Artificial Intelligence* 4 (North-Holland, New York, 1990) 69–76.

ELSEVIER

Artificial Intelligence 82 (1996) 75-127

Artificial Intelligence

ALX, an action logic for agents with bounded rationality

Zhisheng Huang [1], Michael Masuch [2], László Pólos

Center for Computer Science in Organization and Management (CCSOM), University of Amsterdam, Sarphatistraat 143, 1018 GD Amsterdam, Netherlands

Received June 1993; revised November 1994

Abstract

We propose a modal action logic that combines ideas from H.A. Simon's *bounded rationality*, S. Kripke's *possible world semantics*, G.H. von Wright's *preference logic*, Pratt's *dynamic logic*, Stalnaker's *minimal change* and more recent approaches to *update semantics*. ALX (the xth action logic) is sound, complete and decidable, making it the first complete logic for two-place preference operators. ALX avoids important drawbacks of other action logics, especially the counterintuitive necessitation rule for goals (every theorem must be a goal) and the equally counterintuitive closure of goals under logical implication.

1. Introduction

Action logics are usually developed for (hypothetical) use by intelligent robots [6, 14, 40, 59, 71], as a description language of program behavior [24, 47], or as a contribution to philosophical logic [64]. Our effort is motivated by a different concern. We want to develop a formal language for social science theories, especially for theories of organizations. The difference in motivation leads to a new approach to action logic. We combine ideas from various strands of thought, notably H.A. Simon's notion of *bounded rationality*, G.H. Wright's approach to *preferences*, Kripke's *possible world semantics* in combination with binary modal operators, Pratt's *dynamic logic*, Stalnaker's notion of *minimal change*, and more recent ideas from *belief revision* and *update semantics* [15, 25]. Although fairly simple in its construction, ALX is good at handling some crucial problems of action logic. In particular, ALX avoids the counterintuitive necessitation

[1] E-mail: huang@ccsom.uva.nl.

[2] Corresponding author. Fax: (31-20) 5252800. E-mail: michael@ccsom.uva.nl.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0004-3702(94)00090-5

rule for goals (every theorem must be a goal) and the equally counterintuitive closure of goals under logical implication. ALX is complete, decidable and enjoys the finite model property.

2. The framework for ALX

Most social science theories are expressed in natural language. They lack a formal scaffold that would allow one to check their consistency in a rigorous fashion, or to disambiguate natural language statements. As a consequence, these theories have acquired a reputation for "softness"—a soft way of saying that their logical properties are somewhat dubious. Reformulating them in a formal language with known properties would make consistency checking or disambiguation easier. Also, it would pave the way for other tasks, such as the examination of a theory's deductive closure properties. Understanding the deductive closure properties of a set of formulas is essential to automating the generation of theories from a given set of assumptions and introducing AI into theory building [45].

We focus on action logic as a formal language, because actions of individual or collective agents are key to the understanding of social phenomena. In fact, most social scientists agree that adequate theories of social relations must be action theories [1, 12, 16, 39, 52, 61]. Yet actions engender change and change is notoriously hard to grasp in the extensional context of first order languages [11]. This drives our attempt to develop a new logic, rather than taking First-Order Logic off the shelf. We call the new logic ALX (the xth action logic).

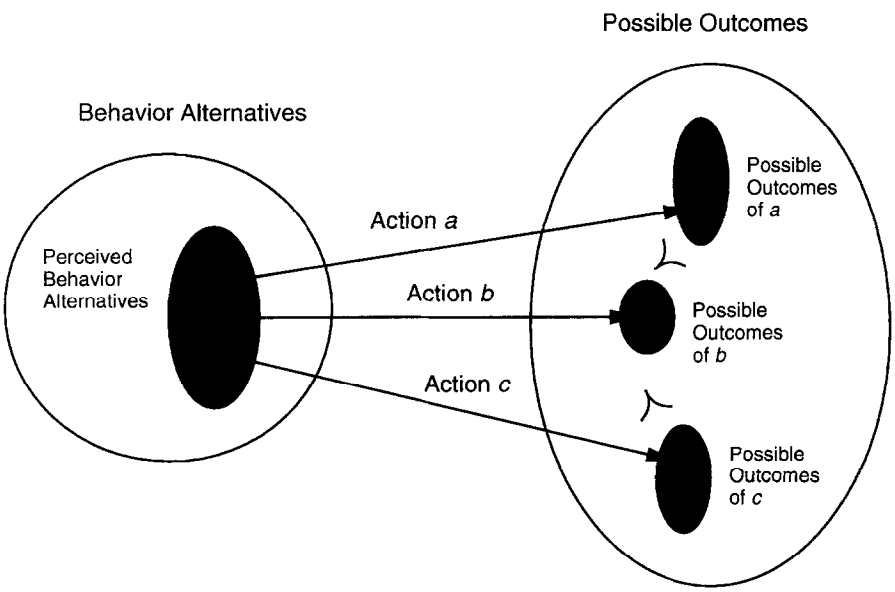


Fig. 1. Simon's bounded rationality.

Herbert A. Simon's conceptualization of *bounded rationality* [65] serves as a point of departure (see Fig. 1). His approach is intuitively appealing and had great impact on the postwar social sciences. Simon wanted to overcome the omniscience claims of the traditional conceptualizations of rational action. He assumed (1) an agent, with (2) a set of behavior alternatives, (3) a set of future states of affairs (each such state being the outcome of a choice among the behavior alternatives), and (4) a preference order over future states of affairs. The omniscient agent, endowed with "perfect rationality", would know all behavior alternatives and the exact outcome of each alternative; the agent would also have a complete preference order for those outcomes. An agent with bounded rationality, in contrast, would not know all the alternatives, would not know the exact outcome of each, and would not have a complete preference order for those outcomes.

Kripke's *possible world semantics* provides a natural setting for Simon's conceptualization. We assume a set of possible worlds with various relations defined over this set (we may also call those possible worlds *states*). One can see a behavior alternative as a mapping from states to states, so each behavior alternative constitutes an accessibility relation. An accessibility relation, in turn, can be interpreted as an opportunity for action, i.e., as an opportunity for changing the world by moving from a given state to another state. Accessibility relations are expressed by one-place modal operators, as in dynamic logic [24]. For example, the formula $\langle a \rangle \phi$ would express the fact that the agent has an action a at his disposal such that effecting a in the present situation would result in the situation denoted by proposition ϕ.

The perfectly rational agent would have a complete description of his actual state, a complete knowledge of all accessibility relations and a complete preference order over states. Agents with bounded rationality are less well informed. They may have an incomplete description of their actual state (we call those descriptions *situations*), incomplete knowledge of the accessibility relations, and an incomplete preference order over situations. In particular, one can assume that their knowledge and the range of their actions is finite. As it turns out, this assumption is crucial for defining a straightforward concept of goal in ALX's object language.

Situations are represented as sets of states and expressed by propositions. Propositions, in turn, denote the set of states where they obtain. So, the more specific an agent's knowledge about a situation, the more detailed the propositional description of that situation would be. The limit case, a complete description, would uniquely identify one state. Less specific descriptions would lack that uniqueness, identifying the set of those states where the description would hold (but remaining uncommitted about other "aspects" not covered by the description). This is the standard approach to representing incomplete information, used in denotational semantics [62,63] and epistemic logic [20].[3]

[3] Framing bounded rationality in terms of possible worlds semantics reveals a fine point usually ignored: one can see that omniscience—the limiting case—is contingent upon the choice of the language. Full rationality in an absolute sense would require a language isormorphic to the universe "out there", but such a language is not available. Any formal theory about full rationality has to make simplifying assumptions about the world, but those assumption, by their simplifying nature, violate the ontology of full rationality in some sense or another.

Preferences—not goals—provide the basis for rational action in ALX. Following von Wright [72], a preference statement is understood as a statement about situations. For example, the statements that "I prefer oranges to apples" is interpreted as the fact that "I prefer the states in which I have an orange to the states in which I have an apple." Following von Wright again, we assume that an agent who claims to prefer oranges to apples should prefer a situation where he has an orange but *no* apple to a situation where he has an apple but *no* orange. Preferences are expressed via two-place modal operators; if the agent prefers the proposition ϕ to the proposition ψ, we write $\phi \boldsymbol{P} \psi$.

Normally, the meaning of a preference statement is context-dependent, even when this is not made explicit. An agent may claim to prefer an apple to an orange—and actually mean it—but he may prefer an orange to an apple later—perhaps because then he already had an apple. To capture this context dependency, we borrow the notion of minimal change from Stalnaker's approach to conditionals [68]. The idea is to apply the conjunction expansion principle only to situations that are minimally different from the agent's present situation—just as different as they really need to be in order to make the propositions true about which preferences are expressed. We introduce a binary function, *cw*, to the semantics that determines a set of "closest" states relative to a given state, such that the new states fulfill some specified conditions, but resemble the old state as much as possible in all other respects. For situations (sets of states), we apply *cw* to each element of the situation separately. This allows us to avoid some technical problems arising in conditional logic [38, 50, 68].[4]

ALX provides a complete syntactic characterization of preferences, so one can derive new preference statements from old ones by using its machinery. Closing the set of preference statements under the rules of inferences yields a preference order that serves as the basis for deriving goals. *Goals*, in turn, are defined in terms of preferences and accessibilities (we will argue that there are several plausible goal definitions, corresponding to increasingly stronger notions of rationality). Note that goals need not be unique; this follows from the fact that world descriptions and preference orders are usually incomplete. Also, the goal set need not be closed under logical implication, so agents need not treat undesired consequences of desired outcomes (e.g., pain as a consequence of having one's teeth repaired) as goals (as opposed to action logics that use the concept of "goal" as the primitive notion of rational guidance).

3. Formal syntax and semantics

3.1. Syntax

ALX is a multimodal propositional logic. The propositional alphabet consists of a countable set of lower-case Latin symbols p_i to denote primitive propositions. The action alphabet has a finite set of actions a_i. Lower-case Greek letters $\phi, \psi, \rho, \ldots$ (with or without subscript) denote well-formed formulae. We have $\langle a \rangle \psi$ to denote the one-place existential accessibility relation for action a and $\boldsymbol{P}$ to denote the two-place preference

[4] The reader is referred to the discussion part of this paper for details.

relation. $\circ$ serves as two-place operator for *updates*; updates are changes caused by an action. Note that updates in ALX refer to real state changes, not epistemological ones [15], so an update does not produce a new knowledge state, but a new situation.

Definition 1 (*Syntax*). Let $ATOM = \{p_i : i < \omega\}$ and $ACTION = \{a_1, \ldots, a_k\}$ for some $k \in \omega$ with ω standing for the ordinality of natural numbers. The set of formulae FML is defined recursively as follows:

- $ATOM \subseteq FML$,
- $\phi \in FML \Rightarrow \neg\phi \in FML$,
- $\phi, \psi \in FML \Rightarrow (\phi \wedge \psi) \in FML$,
- $\phi \in FML, a \in ACTION \Rightarrow (\langle a \rangle \phi) \in FML$,
- $\phi, \psi \in FML \Rightarrow (\phi \circ \psi) \in FML$,
- $\phi, \psi \in FML \Rightarrow (\phi \boldsymbol{P} \psi) \in FML$.

Define $\bot$ as $\phi \wedge \neg\phi$ for an arbitrary ϕ, and $[a]\phi$ as $\neg\langle a \rangle \neg\phi$. Define the Boolean connectives $\{\vee, \rightarrow, \leftrightarrow\}$ and the truth constant $\top$ from the given Boolean connectives in the usual way.

3.2. Semantics

Definition 2 (*ALX models*). We call $M = \langle W, cw, \succ, \{R^a\}_{a \in ACTION}, V \rangle$ an ALX model if:

- W is a set of possible worlds,
- $cw : W \times \mathcal{P}(W) \rightarrow \mathcal{P}(W)$ is a closest world function,
- $\succ \subseteq \mathcal{P}(W) \times \mathcal{P}(W)$ is a comparison relation for preferences,
- $R^a \subseteq W \times W$ is an accessibility relation for each a in $ACTION$,
- $V : ATOM \rightarrow \mathcal{P}(W)$ is an assignment function for primitive propositions,

and if M satisfies the following conditions:

(CS1) $cw(w, X) \subseteq X$.

(CS2) $w \in X \Rightarrow cw(w, X) = \{w\}$.

(CSC) $cw(w, X) \cap Y \subseteq cw(w, X \cap Y)$.

(NORM) $(\emptyset \not\succ X)$, $(X \not\succ \emptyset)$.

(TRAN) $cw(w, X \cap \overline{Y}) \succ cw(w, Y \cap \overline{X})$ and $cw(w, Y \cap \overline{Z}) \succ cw(w, Z \cap \overline{Y})$
$\Rightarrow cw(w, X \cap \overline{Z}) \succ cw(w, Z \cap \overline{X})$,
where $\overline{Y} = W - Y$.

(CS1), (CS2) and (CSC) constrain the closest world function. (CS1) ensures that the closest ϕ-worlds (relative to a given world) are indeed ϕ-worlds; (CS2) ensures that w is its own (and unique) closest ϕ-world if ϕ is true at w. (CSC) says that if ψ is true at the closest ϕ-world, then the closest ϕ-world is also a closest ϕ-and-ψ-world. (NORM) and (TRAN) constrain the semantic preference relation. They require normality and transitivity; "normality" stipulates that no comparison between two sets of worlds would involve an empty set of worlds.

In the following, we will use $M = \langle W, cw, \succ, R^a, V\rangle$ to denote $M = \langle W, cw, \succ, \{R^a\}_{a \in ACTION}, V\rangle$ if the omission causes no ambiguity.

Definition 3 (*Interpretation function*). Let *FML* be as above and let $M = \langle W, cw, \succ, R^a, V\rangle$ be an ALX model. The interpretation function $[\![\ ]\!]_M : FML \to \mathcal{P}(W)$ is defined as follows:

$$
\begin{aligned}
[\![p_i]\!]_M &= V(p_i),\\
[\![\neg\phi]\!]_M &= W \setminus [\![\phi]\!]_M,\\
[\![\phi \wedge \psi]\!]_M &= [\![\phi]\!]_M \cap [\![\psi]\!]_M,\\
[\![\langle a\rangle\phi]\!]_M &= \{w \in W : \exists w' \in W(R^a ww' \text{ and } w' \in [\![\phi]\!]_M)\},\\
[\![\phi \circ \psi]\!]_M &= \{w \in W : \exists w' \in W(w' \in [\![\phi]\!]_M \text{ and } w \in cw(w', [\![\psi]\!]_M))\},\\
[\![\phi \boldsymbol{P} \psi]\!]_M &= \{w \in W : cw(w, [\![\phi \wedge \neg\psi]\!]_M) \succ cw(w, [\![\neg\phi \wedge \psi]\!]_M)\}.
\end{aligned}
$$

The interpretation of the primitive propositions and the Boolean connectives is straightforward. A proposition letter p_i evaluates to the set of worlds where p_i obtains, the negation of a proposition ϕ evaluates to the complement of the ϕ-worlds, and the conjunction of ϕ and ψ evaluates to the intersection of the ϕ-worlds and the ψ-worlds. The interpretation of $\langle a\rangle\phi$ yields the set of worlds from where the agent can access at least one ϕ-world via action a. We use the "existential" version of the action modality, because real-life decisions typically depend on the *possibility* of a specific action in a specific situation.

The interpretation of $\phi \circ \psi$ yields the set of worlds where ψ holds so that one could have got there from a closest ϕ-world. Note that $\phi \circ \psi$ is a backward-looking operator [15]. Note also that $\phi \circ \psi$ is closely related to the intensional conditional ("wiggle") known from Stalnaker's and D. Lewis' work [38,68], where $\phi \rightsquigarrow \psi$ is meant to express "if ϕ were the case, then ψ would be the case". [5] One can express (although not term-define) the wiggle in terms of the update operator via the so-called *Ramsey rule* [15]:

$$\vdash (\chi \to (\phi \rightsquigarrow \psi)) \Leftrightarrow \vdash ((\chi \circ \phi) \to \psi).$$

The interpretation of $\phi \boldsymbol{P} \psi$ yields the set of worlds where the agent prefers (at each of those worlds) the closest ϕ-and-not-ψ-worlds to the closest ψ-and-not-ϕ-worlds.

Define the forcing relation as:

$$M, w \Vdash \phi \overset{\text{def}}{\Longleftrightarrow} w \in [\![\phi]\!]_M.$$

Definition 4 (*The logic ALX*). Let *FML* be as above, let *Mod* be the class of all ALX models and let $[\![\ ]\!]_M$ be as above too, defined for every model $M \in Mod$. We call the logic ALX = $\langle FML, Mod, [\![\ ]\!]_M\rangle$ ALX logic.

[5] In fact, in Stalnaker, the "wiggle" is a "corner" (>); we prefer the $\rightsquigarrow$ because it frees the > for other uses.

Define the semantic consequence relation, $\models$, as usual:

$$M = \langle W, cw, \succ, R^a, V\rangle \models \phi \overset{\text{def}}{\Longleftrightarrow} (\forall w \in W)(M, w \Vdash \phi).$$

$$M \models \Gamma \overset{\text{def}}{\Longleftrightarrow} (\forall \gamma \in \Gamma)(M \models \gamma).$$

$$Mod(\Gamma) \overset{\text{def}}{\Longleftrightarrow} \{M \in Mod : M \models \Gamma\}.$$

$$\Gamma \models \phi \overset{\text{def}}{\Longleftrightarrow} Mod(\Gamma) \subseteq Mod(\{\phi\}).$$

Definitions 2–4 provide a semantic characterization of ALX. The next definition provides a complete syntactic characterization.

Definition 5 (*ALX inference system*). Let ALXS be the following set of axioms and rules of inference.

(BA)	*all propositional tautologies.*	
(A1)	$\langle a\rangle\bot$	$\leftrightarrow \bot$
(A2)	$\langle a\rangle(\phi \vee \psi)$	$\leftrightarrow \langle a\rangle\phi \vee \langle a\rangle\psi$
(A3)	$\langle a\rangle(\phi \wedge \psi)$	$\rightarrow \langle a\rangle\phi \wedge \langle a\rangle\psi$
(U1)	$\phi \circ \psi$	$\rightarrow \psi$
(U2)	$\phi \wedge \psi$	$\rightarrow \phi \circ \psi$
(U3)	$\neg(\phi \circ \bot), \neg(\bot \circ \phi)$	
(U4)	$(\phi \vee \psi) \circ \chi$	$\leftrightarrow \phi \circ \chi \vee \psi \circ \chi$
(U5)	$(\phi \wedge \psi) \circ \psi$	$\rightarrow \phi$
(U6)	$(\phi \circ \psi) \wedge \chi$	$\rightarrow \phi \circ (\psi \wedge \chi)$
(CEP)	$\phi \boldsymbol{P} \psi$	$\leftrightarrow (\phi \wedge \neg\psi)\boldsymbol{P}(\neg\phi \wedge \psi)$
(TR)	$(\phi \boldsymbol{P} \psi) \wedge (\psi \boldsymbol{P} \chi)$	$\rightarrow (\phi \boldsymbol{P} \chi)$
(N)	$\neg(\bot \boldsymbol{P} \phi), \neg(\phi \boldsymbol{P} \bot)$	
(MP)	$\vdash \phi$ & $\vdash \phi \rightarrow \psi$	$\Rightarrow \vdash \psi$
(NECA)	$\vdash \phi$	$\Rightarrow \vdash [a]\phi$
(MONA)	$\vdash \langle a\rangle\phi$ & $\vdash \phi \rightarrow \psi$	$\Rightarrow \vdash \langle a\rangle\psi$
(MONU)	$\vdash \phi \circ \psi$ & $\vdash \phi \rightarrow \phi'$	$\Rightarrow \vdash \phi' \circ \psi$
(SUBA)	$\vdash (\phi \leftrightarrow \phi')$	$\Rightarrow \vdash (\langle a\rangle\phi) \leftrightarrow (\langle a\rangle\phi')$
(SUBU)	$\vdash (\phi \leftrightarrow \phi')$ & $\vdash (\psi \leftrightarrow \psi')$	$\Rightarrow \vdash (\phi \circ \psi) \leftrightarrow (\phi' \circ \psi')$
(SUBP)	$\vdash (\phi \leftrightarrow \phi')$ & $\vdash (\psi \leftrightarrow \psi')$	$\Rightarrow \vdash (\phi \boldsymbol{P} \psi) \leftrightarrow (\phi' \boldsymbol{P} \psi')$

Most axioms are straightforward. As usual, we have the propositional tautologies (BA). Since ALX is a *normal* modal logic, the absurdum is not true anywhere, so it is not accessible (A1). The action modalities behave as usual, so they distribute over disjunction both ways, but over conjunction only in one direction (A2) and (A3).

Indeed, we can get to ϕ-or-ψ-worlds via action a if and only if we can get via a to a ϕ-world or to a ψ-world. However, being able to get to ϕ-worlds via action a *and* being able to get to ψ-worlds via action a does not necessarily mean that a can get us to a world that is both ϕ and ψ.

As mentioned above, $\circ$ is a backward-looking operator. So, a successful ψ-update ends up in a ψ-world (U1) and the truth of both ϕ and ψ at a world allows us to perform a vacuous ψ-update, i.e., to remain at that world (U2). (U3) reiterates the normality condition for updates. Since there is no world where the absurdum is true, an update with the absurdum cannot succeed. (U4) posits the left distribution of the disjunction over the update operator. The intuition is that if we have got to a χ-world from a ϕ- *or* a ψ-world, we have updated either from a ϕ-world or from a ψ-world. (U5) tells us that a void update is not going to change any conditions. (U6) posits that if χ holds after updating ϕ with ψ, then we can update ϕ with $\psi \wedge \chi$ and obtain the same result. Readers more familiar with closest world functions may already sense how the update operator will mimic the closest world function in the syntax, helping to construct a canonical model in the completeness proof.

The axioms for the preference operator posit the conjunction expansion principle (CEP), transitivity (TR), and normality (N). So, if we prefer ϕ to ψ, we will also prefer the absence of ψ to the absence of ϕ. If we prefer ϕ to ψ, we are apt to prefer ϕ-and-not-ψ to ψ-and-not-ϕ. We have transitivity because we think that it is a natural principle of preference orders. Normality is required to avoid inconsistent preference statements. For example, without normality, we get a violation of irreflexivity via the only-if part of the contraposition principle.[6] We need not state irreflexivity as an axiom, since it is derivable from (CEP) and (N). By the same token, we can derive contraposition and asymmetry for the preference operator.

Proposition 6 (More properties of the preference operator). *The following formulas are theorems of ALX*:

(CP) $\phi \boldsymbol{P} \psi \leftrightarrow (\neg\psi)\boldsymbol{P}(\neg\phi)$,

(IR) $\neg(\phi \boldsymbol{P} \phi)$,

(NT) $\neg(\top \boldsymbol{P} \phi), \neg(\phi \boldsymbol{P} \top)$,

(AS) $\phi \boldsymbol{P} \psi \rightarrow \neg(\psi \boldsymbol{P} \phi)$.

Proof. See Appendix A. □

We have modus ponens and the necessitation rule for the universal action modality (NECA) and monotonicity for the existential action modality. For the update operator, we have *left* monotonicity, but not *right* monotonicity, the intuition being that a move from a ϕ-world to the closest ψ-world w might end up at a different world than the move to the closest ψ'-world even if ψ implies ψ' at w. Logically equivalent propositions are

[6] An alternative method for preserving consistency, suggested by von Wright [72] and used by Hansson [22], requires the "independence" of propositions in certain axioms. We have used this approach in [28], but it is less straightforward because the definition of "independence" is nontrivial.

substitutible in action, update and preference formulas (SUBA), (SUBU), (SUBP). Note that we do *not* have monotonicity for preferences. Because of this, we are able to avoid the counterintuitive deductive closure of goals.

4. Formal properties of ALX

ALX has pleasant logical properties. We have:

Proposition 7 (Soundness of ALXS). *ALXS is sound, i.e., for an arbitrary set of formulas Δ and an arbitrary formula ϕ,*

$$\Delta \vdash_{\mathrm{ALX}} \phi \Rightarrow \Delta \models \phi.$$

Proof. See Appendix A. □

Next, we have:

Proposition 8 (Completeness of ALXS). *ALX is complete, i.e., for an arbitrary set of formulas Δ and an arbitrary formula ϕ,*

$$\Delta \models \phi \Rightarrow \Delta \vdash_{\mathrm{ALX}} \phi.$$

Proof. See Appendix A. □

Furthermore, ALX is decidable. Stronger even, we have the finite model property for ALX, that is, for each non-theorem ψ there exists a finite model that provides a counterexample for ψ. Since ALX is recursively axiomatizable, this means that ALX is decidable.

Definition 9 (*Finite model property*). A logic S is said to have the finite model property, iff, for arbitrary ϕ such that $\nvdash_S \phi$, there exists a finite model M such that:
(1) $\exists w(M, w \Vdash \neg\phi)$,
(2) $\forall \rho(\vdash_S \rho \Rightarrow \forall w(M, w \Vdash \rho))$.

Theorem 10. *ALX has the finite model property.*

Proof. See Appendix A. □

ALXS is finite, so ALX is finitely axiomatizable. The finite model property together with finite axiomatizability imply decidability (cf. [31, p. 153]. As a consequence, we have:

Corollary 11. *ALX is decidable.*

5. Applying ALX

ALX provides considerable flexibility in defining new modal operators by using the three primitive operators. We concentrate on operators of potential use in defining goals.

In the following, we assume that the preference order of an agent is finite and hence the corresponding set of preference statements. Call this set Σ_P. Recall furthermore that the range of action alternatives is finite, too (as stipulated in Definition 1). Suppose that $\Sigma_P = \{\psi_1, \ldots, \psi_n\}$. In the following, we use the notation $\forall\psi\Phi(\psi)$ (where $\Phi(\psi)$ is an formula that contains ψ) to denote $\Phi(\psi_1) \wedge \cdots \wedge \Phi(\psi_n)$ and $\exists\psi\Phi(\psi)$ to denote $\Phi(\psi_1) \vee \cdots \vee \Phi(\psi_n)$.

Define accessibility as follows:

Definition 12 (*Accessibility*). Let $\boldsymbol{A}\phi$ stand for the fact that situation ϕ is accessible via an action. Define:

$$\boldsymbol{A}\phi \overset{\text{def}}{\Longleftrightarrow} \langle a_1\rangle\phi \vee \langle a_2\rangle\phi \vee \cdots \vee \langle a_k\rangle\phi.$$

Thus, operator $\boldsymbol{A}$ acts as an existential quantifier over action terms; if situation ϕ is accessible via an arbitrary action, then we have $\boldsymbol{A}\phi$. Note how bounded rationality impinges on this definition. In defining accessibility we can stay inside the object language because we assume that agents are not omnipotent and have only finitely many action alternatives at their disposal.

Define a "good" situation ϕ as a situation that the agent prefers to its negation and conversely for a "bad" situation.

Definition 13 (*Good, bad situations*). Let $\boldsymbol{GO}\phi$ stand for a "good" situation ϕ and $\boldsymbol{BA}\phi$ for a "bad" situation ϕ. Define:

$$\boldsymbol{GO}\phi \overset{\text{def}}{\Longleftrightarrow} \phi\boldsymbol{P}\neg\phi, \qquad \boldsymbol{BA}\phi \overset{\text{def}}{\Longleftrightarrow} \neg\phi\boldsymbol{P}\phi.$$

Define an element of the agent's preference order in the obvious way:

Definition 14 (*Element of the preference order*). Let $\boldsymbol{PO}\phi$ stand for an element in the agent's preference order. Define:

$$\boldsymbol{PO}\phi \overset{\text{def}}{\Longleftrightarrow} (\phi\boldsymbol{P}\rho) \vee (\rho\boldsymbol{P}\phi).$$

5.1. Goals

Goals are a crucial notion for action logics. Following the basic notions of bounded rationality (and, for that matter, standard decision theory), we derive goals from preferences; they are not a primitive notion as in other action logics. But there are many ways to base goals on preferences. A situation may be singled out as a goal simply because it is better than its negation, or, perhaps, because it is better than other situations; it may be satisficing, outstanding (extremal), or optimal. Bounded rationality is often identified

with the notion that agents do not optimize, at least not in the sense of putting much energy into the search for extremal solutions; instead, they are said to satisfice. However, the reduction of bounded rationality to satisficing is misleading. Satisficing is, indeed, relevant when the existence, or the accessibility, of potential goal states is *unknown*. If a known alternative meets a given aspiration level, then, as a rule, the agent will not *search* for a better state; conversely, if no known alternative meets the aspiration level, the agent will search for better solutions, at least up to a certain point. However, agents might act irrational if they do not pursue *known* better accessible alternatives; if they never do, aspiration levels could only go down). Bounded rationality has been introduced in order to develop a more realistic framework of rational decision making, and a drive for *improving* one's situation is apparent in many human decisions.

We present four goal definitions (good, satisficing, extremal, optimal), and discuss some obvious modifications of these definitions.

Agents might opt for a state simply because it is better than its negation, particularly if only a few alternatives are considered. For example, if an agent finds himself late at night far from home without a car, he might base his decision to take a taxi on the simple deliberation that it is better to take a taxi than not to take a taxi. A "good" goal can be defined by using the "good" operator $\boldsymbol{GO}$:

Definition 15 (*Good goal*). Let $\boldsymbol{G}^{\mathrm{g}}\phi$ denote the fact that ϕ is a good goal. Define:

$$\boldsymbol{G}^{\mathrm{g}}\phi \overset{\mathrm{def}}{\Longleftrightarrow} \boldsymbol{GO}\phi.$$

Thus a good goal is a situation that is preferred to its negation.

The second definition involves a satisficing goal. As noted above, satisficing—important as it is—is a procedural addendum to the definition of bounded rationality. Whereas the declarative part of bounded rationality concerns incomplete knowledge, the procedural part concerns the question of what to do when the knowledge is not complete enough [67]. So, satisficing states are, in fact, satisfactory states made accessible via search. Let $\boldsymbol{S}\phi$ stand for an arbitrary satisficing situation ϕ and relax the definition of action terms by allowing for mnemonic expressions:

$$\boldsymbol{S}\phi \Leftrightarrow \langle \textit{satisficing-search} \rangle \phi \wedge \boldsymbol{PO}\phi.$$

Note that this definition does not exclude the possibility that the search is void in cases that the satisficing solution is already at hand. Note also that the definition is using the name of search, but is not defining, or describing, the search process itself. Define a satisficing goal in terms of a satisficing state:

Definition 16 (*Satisficing goal*). Let $\boldsymbol{G}^{\mathrm{s}}\phi$ denote the fact that ϕ is a satisficing goal. Define:

$$\boldsymbol{G}^{\mathrm{s}}\phi \overset{\mathrm{def}}{\Longleftrightarrow} \boldsymbol{S}\phi.$$

As argued above, agents may try to maximize, or even optimize, if the context supports the search for extremal values. For example, in production planning, optimal

solutions are sought and implemented on a daily basis. Whether a solution is maximal or optimal depends, of course, on the structure of the preference order of an agent. If it is partial, but not total, the order may contain several maximal, incomparable elements. If, furthermore, more than one maximal element is accessible, then an optimal goal (in the intuitive sense of a best overall solution) cannot be defined. Conversely, an optimal goal can be identified if the order is total and at least one situation is accessible. By the same token, a partial order gives rise to an optimal goal if only one maximal situation is accessible. We define a "best choice" as a maximal goal and specify the conditions under which such a best choice may, in fact, be optimal.

Definition 17 (*Maximal goal*). Let $G^{\mathrm{bc}}\phi$ denote the fact that ϕ is a best choice. Define:

$$G^{\mathrm{bc}}\phi \overset{\mathrm{def}}{\Longleftrightarrow} \boldsymbol{PO}\phi \wedge \forall\chi(\chi\boldsymbol{P}\phi \rightarrow \neg\boldsymbol{A}\chi).$$

Thus, a best choice is an accessible situation to which no other accessible situation is preferred. A best choice ϕ is optimal, if ϕ is unique:

Definition 18 (*Optimal goal*). Let $G^{\mathrm{op}}\phi$ denote the fact that ϕ is optimal. Define:

$$G^{\mathrm{op}}\phi \overset{\mathrm{def}}{\Longleftrightarrow} G^{\mathrm{bc}}\phi \wedge \forall\psi((G^{\mathrm{bc}}\psi) \rightarrow (\phi \leftrightarrow \psi)).$$

Ironically, best choices need not be good nor satisficing. In a tight spot, an agent's best alternative might simply be the best among dubious alternatives.

The above definitions can be modified according to the domain. For example, a stronger notion of rationality may require that goals be consistent, so that agents will not select both ϕ and $\neg\phi$ as goals in the same situation. We did not require consistency upfront, because there are many applications of bounded rationality that do allow for contradictory goals [41,42], but consistency can be built into the goal definitions by requiring that a goal be, at least, a good goal. Because of the irreflexivity of the $\boldsymbol{P}$ operator, good goals are always contradiction-free. Define a consistent satisficing goal as follows:

Definition 19 (*Satisficing Consistent Goal*). Let $G^{\mathrm{sc}}\phi$ denote the fact that ϕ is a satisficing, consistent goal. Define:

$$G^{\mathrm{sc}}\phi \overset{\mathrm{def}}{\Longleftrightarrow} \boldsymbol{GO}\phi \wedge \boldsymbol{S}\phi.$$

A consistent best choice can be defined analogously. Optimal goals are always consistent because they are unique.

Another reasonable modification of the goal definitions is obtained by imposing the requirement of accessibility. Sometimes, goals are pursued even when the agent is uncertain whether they are accessible (setting seemingly, but not really, inaccessible goals is sometimes hailed as post-modern management style [9]). In better-understood circumstances, however, accessibility may appear as a reasonable requirement for a

goal definition. All goal definitions above can be strengthened accordingly by adding the accessibility requirement. Another useful modification is obtained by distinguishing between *maintenance* and *achievement* goals. Again, it is easy to see how to do this: add the goal situation as a conjunct to the definiens of the respective goal definition in case of a maintenance goal and add the negation in case of an achievement goal. For example, a "good" achievement goal can be defined as follows:

Definition 20 (*Good achievement goal*). Let $\boldsymbol{G}^{\mathrm{ga}}\phi$ denote the fact that ϕ is a good achievement goal. Define:

$$\boldsymbol{G}^{\mathrm{ga}}\phi \overset{\mathrm{def}}{\Longleftrightarrow} \boldsymbol{G}^{\mathrm{g}}\phi \wedge \neg\phi.$$

It might go without saying that the definition of extremal goals must always make a stipulation about accessibility; otherwise, the sky is the only limit.

5.2. Using goal definitions: an example

Although the underlying propositional language imposes obvious limitations on the present version of the logic, ALX can already serve as a knowledge-representation tool. We demonstrate this by representing Max Weber's typology of rationality which informed large parts of twentieth-century sociology [16, 52, 53]. Max Weber distinguishes between three "types" of rationality: (1) *traditional* rationality, (2) *value* rationality, (3) *goal* rationality. Traditional rationality is circumscribed as a mode of behavior along stimulus response patterns. Agents follow the tradition, rather than seeking to improve their lives. So, if a situation is a goal, then it remains a goal, regardless of the precondition. Let $\boldsymbol{G}$ stand for an arbitrary goal; then we can characterize traditional rationality by the formula

$$\boldsymbol{G}\phi \circ \chi \rightarrow \boldsymbol{G}\phi.$$

The second type of rationality, value rationality, is circumscribed as adherence to preset goals that are singled out for their intrinsic value and without regard for possible ramifications. We can express this by stipulating that all goals must be good goals and, furthermore, that the ramifications of such goals do not count and hence, should not appear in the preference order:

$$\boldsymbol{G}\phi \rightarrow \boldsymbol{G}^{\mathrm{g}}\phi \wedge \forall\psi((\phi \rightsquigarrow \psi) \wedge \neg(\phi \leftrightarrow \psi) \rightarrow \neg\boldsymbol{PO}\psi).$$

The third type of rationality is circumscribed as the unconditional search for optimal solutions (*goal rationality*). In this rationality mode all goals are, at least, best choices:

$$\boldsymbol{G}\phi \rightarrow \boldsymbol{G}^{\mathrm{bc}}\phi.$$

Weber cautions his readers repeatedly against taking his typology for a complete classification. Our formal representation immediately shows, that, in fact, the set of building blocks entering the characterizations can give rise to many alternative rationality types. Much of the confusion about Weber's typology would, in fact, go away, if one would realize this more clearly.

6. Discussion

ALX provides the skeleton of a preference-driven action logic, based on a propositional description language and three types of modal operators, a preference operator, an update operator and a set of action modalities.

6.1. Preferences

Although preference-based decision making has received some attention in the AI-literature [8, 70], ALX is actually the first preference-based action logic.

Modal preference logic was introduced by Halldén [17] and codified by von Wright [72, 73], whose preference operator satisfies irreflexivity, transitivity and the conjunction expansion principle. We modified his approach by adding normality and making preferences context-dependent. Our choice of *irreflexivity* was made for technical reasons. The machinery for reflexive preferences is more complex; also, reflexive preference statements are less intuitive (indeed, it is not easy to express reflexive preferences in natural language). *Transitivity* of preferences is widely seen as a basic requirement of rational, preference-based decision making and has not been challenged in the basic setup of bounded rationality. However, more radical applications of bounded rationality have done away with transitivity, claiming that organizational choice is often intransitive [42]. ALX can accommodate intransitive preferences since transitivity can be dropped without losing completeness and decidability [29]. This *non*-transitive logic does, in fact, allow for *in*transitive preferences, as the following example shows. Call an ALX logic without (TR) $\mathrm{ALX}^{-\mathrm{TR}}$.

Claim 21. *There exists an $ALX^{-\mathrm{TR}}$ model $M = \langle W, cw, \succ, R^a, V\rangle$ and a world $w \in W$ such that preferences are not transitive, even though the comparison relation $\succ$ is transitive. In particular, we claim that $M, w \models (p\boldsymbol{P}q) \wedge (q\boldsymbol{P}r) \wedge \neg(p\boldsymbol{P}r)$.*

Proof. Suppose that the set of primitive propositions is $\{p, q, r\}$. We define the model $M = \langle W, cw, \succ, R^a, V\rangle$ as follows:

$$W = \{w_{pqr}, w_{pq}, w_{pr}, w_{qr}, w_p, w_q, w_r, w_\emptyset\}.$$

We define cw (to the extent that we need it for the example).

$$cw(w_p, [\![p \wedge \neg q]\!]_M) = \{w_p\}.$$
$$cw(w_p, [\![\neg p \wedge q]\!]_M) = \{w_q\}.$$
$$cw(w_p, [\![q \wedge \neg r]\!]_M) = \{w_{pq}\}.$$
$$cw(w_p, [\![\neg q \wedge r]\!]_M) = \{w_{pr}\}.$$
$$cw(w_p, [\![p \wedge \neg r]\!]_M) = \{w_p\}.$$
$$cw(w_p, [\![\neg p \wedge r]\!]_M) = \{w_r\}.$$

$$\succ = \{\langle w_p, w_q\rangle, \langle w_{pq}, w_{pr}\rangle\}.$$

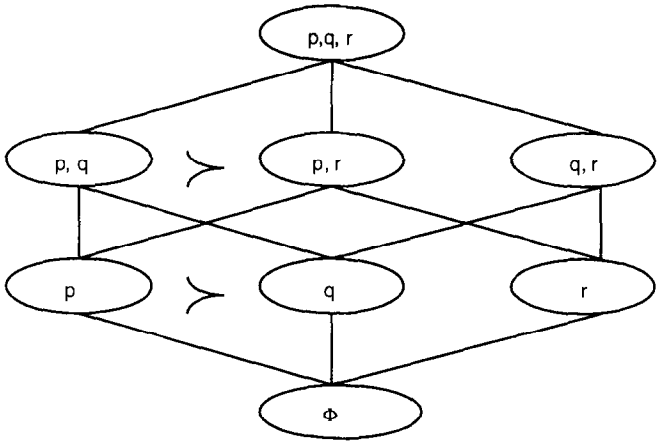


Fig. 2. Counterexample against the transitivity.

$$V(p) = \{w_{pqr}, w_{pq}, w_{pr}, w_p\}.$$
$$V(q) = \{w_{pqr}, w_{pq}, w_{qr}, w_q\}.$$
$$V(r) = \{w_{pqr}, w_{pr}, w_{qr}, w_r\}.$$

It is easy to see that the above model M is an $\mathrm{ALX}^{-\mathrm{TR}}$ model and that $M, w_p \models (p\boldsymbol{P}q) \wedge (q\boldsymbol{P}r) \wedge \neg(p\boldsymbol{P}r)$ (see also Fig. 2). □

Normality (N) was originally added to ALX in order to achieve completeness. Furthermore, (N) turns out to be instrumental in blocking counterexamples against the conjunction expansion principle.

The *conjunction expansion principle* (CEP) itself has always raised eyebrows and contributed to preference logic's unpopularity [49]. We have kept (CEP) for several reasons. First, we do not think that (CEP) restricts the logic's ability to represent preferences or preference-related notions, such as goals. At least, we cannot think of examples where it makes sense to prefer ϕ above ψ but not to prefer ϕ-and-not-ψ to ψ-and-not-ϕ. Second, we need (CEP) in the construction of context-dependent preferences. If the conjunction expansion principle is dropped from the semantics, the result is the following interpretation function of the preference operator:

$$M, w \Vdash \phi\boldsymbol{P}\psi \quad \text{iff} \quad cw(w, [\![\phi]\!]_M) \succ cw(w, [\![\psi]\!]_M),$$

and on this interpretation function, the following formula becomes valid:

$$\phi \wedge \psi \rightarrow \neg(\phi\boldsymbol{P}\psi).$$

But this formula is obviously counterintuitive. Third, ALX does not give rise to the traditional counterexamples against (CEP), because (N) blocks these examples. As we argue elsewhere [28], these examples are based on implicit partiality. For instance, assume, as in [4], that it is better that Smith *and* his wife are happy $(p \wedge q)$, than that Smith alone is happy: $(p \wedge q)\boldsymbol{P}p$. Conjunction expansion yields $(p \wedge q \wedge \neg p)\boldsymbol{P}\neg(p \wedge q) \wedge p$

and hence the preference for a contradictory state of affairs. This example comes out false in ALX because (N) assures that:

$$(\phi \wedge \psi)\boldsymbol{P}\phi \leftrightarrow \bot\boldsymbol{P}(\phi \wedge \neg\psi) \leftrightarrow \bot.$$

ALX forces the user to make the implication explicit that if Smith is happy alone, his wife is not happy: $(p \wedge q)\boldsymbol{P}(p \wedge \neg q)$. This statement entails no preference for a contradictory state of affairs; it is equivalent to its conjunction expansion. Other counterexamples to the conjunction expansion principle also exploit implicit partiality [4,21] and are blocked in the same way in ALX.[7]

ALX has a situational semantics for preference relations: the agent is supposed to hold a preference of ϕ above ψ iff she would prefer ϕ-and-not-ψ to ψ-and-not-ϕ under conditions as similar as possible to her actual situation. Obviously, situational preferences can be unstable; the agent may hold a specific preference in one situation and an opposite preference in another. Although unstable preferences play an important role in many applications of bounded rationality [3,41,42,51], stable preferences might still be handy for theoretical purposes (e.g., when using the logic to represent economic theories where stability of preferences is often assumed [10]). A stable preference relation would be one that does not change from world to world. We have discussed stable preferences elsewhere [28]; stability of the preference relation does obtain, for example, if a preference depends only on a finite (possibly empty) set of conditions that can be expressed as propositions. Stable preferences can be characterized with the following axiom:

$$\text{(UOP)} \quad (\phi\boldsymbol{P}\psi) \circ \chi \rightarrow (\phi\boldsymbol{P}\psi),$$

as shown in [28].

6.2. Minimal change and actions

Our notion of action is adopted from dynamic logic and its axioms (A1)–(A3) are not problematic. It should be clear, however, that this notion of action is, in a sense, contemplative: it answers the question of whether a particular ϕ is accessible via action a (or conversely, what would happen if the agent does a), but it does not answer the question of whether the agent will, in fact, do a.

We have used the notion of minimal change to reflect the context dependency of preference statements, but minimal change serves other purposes as well. Stalnaker introduced minimal change to modal logic in order to capture the semantics of the counterfactual conditionals that reflect causality [38,68]. Our update operator is a backward-looking dual to Stalnaker's conditional, as the "Ramsey rule" shows.

In our setup, the action operator is not bound to minimal change. Since actions entail causal effects, minimal change should appear in the semantics of the action operator [14,33,71]. This could address some nastier problems of action logics, in particular the *qualification, frame,* and *ramification* problem. Actions may require a specific context

[7] Contraposition has also been vigorously attacked with similar examples. The similarity is not surprising, since (CP) is, in fact, redundant; it follows from (CEP) and (N) as we have proved in Proposition 6.

for execution (qualification) that the action description must take into account. Linking actions to minimal change can provide an implicit qualification of the context of a specific action through the accessibility relation for that action. Using minimal change, this context is given by the actual state (that either will or will not permit action a to be executed); no additional specification of the context is required, once the accessibility relation for action a is given. The *frame of change* is given by those conditions that do not have to change as a function of a's execution. And the ramifications of an action are "automatically" captured by identifying the set of its weakest postconditions.

ALX does not put strong constraints on the closest world function. Stronger constraints might be desirable, perhaps even a full-fledged definition. We have refrained from defining the closest world function for ALX for two reasons. First, we wanted to provide a "logicians logic", i.e., a logic whose formal semantics is more than a faithful mirror of its syntax. As a consequence, we have used the standard semantic setup without strong restrictions on the definition of models and have not given a description of the properties of possible worlds. A definition of the closest world function would require such a description. Second, we are not sure about the exact meaning of the notion of "closest worlds", despite various attempts in the literature to provide a definition [14, 22, 23, 33, 36, 71]. There are two main problems: the definitions do not restrict the set of closest worlds as much as intuition seems to require (so the set contains more worlds than it should); also, the definitions do not clearly distinguish between epistemically closest worlds and causally closest worlds. This distinction is required, however, because the closest accessible world might very well be further away than the closest imaginable world, and this difference is important for an action logic.

We can illustrate this point by looking at minimal change as a consequence of an action. After all, the standard interpretation of actions is in terms of causality, hence in terms of minimal change. Unfortunately, there are several ways to conceptualize minimal change with respect to actions. We use $\langle a\rangle^{\#}\phi$ to denote the set of worlds where, by doing action a, the agent can achieve a minimally different ϕ-situation. One way to conceptualize such a change would be in terms of the closest world accessible via action a. Call this kind of "minimal change action" $\langle a\rangle^{\#1}$. The corresponding truth condition is:

$$[\![\langle a\rangle^{\#1}\phi]\!]_M = \{w : \exists w' \in W(w' \in cw(w, \{x : R^a wx\}) \text{ and } w' \in [\![\phi]\!]_M)\}.$$

According to this truth condition, $\langle a\rangle^{\#1}\phi$ first looks at the closest worlds accessible via action a and from this set picks the ϕ-worlds (see Fig. 3).

Consider, as an example, that a denotes the action of "slamming the door" and assume that *slamming* the door will cause the picture to fall off the wall (as opposed to, say, "closing the door" that will leave the picture unharmed). Assume ϕ stands for the fact that the door is shut. $\langle a\rangle^{\#1}\phi$ now looks at a world where the door is shut and the picture fell off the wall.

A second possible definition would approach the minimal change via minimally different ϕ-worlds. Call the corresponding minimal change action $\langle a\rangle^{\#2}$ (see Fig. 4). The corresponding truth condition is:

$$[\![\langle a\rangle^{\#2}\phi]\!]_M = \{w : \exists w' \in W(w' \in cw(w, [\![\phi]\!]_M) \text{ and } R^a ww')\}.$$

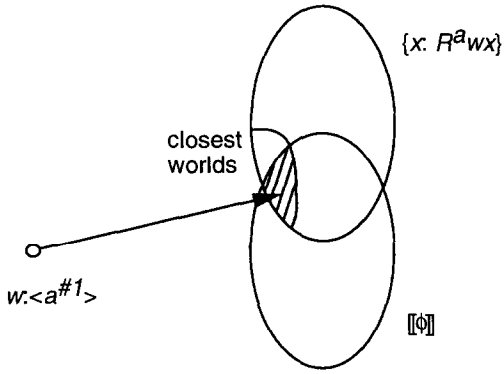


Fig. 3. Minimal change action $\langle a \rangle^{\#1}$.

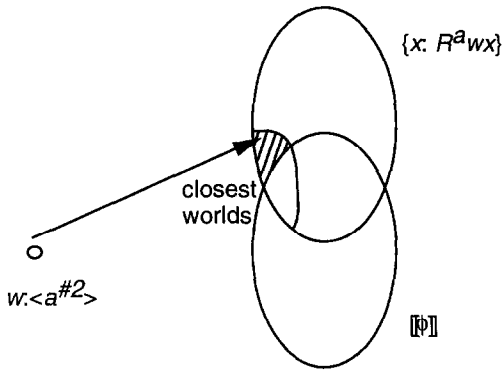


Fig. 4. Minimal change action $\langle a \rangle^{\#2}$.

Reconsider the previous example for $\langle a \rangle^{\#2}$. Slamming the door would now get us to worlds where the door is shut and the picture is back on the wall.

Since an additional action is implicit in $\langle a \rangle^{\#2}$, this kind of minimal change appears less intuitive than $\langle a \rangle^{\#1}$. But $\langle a \rangle^{\#1}$ has a drawback as well: $\langle a \rangle^{\#1}$ will give counterintuitive results if the intersection of accessible worlds and ϕ-worlds is not empty and the intersection of the closest a-accessible worlds with the ϕ-worlds is. This could happen, for example, if slamming the door would not shut the door (say, because of reverberation of the door frame). To cover this possibility we might want to look at the closest world in the intersection of a-accessible worlds and ϕ-worlds. On this view, the minimal change action is not going to return the empty set if the intersection of the accessible worlds and the closest worlds is not empty. Denote this kind of minimal change by $\langle a \rangle^{\#3}$ (see Fig. 5). The corresponding truth condition is:

$$[\![\langle a \rangle^{\#3}\phi]\!]_M = \{w : \exists w' \in W(w' \in cw(w, \{x : R^a wx\} \cap [\![\phi]\!]_M))\}.$$

But $\langle a \rangle^{\#3}$ cannot be the last word either, because it leaves undecided the question of whether the picture is on the wall or not. In sum, we should avoid a full-fledged definition of the closest world function until we can decide this—and possibly other—questions.

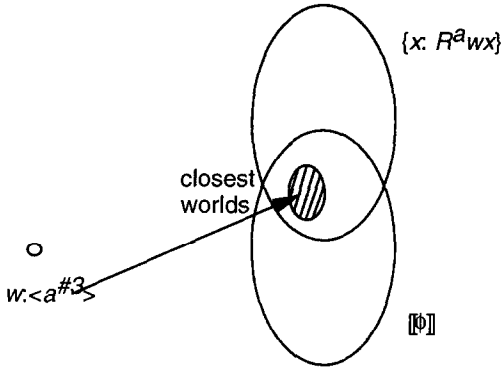


Fig. 5. Minimal change action $\langle a \rangle^{\#3}$.

6.3. Goals

Goal is an important primitive notion in other action logics [5,6,59], where the goal operators act as universal modalities. As a consequence, these logics have the necessitation rule for goals (if α is a theorem, then α is a goal) and the closure of goals under logical implication (if α is a goal and $\alpha \rightarrow \beta$ is a theorem, then β must be a goal). The necessitation rule and the deductive closure of goals have fairly severe counterintuitive implications. For example, if pain is always a consequence of having one's teeth fixed, then the pain itself becomes as a goal. Also, it does not make sense to treat tautologies as goals, as the necessitation rule would require. Much recent work in action logic has gone into systems that try to avoid these consequences by introducing an array of goal-related notions [5,6,59]. Unfortunately, these complications bring in other or additional counterintuitive effects of goals. For example, in Cohen and Levesque's logic [5,6], it is a theorem that if an agent believes that a fact holds, then the fact becomes the agent's goal. Rao and Georgeff's paper [59] avoids both necessitation and logical closure for certain epistemically qualified goals (agents need not adopt as goals what they *believe* to be *inevitably always* true and they need not to adopt ψ as a goal if they *believe* $\phi \rightarrow \psi$ to be *inevitably always* true and if they have ϕ as a goal). But in order to obtain these results, Rao and Georgeff have to make other counterintuitive assumptions. For example, they must assume that any believe-accessible world contains a goal. ALX can avoid both the necessitation rule and the deductive closure of goals by much simpler means, since we need not require monotonicity for the preference operator.

Proposition 22. (i) *Goals (as defined in this paper) are not closed under logical implication, i.e.,* $\models (\phi \rightarrow \psi)$ *does not imply* $\models (\boldsymbol{G}\phi \rightarrow \boldsymbol{G}\psi)$. (ii) *Furthermore, goals do not satisfy the necessitation rule, i.e.,* $\models \phi$ *does not imply* $\models \boldsymbol{G}\phi$.

Proof. (i) We construct a model $M = \langle W, cw, \succ, R^a, V \rangle$ for which $M \models (\phi \rightarrow \psi)$ and $M \not\models \boldsymbol{G}\phi \rightarrow \boldsymbol{G}\psi$ holds. Let $W = \{w1, w2\}$, let $cw : W \times \mathcal{P}(W) \rightarrow \mathcal{P}(W)$ be a function that satisfies (CS1)–(CSC). Let $\succ = \{\langle \{w1\}, \{w2\} \rangle\}$ and let R^a be any set. Define V

as follows: $V(p) = \{w1\}$, $V(q) = \{w1, w2\}$ and $V(r) = \{w2\}$.

It is easy to see that $[\![p]\!]_M \subset [\![q]\!]_M$, so that $M \models (p \rightarrow q)$. On the other hand, we have:

$$cw(w1, [\![p \wedge \neg r]\!]_M) = \{w1\}, \qquad cw(w1, [\![r \wedge \neg p]\!]_M) = \{w2\},$$
$$cw(w1, [\![q \wedge \neg r]\!]_M) = \{w1\}, \qquad cw(w1, [\![r \wedge \neg q]\!]_M) = \emptyset.$$

Therefore,

$$M, w1 \Vdash (p\boldsymbol{P}r) \wedge \neg(q\boldsymbol{P}r).$$

Hence,

$$M, w1 \Vdash \neg((p\boldsymbol{P}r) \rightarrow (q\boldsymbol{P}r)).$$

Thus the preference operator is not closed under logical implication. As a consequence, goals defined in terms of preferences are not closed under logical implication either.

(ii) We have shown that

$$\neg(\top \boldsymbol{P} \phi)$$

is a theorem of ALX, hence the preference operator does not satisfy the necessitation rule. □

6.4. Decision and planning

ALX is designed to represent theories about human actions, particularly theories about organizations. These theories are usually built around a decision cycle that has an agent pondering goals as a function of his problems and his action alternatives [35,54]. Planning, as understood in AI, is not a typical problem for such theories, because it is primarily a procedural problem: given a goal, find an an optimal sequence of actions and find it fast. ALX's logical properties (completeness, decidability) guarantee that an existence proof of an action sequence leading to a particular goal can always be found, provided this action sequence is feasible, i.e., a substructure of the accessibility relation R^a. This may be more than can be said of some other planners. As an example, consider the case of conjunctive planning discussed in [43].

We have a machine to buy cakes and apples; a cake costs a dollar and an apple three quarters. Due to an unfortunate design, the machine only accepts dollars and it returns a quarter when the user buys an apple; to alleviate in part this problem, the machine can change four quarters into a dollar (see Fig. 6).

One meaningful planning problem is: Assume a user has five quarters in his pocket: can he get a cake and have some change left? We can represent the domain as follows:

(1) having-one-dollar $\rightarrow$ [*buying-a-cake*]having-a-cake.

(2) having-one-dollar $\rightarrow$
[*buying-an-apple*](having-an-apple $\wedge$ having-one-quarter-left).

(3) having-four-quarters $\rightarrow$ [*change*]having-one-dollar.

(4) having-five-quarters $\rightarrow$ having-four-quarters $\wedge$ having-one-quarter-left.

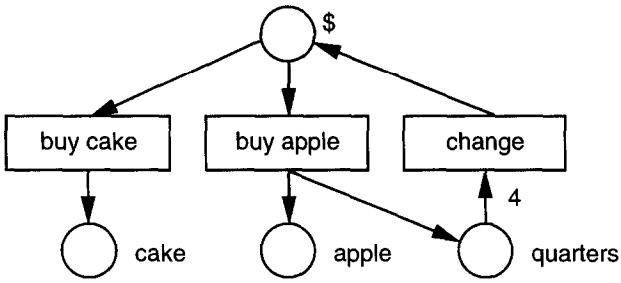


Fig. 6. A simple conjunctive planning problem.

In the next three assumptions, we exploit the fact that the universal action modality can be void:

(5) having-one-quarter-left $\rightarrow$ [*buying-a-cake*]having-one-quarter-left.
(6) having-one-quarter-left $\rightarrow$ [*buying-an-apple*]having-one-quarter-left.
(7) having-one-quarter-left $\rightarrow$ [*change*]having-one-quarter-left.
(8) having-one-quarter-left $\rightarrow$ having-some-quarter-left.

The following statement describes the specific situation of the user:

(9) having-five-quarters.

The planning problem is whether or not there exists an action sequence that makes the following state accessible:

(having-a-cake $\wedge$ having-some-change-left).

In proving this state from the premises, we generate the required action sequence: the proof is as follows:[8]

	having-five-quarters	(9)
$\Rightarrow$	having-four-quarters $\wedge$ having-one-quarter-left	(4)
$\Rightarrow$	having-one-quarter-left $\wedge$ [*change*]having-one-dollar	(3)
$\Rightarrow$	having-one-quarter-left $\wedge$ [*change*][*buying-a-cake*]having-a-cake	(1, MONA)

[8] In the proof, we use the ALX theorem $[a](\phi \wedge \psi) \leftrightarrow [a]\phi \wedge [a]\psi$, which is derivable from ALXS, since:

	$\langle a\rangle(\neg\phi \vee \neg\psi) \leftrightarrow \langle a\rangle\neg\phi \vee \langle a\rangle\neg\psi$	(A2)
$\Leftrightarrow$	$\neg\langle a\rangle(\neg\phi \vee \neg\psi) \leftrightarrow \neg(\langle a\rangle\neg\phi \vee \langle a\rangle\neg\psi)$	(Meta-reasoning)
$\Leftrightarrow$	$\neg\langle a\rangle\neg(\phi \wedge \psi) \leftrightarrow \neg\langle a\rangle\neg\phi \wedge \neg\langle a\rangle\neg\psi$	(Meta-reasoning)
$\Leftrightarrow$	$[a](\phi \wedge \psi) \leftrightarrow [a]\phi \wedge [a]\psi$	(Definition of [ ]).

$\Rightarrow$ [*change*]having-one-quarter-left $\wedge$
[*change*] [*buying-a-cake*]having-a-cake (7)
$\Rightarrow$ [*change*] [*buying-a-cake*]having-one-quarter-left $\wedge$
[*change*] [*buying-a-cake*]having-a-cake (5, MONA)
$\Rightarrow$ [*change*] [*buying-a-cake*]having-some-quarter-left $\wedge$
[change] [*buying-a-cake*]having-a-cake (8, MONA)
$\Rightarrow$ [*change*] [*buying-a-cake*]
(having-some-quarter-left $\wedge$ having-a-cake) (ALX theorem)
$\Rightarrow$ [*change*] [*buying-a-cake*]
(having-a-cake $\wedge$ having-some-quarter-left) (SUBA).

ALX has no machinery for finding plans efficiently. Also, ALX has, in its present version, no way of linking actions (or, for that matter, attitudes) to the execution of plans; its action modalities are contemplative, so there is no direct way to express that a decision or an action has, in fact, occurred. An indirect way to simulate decisions would be by defining a necessity operator[9] and saying that choosing a particular goal makes this goal necessary

$$G\phi \rightarrow \Box G\phi.$$

But this expression does not fully capture the intuitive meaning of "decision". A *do/has-done* operator would be helpful, but such an operator requires a semantic setup that includes time explicitly.

ALX could represent a planning procedure that builds an evaluation gradually as the planning process progresses, simply by conditioning preferences on states, so that certain states allow for an evaluation whereas others don't. But, again, this representation would not cover the dynamic flavor of a real planning process where "the world out there" acts as an oracle and the task is to make this oracle talk and talk fast.

6.5. Expressive power

There are still many limitations to ALX's expressive power. The most important, we think, is ALX's present restriction to a propositional description language; ALX's construction suggests a straightforward extension to first-order logic. A second important limitation is due to the absence of a belief operator. We cannot distinguish between "objectively" available knowledge and the knowledge available to an individual decision maker, so we cannot model the difference between objective and individual knowledge. For example, we cannot distinguish between disbelief in inaccessibility and accessibility, although this distinction is important when a rational agent is pondering its goals. Third, we may need time operators to represent the process of causality or to define a do-operator. For example, a time operator would be useful to express a notion of "tradition", or of expectations regarding the future. Last, but not least, we may want to extend ALX

[9] This can be done as follows: $\Box\phi \overset{\text{def}}{\Longleftrightarrow} \neg\phi \rightsquigarrow \phi$.

to multi-agent acting, by allowing for indexing modal operators with agent terms and for quantification over agents and actions.

We have tried to incorporate important elements of bounded rationality into ALX. The basic message of bounded rationality is quite simple: remember the limits of human information-processing capacity. Yet it is one thing to recognize the abstract existence of these limits and another one to find out where these limits are drawn. In the first case, one has to make sure that omniscience claims or omnipotence claims are avoided. In the second case, one has to identify which information is processed and how. No logic would be able to fully answer the second question, since it is to a large extent an empirical one. However, the question does have some general aspects that we did not address in this paper. It was not very difficult to transpose the declarative aspects of Simon's original conceptualization of bounded rationality into an action logic. However, its procedural aspects, especially search, may require a completely different semantic semantic setup. Incorporating search explicitly in the logic (rather than just naming it, as we did in the definition of satisficing states) seems to require introducing "information" as a distinct object to the logic. This, in turn, seems to require at least a partial logic; perhaps that future work may be able to exploit the progress of situation semantics in this area [57,58].

7. Conclusions and future direction

ALX is the first preference-based action logic. Its basic construction is fairly simple: let an agent have context-dependent preferences, give it action alternatives and let it deliberate about its actions on the basis of preferences and action alternatives. Furthermore, add a notion of causality that reflects the idea of minimal change. The context-dependent construction of preferences and the ability to build nested preferences gives ALX a strong expressive potential—although realizing this potential will require an extension of the logic to a multi-agent setup and to a first-order description language. Also, ALX's basic construction avoids important weaknesses of other action and preference logics. For example, goals in ALX need not be closed under logical implication, nor are tautologies automatically goals. Furthermore, the infamous conjunction expansion principle (CEP) from von Wright's logic is tamed by normality (N). ALX is complete and decidable, which prepares the ground for the development of an ALX-theorem prover. Furthermore, its pleasant logical properties would even allow for the use of ALX as a planner, albeit an inefficient one. However, ALX's primary task is knowledge representation. Preliminary experiments with a first-order version of ALX show that the combination of preference, action and update modalities allows for a flexible representation of a variety of theoretical problems in organization theory [46]. Our plan is to build a sequence of more expressive versions of ALX, starting with an extension to first-order logic as a description language, then adding multi-agent facilities, a belief operator, time operators and (if possible) a "do" operator. Work on ALX will continue.

Appendix A. Proofs

A.1. More properties of the preference operator

Proposition 6 (More properties of the preference operator). *The following formulas are theorems of ALX*:

$$
\begin{array}{ll}
\text{(CP)} & \phi \boldsymbol{P} \psi \leftrightarrow (\neg\psi)\boldsymbol{P}(\neg\phi), \\
\text{(IR)} & \neg(\phi \boldsymbol{P} \phi), \\
\text{(NT)} & \neg(\top \boldsymbol{P} \phi), \neg(\phi \boldsymbol{P} \top), \\
\text{(AS)} & \phi \boldsymbol{P} \psi \rightarrow \neg(\psi \boldsymbol{P} \phi).
\end{array}
$$

Proof.

(CP) $\phi \boldsymbol{P} \psi \leftrightarrow (\neg\psi)\boldsymbol{P}(\neg\phi)$.

$$
\begin{array}{lll}
 & \vdash \phi \boldsymbol{P} \psi & \\
\Leftrightarrow & \vdash (\phi \wedge \neg\psi)\boldsymbol{P}(\psi \wedge \neg\phi) & \text{(CEP)} \\
\Leftrightarrow & \vdash (\neg\psi \wedge \neg(\neg\phi))\boldsymbol{P}(\neg\phi \wedge (\neg(\neg\psi))) & \text{(SUBP)} \\
\Leftrightarrow & \vdash \neg\psi \boldsymbol{P} \neg\phi & \text{(CEP)}
\end{array}
$$

(IR) $\neg(\phi \boldsymbol{P} \phi)$.

$$
\begin{array}{lll}
 & \vdash (\phi \boldsymbol{P} \phi) & \\
\Rightarrow & \vdash (\phi \wedge \neg\phi)\boldsymbol{P}(\phi \wedge \neg\phi) & \text{(CEP)} \\
\Rightarrow & \vdash \bot \boldsymbol{P} \bot & \text{(Definition of } \bot) \\
\Rightarrow & \vdash \bot & \text{(N)}
\end{array}
$$

So $\vdash (\phi \boldsymbol{P} \phi) \rightarrow \bot$. Therefore, $\vdash \neg(\phi \boldsymbol{P} \phi)$.

(NT) $\neg(\top \boldsymbol{P} \phi)$, $\neg(\phi \boldsymbol{P} \top)$.

$$
\begin{array}{lll}
 & \vdash (\top \boldsymbol{P} \phi) & \\
\Rightarrow & \vdash (\neg\phi)\boldsymbol{P}(\neg\top) & \text{(CP)} \\
\Rightarrow & \vdash (\neg\phi)\boldsymbol{P}\bot & \text{(Definition of } \top) \\
\Rightarrow & \vdash \bot & \text{(N)}
\end{array}
$$

So, $\vdash \neg(\top \boldsymbol{P} \phi)$. The proof for the second half of (NT) is similar.

(AS) $\phi \boldsymbol{P} \psi \rightarrow \neg(\psi \boldsymbol{P} \phi)$.

$$
\begin{array}{lll}
 & \vdash (\phi \boldsymbol{P} \psi) \wedge (\psi \boldsymbol{P} \phi) & \\
\Rightarrow & \vdash \phi \boldsymbol{P} \phi & \text{(TR)} \\
\Rightarrow & \vdash \bot & \text{(IR)} \quad \square
\end{array}
$$

A.2. Soundness

Lemma 23 (Soundness of the update axioms). (U1)–(U6) *are valid on the class of ALX models.*

Proof.

(U1) $\phi \circ \psi \rightarrow \psi$.

$$
\begin{array}{ll}
 & M, w \Vdash \phi \circ \psi \\
\Leftrightarrow & \exists i \in [\![\phi]\!]_M (w \in cw(i, [\![\psi]\!]_M)) \quad \text{(Truth condition)} \\
\Rightarrow & \exists i \in [\![\phi]\!]_M (w \in [\![\psi]\!]_M) \quad \text{(CS1)} \\
\Rightarrow & w \in [\![\psi]\!]_M \quad \text{(Meta-reasoning)} \\
\Leftrightarrow & M, w \Vdash \psi \quad \text{(Definition of } \Vdash\text{)}
\end{array}
$$

(U2) $\phi \wedge \psi \rightarrow \phi \circ \psi$.

$$
\begin{array}{ll}
 & M, w \Vdash \phi \wedge \psi \\
\Leftrightarrow & M, w \Vdash \phi \text{ and } M, w \Vdash \psi \quad \text{(Truth condition)} \\
\Leftrightarrow & w \in [\![\phi]\!]_M \text{ and } w \in [\![\psi]\!]_M \quad \text{(Definition of } \Vdash\text{)} \\
\Rightarrow & w \in [\![\phi]\!]_M \text{ and } cw(w, [\![\psi]\!]_M) = \{w\} \quad \text{(CS2)} \\
\Rightarrow & \exists w (w \in [\![\phi]\!]_M \text{ and } w \in cw(w, [\![\psi]\!]_M)) \quad \text{(Meta-reasoning)} \\
\Leftrightarrow & M, w \Vdash \phi \circ \psi \quad \text{(Truth condition)}
\end{array}
$$

(U3) $\neg(\bot \circ \phi), \neg(\phi \circ \bot)$.

$$
\begin{array}{ll}
 & M, w \Vdash \bot \circ \phi \\
\Leftrightarrow & \exists i (i \in [\![\bot]\!]_M \text{ and } w \in cw(i, [\![\phi]\!]_M)) \quad \text{(Truth condition)} \\
\Rightarrow & \exists i (i \in [\![\bot]\!]_M) \quad \text{(Meta-reasoning)} \\
\Rightarrow & \mathit{False} \quad \text{(Meta-reasoning)}
\end{array}
$$

(U4) $(\phi \vee \psi) \circ \chi \leftrightarrow (\phi \circ \chi) \vee (\psi \circ \chi)$.

$$
\begin{array}{ll}
 & M, w \Vdash (\phi \vee \psi) \circ \chi \\
\Leftrightarrow & \exists i (i \in [\![\phi \vee \psi]\!]_M \text{ and } w \in cw(i, [\![\chi]\!]_M)) \quad \text{(Truth condition)} \\
\Leftrightarrow & \exists i ((i \in [\![\phi]\!]_M \text{ and } w \in cw(i, [\![\chi]\!]_M)) \text{ or} \\
 & \quad (i \in [\![\psi]\!] \text{ and } w \in cw(i, [\![\chi]\!]_M))) \quad \text{(Meta-reasoning)} \\
\Leftrightarrow & w \Vdash (\phi \circ \chi) \text{ or } w \Vdash (\psi \circ \chi) \quad \text{(Truth condition)} \\
\Leftrightarrow & w \Vdash (\phi \circ \chi) \vee (\psi \circ \chi) \quad \text{(Truth condition)}
\end{array}
$$

(U5) $(\phi \wedge \psi) \circ \psi \rightarrow \phi$.

$$
\begin{array}{ll}
 & M, w \Vdash (\phi \wedge \psi) \circ \psi \\
\Leftrightarrow & \exists i (i \in [\![\phi \wedge \psi]\!]_M \text{ and } w \in cw(i, [\![\psi]\!]_M)) \quad \text{(Truth condition)} \\
\Rightarrow & \exists i (i \in [\![\phi]\!]_M \text{ and } i \in [\![\psi]\!]_M \text{ and } w \in cw(i, [\![\psi]\!]_M)) \quad \text{(Truth condition)} \\
\Rightarrow & \exists i (i \in [\![\phi]\!]_M \text{ and } w = i) \quad \text{(CS2)} \\
\Rightarrow & w \in [\![\phi]\!]_M \quad \text{(Meta-reasoning)} \\
\Rightarrow & M, w \Vdash [\![\phi]\!]_M \quad \text{(Definition of } \Vdash\text{)}
\end{array}
$$

(U6) $(\phi \circ \psi) \wedge \chi \rightarrow \phi \circ (\psi \wedge \chi)$.

$$
\begin{array}{ll}
M, w \Vdash (\phi \circ \psi) \wedge \chi & \\
\Leftrightarrow \exists i(i \in [\![\phi]\!]_M \text{ and } w \in cw(i, [\![\psi]\!]_M)) \text{ and } w \in [\![\chi]\!]_M & \text{(Truth condition)} \\
\Rightarrow \exists i(i \in [\![\phi]\!]_M \text{ and } w \in cw(i, [\![\psi]\!]_M) \cap [\![\chi]\!]_M)) & \text{(Meta-reasoning)} \\
\Rightarrow \exists i(i \in [\![\phi]\!]_M \text{ and } w \in cw(i, [\![\psi \wedge \chi]\!]_M) & \text{(CS4)} \\
\Leftrightarrow M, w \Vdash \phi \circ (\psi \wedge \chi) & \text{(Truth condition)} \quad \square
\end{array}
$$

Lemma 24 (Soundness of the preference axioms). (CEP), (TR), *and* (N) *are valid on the class of ALX models.*

Proof. For any ALX model $M = \langle W, cw, \succ, R_a, V\rangle$ and any $w \in W$:

(CEP) $\phi \boldsymbol{P} \psi \leftrightarrow (\phi \wedge \neg\psi) \boldsymbol{P} (\psi \wedge \neg\phi)$.

$$
\begin{array}{ll}
M, w \Vdash \phi \boldsymbol{P} \psi & \\
\Leftrightarrow cw(w, [\![\phi \wedge \neg\psi]\!]_M) \succ cw(w, [\![\psi \wedge \neg\phi]\!]_M) & \text{(Truth condition)} \\
\Leftrightarrow cw(w, [\![(\phi \wedge \neg\psi) \wedge \neg(\psi \wedge \neg\phi)]\!]_M) \succ & \\
\quad cw(w, [\![(\psi \wedge \neg\phi) \wedge \neg(\phi \wedge \neg\psi)]\!]_M) & \text{(Propositional logic)} \\
\Leftrightarrow M, w \Vdash (\phi \wedge \neg\psi) \boldsymbol{P} (\psi \wedge \neg\phi) & \text{(Truth condition)}
\end{array}
$$

(TR) $(\phi \boldsymbol{P} \psi) \wedge (\psi \boldsymbol{P} \chi) \rightarrow (\phi \boldsymbol{P} \chi)$.

$$
\begin{array}{ll}
M, w \Vdash (\phi \boldsymbol{P} \psi) \wedge (\psi \boldsymbol{P} \chi) & \\
\Leftrightarrow cw(w, [\![\phi \wedge \neg\psi]\!]_M) \succ cw(w, [\![\psi \wedge \neg\phi]\!]_M) \text{ and} & \\
\quad cw(w, [\![\psi \wedge \neg\chi]\!]_M) \succ cw(w, [\![\chi \wedge \neg\psi]\!]_M) & \text{(Truth condition)} \\
\Rightarrow cw(w, [\![\phi \wedge \neg\chi]\!]_M) \succ cw(w, [\![\chi \wedge \neg\phi)]\!]_M) & \text{(TRAN)} \\
\Leftrightarrow M, w \Vdash (\phi \boldsymbol{P} \chi) & \text{(Truth condition)}
\end{array}
$$

(N) $\neg(\bot \boldsymbol{P} \phi), \neg(\phi \boldsymbol{P} \bot)$.

$$
\begin{array}{ll}
M, w \Vdash \bot \boldsymbol{P} \phi & \\
\Leftrightarrow cw(w, [\![\bot \wedge \neg\phi]\!]_M) \succ cw(w, [\![\phi \wedge \top]\!]_M) & \text{(Truth condition)} \\
\Rightarrow cw(w, \emptyset) \succ cw(w, [\![\phi]\!]_M) & \text{(Propositional logic)} \\
\Rightarrow \emptyset \succ cw(w, [\![\phi]\!]_M) & \text{(CS1)} \\
\Rightarrow \mathit{False} & \text{(NORM of } \succ\text{)}
\end{array}
$$

The proof about $\neg(\phi \boldsymbol{P} \bot)$ goes symmetrically. $\square$

Lemma 25 (Soundness of the action axioms). (A1)–(A3) *are valid on the class of ALX models.*

Proof. For any ALX model $M = \langle W, cw, \succ, R_a, V\rangle$ and any $w \in W$:

(A1) $\langle a\rangle\bot \leftrightarrow \bot$.

$$
\begin{array}{lll}
 & M, w \Vdash \langle a\rangle\bot & \\
\Leftrightarrow & \exists z(R^a wz \text{ and } z \in [\![\bot]\!]_M) & \text{(Truth condition)} \\
\Rightarrow & \exists z(z \in \emptyset) & \text{(Meta-reasoning)} \\
\Rightarrow & \mathit{False} & \text{(Meta-reasoning)}
\end{array}
$$

By propositional logic, $\bot \rightarrow \langle a\rangle\bot$. So $\langle a\rangle\bot \leftrightarrow \bot$.

(A2) $\langle a\rangle(\phi \vee \psi) \leftrightarrow \langle a\rangle\phi \vee \langle a\rangle\psi$.

$$
\begin{array}{lll}
 & M, w \Vdash \langle a\rangle(\phi \vee \psi) & \\
\Leftrightarrow & \exists z(R^a wz \text{ and } (z \in [\![\phi]\!]_M \text{ or } z \in [\![\psi]\!]_M)) & \text{(Truth condition)} \\
\Leftrightarrow & \exists z(R^a wz \text{ and } z \in [\![\phi]\!]_M) \text{ or } \exists z(R^a wz \text{ and } z \in [\![\psi]\!]_M) & \text{(Meta-reasoning)} \\
\Leftrightarrow & M, w \Vdash (\langle a\rangle\phi \vee \langle a\rangle\psi) & \text{(Truth condition)}
\end{array}
$$

(A3) $\langle a\rangle(\phi \wedge \psi) \rightarrow (\langle a\rangle\phi \wedge \langle a\rangle\psi)$.

$$
\begin{array}{lll}
 & M, w \Vdash \langle a\rangle(\phi \wedge \psi) & \\
\Leftrightarrow & \exists z(R^a wz \text{ and } z \in [\![\phi \wedge \psi]\!]_M) & \text{(Truth condition)} \\
\Leftrightarrow & \exists z(R^a wz \text{ and } z \in [\![\phi]\!]_M \text{ and } z \in [\![\psi]\!]_M) & \text{(Truth condition)} \\
\Rightarrow & \exists z(R^a wz \text{ and } z \in [\![\phi]\!]_M) \text{ and} & \\
 & \exists z(R^a wz \text{ and } z \in [\![\psi]\!]_M) & \text{(Meta-reasoning)} \\
\Leftrightarrow & M, w \Vdash \langle a\rangle\phi \text{ and } M, w \Vdash \langle a\rangle\psi & \text{(Truth condition)} \\
\Leftrightarrow & M, w \Vdash (\langle a\rangle\phi \wedge \langle a\rangle\psi) & \text{(Truth condition)} \quad \square
\end{array}
$$

Lemma 26 (Soundness of the inference rules). (MP), (SUBA), (SUBU), (SUBP), (NECA), (MONA) *and* (MONU) *are validity-preserving for the class of ALX models.*

Proof. For any ALX model $M = \langle W, cw, \succ, R^a, V\rangle$:

(MP) $\vdash \phi$, $\vdash (\phi \rightarrow \psi) \Rightarrow \vdash \psi$.

$$
\begin{array}{lll}
 & \phi \text{ and } (\phi \rightarrow \psi) \text{ are valid for } M & \\
\Rightarrow & \forall w \in W(M, w \Vdash \phi) \text{ and} & \\
 & \forall w \in W(M, w \Vdash (\phi \rightarrow \psi)) & \text{(Definition of validity)} \\
\Rightarrow & \forall w \in W(M, w \Vdash \phi \text{ and } M, w \Vdash (\phi \rightarrow \psi)) & \text{(Meta-reasoning)} \\
\Rightarrow & \forall w \in W(M, w \Vdash \phi \wedge (\phi \rightarrow \psi)) & \text{(Truth condition)} \\
\Rightarrow & \forall w \in W(M, w \Vdash \psi) & \text{(Definition of } \rightarrow\text{)} \\
\Rightarrow & \psi \text{ is valid for } M & \text{(Definition of validity)}
\end{array}
$$

(SUBA) $\vdash (\phi \leftrightarrow \phi') \Rightarrow \vdash (\langle a\rangle\phi \leftrightarrow \langle a\rangle\phi')$.

$\phi \leftrightarrow \phi'$ and $\langle a\rangle\phi$ are valid for M
$\Rightarrow \forall w \in W(M, w \Vdash (\phi \leftrightarrow \phi'))$ and
$\forall w \in W(M, w \Vdash \langle a\rangle\phi)$ (Definition of validity)
$\Rightarrow \forall w \in W(M, w \Vdash \phi \leftrightarrow \phi')$ and
$\forall w \in W(\exists w' \in W)(R^a ww'$ and $M, w' \Vdash \phi))$ (Truth condition)
$\Rightarrow (\forall w \in W)(\exists w' \in W)(R^a ww'$ and $M, w' \Vdash \phi')$ (Meta-reasoning)
$\Rightarrow (\forall w \in W)(M, w \Vdash \langle a\rangle\phi')$ (Truth condition)
$\Rightarrow \langle a\rangle\phi$ is valid for M (Definition of validity)

Therefore, $\vdash \phi \leftrightarrow \phi' \Rightarrow \vdash \langle a\rangle\phi \rightarrow \langle a\rangle\phi'$ is validity-preserving on a model. Symmetrically, we can show that $\vdash \phi \leftrightarrow \phi' \Rightarrow \vdash \langle a\rangle\phi' \rightarrow \langle a\rangle\phi$ is validity-preserving. So (SUBA) is sound.

The soundness of (SUMP) and (SUMU) is established in a similar fashion.

(NECA) $\vdash \phi \Rightarrow \vdash [a]\phi$.

ϕ is valid for M
$\Rightarrow (\forall w \in W)(M, w \Vdash \phi)$ (Definition of validity)
$\Rightarrow (\forall w \in W)(\forall w' \in W)(R^a ww' \Rightarrow M, w' \Vdash \phi)$ (Meta-reasoning)
$\Rightarrow (\forall w \in W)(M, w \Vdash [a]\phi)$ (Truth condition)
$\Rightarrow [a]\phi$ is valid for M. (Definition of validity)

(MONA) $\vdash \langle a\rangle\phi, \vdash (\phi \rightarrow \psi) \Rightarrow \vdash \langle a\rangle\psi$.

$\langle a\rangle\phi$ and $\phi \rightarrow \psi$ are valid for M
$\Rightarrow \forall w \in W(M, w \Vdash \langle a\rangle\phi)$ and $[\![\phi]\!]_M \subseteq [\![\psi]\!]_M$ (Definition of validity)
$\Rightarrow \forall w \in W(\exists w'(R^a ww'$ and $w' \in [\![\phi]\!])$ and
$[\![\phi]\!]_M \subseteq [\![\psi]\!]_M$ (Truth condition)
$\Rightarrow \forall w \in W(\exists w'(R^a ww'$ and $w' \in [\![\psi]\!]_M)$ (Meta-reasoning)
$\Rightarrow \forall w \in W(M, w \Vdash \langle a\rangle\psi)$ (Truth condition)
$\Rightarrow \langle a\rangle\psi$ is valid for M (Definition of validity)

(MONU) $\vdash \phi \circ \chi, \vdash (\phi \rightarrow \psi) \Rightarrow \vdash \psi \circ \chi$.

$\phi \circ \chi$ and $\phi \rightarrow \psi$ are valid for M
$\Rightarrow \forall w \in W(M, w \Vdash \phi \circ \chi)$ and $[\![\phi]\!]_M \subseteq [\![\psi]\!]_M$ (Definition of validity)
$\Rightarrow \forall w \in W(\exists w'(w' \in [\![\phi]\!]$ and
$w \in cw(w', [\![\chi]\!]_M)$ and
$[\![\phi]\!] \subseteq [\![\psi]\!]_M)$ (Truth condition)
$\Rightarrow \forall w \in W(\exists w'(w' \in [\![\psi]\!]_M$ and $w \in cw(w', [\![\chi]\!]_M))$ (Meta-reasoning)
$\Rightarrow \forall w \in W(M, w \Vdash \psi \circ \chi)$ (Truth condition)
$\Rightarrow \psi \circ \chi$ is valid for M (Definition of validity) $\square$

Proposition 7 (Soundness of ALXS). *ALXS is sound.*

Proof. Lemmas A.1–A.4 together imply it. □

A.3. More properties of the update operator

Proposition 27 (Update theorems). *The following propositions are sound for the class of ALX models:*

(U1°) $\phi \circ \phi \rightarrow \phi$.
(U2°) $(\phi \circ \psi) \circ \psi) \rightarrow \phi \circ \psi$.
(U3°) $\neg\phi \wedge (\phi \circ \psi) \rightarrow (\phi \wedge \neg\psi) \circ \psi$.
(U4°) $(\phi \circ \psi \rightarrow \bot) \rightarrow ((\phi \rightarrow \bot) \vee (\psi \rightarrow \bot))$.
(U5°) $(\phi \circ \psi) \wedge \neg\phi \rightarrow \neg\psi \circ \psi$.
(U6°) $(\neg\phi \circ \psi) \wedge (\neg\psi \circ \phi) \rightarrow (\neg\phi \circ \phi) \wedge (\neg\psi \circ \psi)$.
(U7°) $(\phi \wedge \psi) \circ \phi \leftrightarrow (\phi \wedge \psi) \circ \psi$.
(U8°) $((\rho \wedge \phi) \circ \phi) \wedge \psi \rightarrow (\rho \wedge \psi) \circ \phi$.

Proof.
(U1°) $\phi \circ \phi \rightarrow \phi$.

$$\begin{aligned}
& M, w \Vdash (\phi \circ \phi) \\
\Leftrightarrow\ & \exists i(i \in [\![\phi]\!]_M \text{ and } w \in cw(i, [\![\phi]\!]_M)) && \text{(Truth condition)} \\
\Leftrightarrow\ & w \in [\![\phi]\!]_M && \text{(CS2)} \\
\Leftrightarrow\ & M, w \Vdash \phi && \text{(Definition of } \Vdash\text{)}
\end{aligned}$$

(U2°) $(\phi \circ \psi) \circ \psi \leftrightarrow \phi \circ \psi$. First we prove ($\Rightarrow$).

$$\begin{aligned}
& M, w \Vdash (\phi \circ \psi) \circ \psi \\
\Leftrightarrow\ & \exists i(i \in [\![\phi \circ \psi]\!]_M \text{ and } w \in cw(i, [\![\psi]\!]_M)) && \text{(Truth condition)} \\
\Leftrightarrow\ & \exists i \exists j(j \in [\![\phi]\!]_M \text{ and } i \in cw(j, [\![\psi]\!]_M) \text{ and} \\
& \qquad w \in cw(i, [\![\psi]\!]_M)) && \text{(Truth condition)} \\
\Rightarrow\ & \exists i \exists j(j \in [\![\phi]\!]_M \text{ and } i \in cw(j, [\![\psi]\!]_M) \text{ and } w = i) && \text{(CS2)} \\
\Leftrightarrow\ & \exists j(j \in [\![\phi]\!]_M \text{ and } w \in cw(j, [\![\psi]\!]_M)) && \text{(Meta-reasoning)} \\
\Leftrightarrow\ & M, w \Vdash \phi \circ \psi && \text{(Truth condition)}
\end{aligned}$$

Next we prove ($\Leftarrow$).

$$\begin{aligned}
& M, w \Vdash (\phi \circ \psi) \\
\Rightarrow\ & M, w \Vdash (\phi \circ \psi) \wedge \psi && \text{(U1)} \\
\Rightarrow\ & M, w \Vdash (\phi \circ \psi) \circ \psi && \text{(U2)}
\end{aligned}$$

Therefore, $(\phi \circ \psi) \leftrightarrow (\phi \circ \psi) \circ \psi$.

(U3°) $\neg\phi \wedge (\phi \circ \psi) \rightarrow (\phi \wedge \neg\psi) \circ \psi$.

$$\begin{aligned}
& M, w \Vdash \neg\phi \wedge (\phi \circ \psi) \\
\Leftrightarrow\ & M, w \Vdash \neg\phi \text{ and } M, w \Vdash \phi \circ \psi && \text{(Truth condition)} \\
\Leftrightarrow\ & M, w \Vdash \neg\phi \text{ and } \exists i((i \in [\![\phi]\!]_M) \text{ and } w \in cw(i, [\![\psi]\!]_M)) && \text{(Truth condition)}
\end{aligned}$$

Case 1: $i \in [\![\neg\psi]\!]_M$.

$$\begin{aligned}
& i \in [\![\neg\psi]\!]_M \\
\Rightarrow\ & \exists i(i \in [\![\phi]\!]_M \text{ and } i \in [\![\neg\psi]\!]_M \text{ and } w \in cw(i, [\![\psi]\!]_M)) && \text{(Assumption)} \\
\Leftrightarrow\ & \exists i(i \in [\![\phi \wedge \neg\psi]\!]_M \text{ and } w \in cw(i, [\![\psi]\!]_M)) && \text{(Truth condition)} \\
\Leftrightarrow\ & M, w \Vdash (\phi \wedge \neg\psi) \circ \psi && \text{(Truth condition)}
\end{aligned}$$

Case 2: $i \in [\![\psi]\!]_M$.

$$\begin{aligned}
& i \in [\![\psi]\!]_M \\
\Rightarrow\ & i = w && (w \in cw(i, [\![\psi]\!]_M) \text{ and (CS2)}) \\
\Rightarrow\ & M, w \Vdash \phi \text{ and } M, w \Vdash \neg\phi && (i \in [\![\phi]\!]_M \text{ and } M, w \Vdash \neg\phi) \\
\Rightarrow\ & \mathit{False}
\end{aligned}$$

(U4°) $(\phi \circ \psi \rightarrow \bot) \rightarrow ((\phi \rightarrow \bot) \vee (\psi \rightarrow \bot))$.

$$\begin{aligned}
& (\phi \wedge \psi) \rightarrow (\phi \circ \psi) \text{ is valid} \\
\Leftrightarrow\ & \neg(\phi \circ \psi) \rightarrow \neg\phi \vee \neg\psi \text{ is valid} \\
\Leftrightarrow\ & (\neg(\phi \circ \psi) \vee \bot) \rightarrow (\neg\phi \vee \bot) \vee (\neg\psi \vee \bot) \text{ is valid} \\
\Leftrightarrow\ & (\phi \circ \psi \rightarrow \bot) \rightarrow (\phi \rightarrow \bot) \vee (\psi \rightarrow \bot) \text{ is valid}
\end{aligned}$$

(U5°) $(\phi \circ \psi) \wedge \neg\phi \rightarrow \neg\psi \circ \psi$.

$$\begin{aligned}
& M, w \Vdash (\phi \circ \psi) \wedge \neg\phi \\
\Leftrightarrow\ & \exists i(i \in [\![\phi]\!]_M \text{ and } w \in cw(i, [\![\psi]\!]_M) \text{ and } w \in [\![\neg\phi]\!]_M) && \text{(Truth condition)} \\
\Rightarrow\ & \exists i \neq w && \text{(Meta-reasoning)}
\end{aligned}$$

Suppose that $M, i \Vdash \psi$, then

$$M, i \Vdash \psi \Rightarrow cw(i, [\![\psi]\!]_M) = \{i\} \Rightarrow w = i \Rightarrow \mathit{False}.$$

Therefore, $M, i \nVdash \psi$, namely, $M, w \Vdash \neg\psi \circ \psi$.

(U6°) $(\neg\phi \circ \psi) \wedge (\neg\psi \circ \phi) \rightarrow (\neg\phi \circ \phi) \wedge (\neg\psi \circ \psi)$.

$$\begin{aligned}
& M, w \Vdash (\neg\psi \circ \phi) \wedge (\neg\phi \circ \psi) \\
\Rightarrow\ & M, w \Vdash (\neg\psi \circ \phi) \wedge \psi \wedge (\neg\phi \circ \psi) \wedge \phi) && \text{(U1)} \\
\Rightarrow\ & M, w \Vdash (\neg\psi \circ \psi) \wedge (\neg\phi \circ \phi) && \text{(U5°)}
\end{aligned}$$

(U7°) $(\phi \wedge \psi) \circ \phi \leftrightarrow (\phi \wedge \psi) \circ \psi$.

$$\begin{array}{ll} & M, w \Vdash (\phi \wedge \psi) \circ \phi \\ \Rightarrow & M, w \Vdash ((\phi \wedge \psi) \circ \phi) \wedge \phi \quad \text{(U1)} \\ \Rightarrow & M, w \Vdash (\psi \wedge \phi) \quad \text{(U5)} \\ \Rightarrow & M, w \Vdash (\psi \wedge \phi) \circ \psi \quad \text{(U2)} \end{array}$$

Therefore, $(\phi \wedge \psi) \circ \phi \rightarrow (\phi \wedge \psi) \circ \psi$. The proof about $(\phi \wedge \psi) \circ \psi \rightarrow (\phi \wedge \psi) \circ \phi$ goes analogously.

(U8°) $((\rho \wedge \phi) \circ \phi) \wedge \psi \rightarrow (\rho \wedge \psi) \circ \phi$.

$$\begin{array}{ll} & M, w \Vdash ((\rho \wedge \phi) \circ \phi) \wedge \psi \\ \Rightarrow & M, w \Vdash \rho \wedge ((\rho \wedge \phi) \circ \phi) \wedge \psi \quad \text{(U5)} \\ \Rightarrow & M, w \Vdash \rho \wedge \phi \wedge \psi \quad \text{(U1)} \\ \Rightarrow & M, w \Vdash (\rho \wedge \psi) \circ \phi \quad \text{(U2)} \end{array}$$ □

A.4. Completeness

The completeness proof for ALX proceeds along the lines of a Henkin-style construction. We give a detailed proof. First, we need a definition of consistency. We say that a formula φ is *consistent* (with respect to an axiom system) if $\neg\varphi$ is not provable (from that axiom system). A finite set $\{\varphi_1, \ldots, \varphi_k\}$ is consistent exactly if the formula $\varphi_1 \wedge \cdots \wedge \varphi_k$ is consistent. An infinite set of formulas is consistent if every finite subset of it is consistent. A set F of formulas is a *maximal consistent set* if it is consistent and any strict superset is inconsistent. With standard techniques of propositional reasoning it can be shown:

Lemma 28 (Lindenbaum's Lemma). *In any axiom system that includes all tautologies of propositional logic and the inference rule* (MP):

(1) *Any consistent set can be extended to a maximal consistent set.*
(2) *If F is a maximal consistent set, then for all formulas φ and ψ:*
  (a) *either $\varphi \in F$ or $\neg\varphi \in F$,*
  (b) *$\varphi \wedge \psi \in F$ iff $\varphi \in F$ and $\psi \in F$,*
  (c) *if $\varphi \in F$ and $\varphi \rightarrow \psi \in F$, then $\psi \in F$,*
  (d) *if $\vdash \varphi$, then $\varphi \in F$.*

The completeness of ALX means that:

(A) For arbitrary formula set Δ and arbitrary formula ϕ, $\Delta \models \phi \Rightarrow \Delta \vdash_{\text{ALX}} \phi$.

It actually turns out to be easier to show the following statement:

(B) For arbitrary formula set Δ, Δ is consistent with ALX $\Leftrightarrow$ Δ has an ALX model.

We show that (A) and (B) are equivalent:

Proof.

(B) $\Rightarrow$ (A).

$$
\begin{array}{ll}
\Delta \not\vdash_{\mathrm{ALX}} \phi & \\
\Rightarrow \Delta \cup \{\neg\phi\} \text{ is consistent with ALX} & \text{(Definition of consistency)} \\
\Rightarrow \Delta \cup \{\neg\phi\} \text{ has an ALX model} & \text{(B)} \\
\Rightarrow \exists M \in Mod(M \models \Delta \text{ and } M \models \neg\phi) & \text{(Definition of } \models) \\
\Rightarrow \exists M \in Mod(M \models \Delta \text{ and } M \not\models \phi) & \text{(Truth condition)} \\
\Rightarrow \text{It is not the case that} & \\
\quad \forall M \in Mod(M \models \Delta \Rightarrow M \models \phi) & \text{(Meta-reasoning)} \\
\Rightarrow \Delta \not\models \phi & \text{(Definition of } \models).
\end{array}
$$

(A) $\Rightarrow$ (B).

$$
\begin{array}{ll}
\Delta \text{ has no ALX model} & \\
\Rightarrow \neg(\exists M \in Mod(M \models \Delta)) & \text{(Definition of } \models) \\
\Rightarrow \forall M \in Mod(M \not\models \Delta) & \text{(Meta-reasoning)} \\
\Rightarrow \forall M \in Mod(M \not\models \Delta \text{ or } M \models \bot) & \text{(Meta-reasoning)} \\
\Rightarrow \forall M \in Mod(M \models \Delta \Rightarrow M \models \bot) & \text{(Meta-reasoning)} \\
\Rightarrow \Delta \models \bot & \text{(Definition of } \models) \\
\Rightarrow \Delta \vdash_{\mathrm{ALX}} \bot & \text{(A)} \\
\Rightarrow \Delta \text{ is inconsistent with ALX} & \text{(Definition of consistency)} \quad \square
\end{array}
$$

Assume that we can construct a canonical model M_c where the possible worlds are maximal consistent sets, then, in order to show the completeness, we have to show that for any formula ϕ,

(1) $\phi \in w \Leftrightarrow w \in [\![\phi]\!]_{M_c}$,

(2) M_c is an ALX model.

So our task is to construct a canonical model that is an ALX model. First, we need two lemmas (for the action and the update operators, respectively) that ensure the existence of certain maximal consistent sets required in the construction of the canonical model. Let W_c be the set of all maximal consistent sets built from the elements of *FML*.

Lemma 29 (Action Lemma).

$$\forall w \in W_c(\langle a\rangle\phi \in w \Rightarrow (\exists z \in W_c)(\phi \in z \textit{ and } (\forall\psi \in z)(\langle a\rangle\psi \in w))).$$

Proof. Suppose that $\langle a\rangle\phi \in w$ and let $F = \{\phi\} \cup \{\psi : \neg\langle a\rangle\neg\psi \in w\}$. Let $w^* = \{\psi : \neg\langle a\rangle\neg\psi \in w\}$.

We show first that (1) w^* and (2) F are consistent. We then show (3) that we can always extend F to an F' such that F' satisfies the condition of the lemma, i.e. $F' = z$.

(1) We claim that w^* is consistent. This is implied by:

(1.1) Assume that $\bot \in w^*$ we then show that $\bot \in w^* \rightarrow \textit{False}$.

$$
\begin{array}{ll}
\bot \in w^* & \\
\Rightarrow \neg\langle a\rangle\neg\bot \in w & \text{(Definition of } w^*) \\
\Rightarrow \neg\langle a\rangle\top \in w & \text{(Propositional logic)} \\
\Rightarrow \neg\langle a\rangle\top \wedge \langle a\rangle\phi \in w & \text{(Assumption)} \\
\Rightarrow \neg\langle a\rangle\top \wedge \langle a\rangle\top \in w & \text{(MONA)} \\
\Rightarrow \mathit{False} & \text{(Maximal consistency of } w)
\end{array}
$$

(1.2) We show that $\phi_1, \phi_2 \in w^* \Rightarrow (\phi_1 \wedge \phi_2) \in w^*$.

$$
\begin{array}{ll}
\phi_1, \phi_2 \in w^* & \\
\Rightarrow \neg\langle a\rangle\neg\phi_1 \in w \text{ and } \neg\langle a\rangle\neg\phi_2 \in w & \text{(Definition of } w^*) \\
\Rightarrow \neg(\langle a\rangle\neg\phi_1 \vee \langle a\rangle\neg\phi_2) \in w & \text{(Propositional logic)} \\
\Rightarrow \neg(\langle a\rangle(\neg\phi_1 \vee \neg\phi_2)) \in w & \text{(A2)} \\
\Rightarrow \neg(\langle a\rangle\neg(\phi_1 \wedge \phi_2)) \in w & \text{(Propositional logic)} \\
\Rightarrow \phi_1 \wedge \phi_2 \in w^* & \text{(Definition of } w^*)
\end{array}
$$

(1.3) For arbitrary ψ, we must show that $\psi \in w^*, \neg\psi \in w^* \Rightarrow \mathit{False}$.

$$
\begin{array}{ll}
\psi \in w^*, \neg\psi \in w^* & \\
\Rightarrow (\psi \wedge \neg\psi) \in w^* & (1.2) \\
\Rightarrow \bot \in w^* & \text{(Propositional logic)} \\
\Rightarrow \mathit{False} & (1.1)
\end{array}
$$

We conclude that w^* is consistent.

(2) We claim that F is consistent. This is implied by: (2.1) $\phi \rightarrow \bot \Rightarrow \mathit{False}$ and (2.2) for any $\psi \in w^*, \phi \wedge \psi \rightarrow \bot \Rightarrow \mathit{False}$.

(2.1) Assume that $\phi \rightarrow \bot$, then:

$$
\begin{array}{ll}
\phi \rightarrow \bot & \\
\Rightarrow \langle a\rangle\bot \in w & (\langle a\rangle\phi \in w \text{ and (MONA)}) \\
\Rightarrow \bot \in w & \text{(A1)} \\
\Rightarrow \mathit{False} & \text{(Maximal consistency of } w)
\end{array}
$$

(2.2) For arbitrary $\psi \in w^*$, assume that $\phi \wedge \psi \rightarrow \bot$. We show that $\phi \wedge \psi \rightarrow \bot \Rightarrow \mathit{False}$.

$$
\begin{array}{ll}
\phi \wedge \psi \rightarrow \bot & \\
\Rightarrow (\phi \rightarrow \neg\psi) & \text{(Propositional logic)} \\
\Rightarrow (\phi \rightarrow \neg\psi) \text{ and } \psi \in w^* \text{ and } \langle a\rangle\phi \in w & \text{(Assumption)} \\
\Rightarrow (\phi \rightarrow \neg\psi) \text{ and } \neg\langle a\rangle\neg\psi \in w \text{ and } \langle a\rangle\phi \in w & \text{(Definition of } w^*) \\
\Rightarrow \neg\langle a\rangle\neg\psi \in w \text{ and } \langle a\rangle\neg\psi \in w & \text{(MONA)} \\
\Rightarrow \mathit{False} & \text{(Maximal consistency of } w)
\end{array}
$$

We conclude that F is consistent.

We show now that any maximal extension F' of F satisfies the lemma. So, let F' be an arbitrary maximal consistent extension of F. We must show that: (3.1) F' exists, (3.2) $\phi \in F'$, (3.3) $\langle a \rangle \psi \notin w \Rightarrow \psi \notin F'$.

(3.1) Straightforward from Lindenbaum's Lemma.
(3.2) From the definition of F'.
(3.3) We have:

$$\begin{array}{ll} & \langle a \rangle \psi \notin w \\ \Rightarrow & \neg \langle a \rangle \psi \in w \quad \text{(Maximal consistency of } w\text{)} \\ \Rightarrow & \neg \langle a \rangle \neg\neg \psi \in w \quad \text{(Propositional logic)} \\ \Rightarrow & \neg \psi \in w^* \quad \text{(Definition of } w^*\text{)} \\ \Rightarrow & \neg \psi \in F \quad \text{(Definition of } F\text{)} \\ \Rightarrow & \neg \psi \in F' \quad \text{(Definition of } F'\text{)} \\ \Rightarrow & \psi \notin F' \quad \text{(Maximal consistency of } F'\text{).} \end{array} \qquad \square$$

The next lemma parallels the Action Lemma for the update operator.

Lemma 30 (Update Lemma).

$$\forall w \in W_c (\phi \circ \chi \in w \Rightarrow (\exists z \in W_c)(\phi \in z \textit{ and } (\forall \psi \in z)(\psi \circ \chi \in w))).$$

Proof. Suppose that $\phi \circ \chi \in w$. Let

$$F = \{\phi\} \cup \{\psi : \neg(\neg\psi \circ \chi) \in w\}, \qquad w^\circ = \{\psi : \neg(\neg\psi \circ \chi) \in w\}.$$

The proof's geometry parallels the Action Lemma. We show first that (1) w° and (2) F are consistent. We then show (3) that we can always extend F to an F' such that F' satisfies the condition of the lemma, i.e. $F' = z$.

(1) We claim that w° is consistent.

(1.1) Assume that $\bot \in w^\circ$, then we can show that $\bot \in w^\circ \Rightarrow$ *False*.

$$\begin{array}{ll} & \bot \in w^\circ \\ \Rightarrow & \neg(\neg\bot \circ \chi) \in w \quad \text{(Definition of } w^\circ\text{)} \\ \Rightarrow & \neg(\top \circ \chi) \in w \quad \text{(Propositional logic)} \\ \Rightarrow & \neg(\top \circ \chi) \wedge (\phi \circ \chi) \in w \quad \text{(Assumption)} \\ \Rightarrow & \neg(\top \circ \chi) \wedge (\top \circ \chi) \in w \quad \text{(MONU)} \\ \Rightarrow & \textit{False} \quad \text{(Maximal consistency of } w\text{)} \end{array}$$

(1.2) We show that $\phi_1, \phi_2 \in w^\circ \Rightarrow (\phi_1 \wedge \phi_2) \in w^\circ$.

$$\begin{array}{ll} & \phi_1, \phi_2 \in w^\circ \\ \Rightarrow & \neg(\neg\phi_1 \circ \chi) \in w \text{ and } \neg(\neg\phi_2 \circ \chi) \in w \quad \text{(Definition of } w^\circ\text{)} \\ \Rightarrow & \neg(\neg\phi_1 \circ \chi \vee \neg\phi_2 \circ \chi) \in w \quad \text{(Maximal consistency of } w\text{)} \\ \Rightarrow & \neg((\neg\phi_1 \vee \neg\phi_2) \circ \chi) \in w \quad \text{(U4)} \\ \Rightarrow & \neg(\neg(\phi_1 \wedge \phi_2) \circ \chi) \in w \quad \text{(Propositional logic)} \\ \Rightarrow & \phi_1 \wedge \phi_2 \in w^\circ \quad \text{(Definition of } w^\circ\text{)} \end{array}$$

(1.3) For arbitrary ψ, we must show that $\psi \in w^\circ, \neg\psi \in w^\circ \Rightarrow False$.

$$\begin{array}{ll} & \psi \in w^\circ, \neg\psi \in w^\circ \\ \Rightarrow & (\psi \wedge \neg\psi) \in w^\circ \qquad (1.2) \\ \Rightarrow & \bot \in w^\circ \qquad \text{(Propositional logic)} \\ \Rightarrow & False \qquad (1.1) \end{array}$$

We conclude that w° is consistent.

(2) We claim that F is consistent. This is implied by: (2.1) $\phi \to \bot \Rightarrow False$ and (2.2) For any $\psi \in w^\circ, \phi \wedge \psi \to \bot \Rightarrow False$.

(2.1) Assume that $\phi \to \bot$, then:

$$\begin{array}{ll} & \phi \to \bot \\ \Rightarrow & \bot \circ \chi \in w \qquad (\phi \circ \chi \in w \text{ and (MONA)}) \\ \Rightarrow & \bot \in w \qquad \text{(U3)} \\ \Rightarrow & False \qquad \text{(Maximal consistency of } w) \end{array}$$

(2.2) For arbitrary $\psi \in w^\circ$, assume that $\phi \wedge \psi \to \bot$. We show that $\phi \wedge \psi \to \bot \Rightarrow False$.

$$\begin{array}{lll} & \phi \wedge \psi \to \bot & \\ \Rightarrow & (\phi \to \neg\psi) & \text{(Propositional logic)} \\ \Rightarrow & (\phi \to \neg\psi) \text{ and } \psi \in w^\circ \text{ and } \phi \circ \chi \in w & \text{(Assumption)} \\ \Rightarrow & (\phi \to \neg\psi) \text{ and } \neg\neg\psi \circ \chi \in w \text{ and } \phi \circ \chi \in w & \text{(Definition of } w^\circ) \\ \Rightarrow & (\neg\neg\psi \circ \chi) \in w \text{ and } (\neg\psi \circ \chi) \in w & \text{(MONU)} \\ \Rightarrow & False & \text{(Maximal consistency of } w) \end{array}$$

We conclude that F is consistent. We now show that any arbitrary maximal extension F' of F satisfies the lemma. So, let F' be an arbitrary maximal consistent extensions of F. We must show that: (3.1) F' exists, (3.2) $\phi \in F'$, (3.3) $\psi \circ \chi \notin w \Rightarrow \psi \notin F'$.

(3.1) Straightforward from Lindenbaum's Lemma.
(3.2) From the definition of F'.
(3.3) We show that as follows:

$$\begin{array}{lll} & \psi \circ \chi \notin w & \\ \Rightarrow & \neg(\psi \circ \chi) \in w & \text{(Maximal consistency of } w) \\ \Rightarrow & \neg(\neg\neg\psi \circ \chi) \in w & \text{(Maximal consistency of } w) \\ \Rightarrow & \neg\psi \in w^\circ & \text{(Definition of } w^\circ) \\ \Rightarrow & \neg\psi \in F & \text{(Definition of } F) \\ \Rightarrow & \neg\psi \in F' & \text{(Definition of } F') \\ \Rightarrow & \psi \notin F' & \text{(Maximal consistency of } F'). \quad \square \end{array}$$

Proposition 8 (Completeness of ALXS). *ALXS is complete for the class of ALX models.*

Proof. We construct a canonical model $M_c = \langle W_c, cw, R^a, \succ, V\rangle$ and show that:
(1) Truth Lemma: $\chi \in w \in W_c \Leftrightarrow w \in [\![\chi]\!]_{M_c}$.
(2) M_c is an ALX model.
Define $M_c = \langle W_c, cw, R^a, \succ, V\rangle$ as follows:

$$W_c = \{i : i \text{ is a maximal consistent set}\},$$
$$w \in cw(j, \{w' : \psi \in w'\}) \quad \text{iff} \quad \forall\rho(\rho \in j \Rightarrow \rho \circ \psi \in w),$$
$$\langle w, x\rangle \in R^a \quad \text{iff} \quad \forall\rho(\rho \in x \Rightarrow \langle a\rangle\rho \in w),$$
$$cw(w, \{w' : \phi \wedge \neg\psi \in w'\}) \succ cw(w, \{w' : \neg\phi \wedge \psi \in w'\}) \quad \text{iff} \quad \phi \boldsymbol{P} \psi \in w,$$
$$V(p_i) = \{w : p_i \in w\}.$$

We prove the Truth Lemma by induction on the complexity of χ.

(1.1) $\chi \equiv p_i$.

$$\begin{aligned} & p_i \in w \\ \Leftrightarrow\ & w \in V(p_i) && \text{(Definition of } V\text{)} \\ \Leftrightarrow\ & w \in [\![p_i]\!]_{M_c} && \text{(Truth condition)} \end{aligned}$$

(1.2) $\chi \equiv \neg\phi$.

$$\begin{aligned} & \neg\phi \in w \\ \Leftrightarrow\ & \phi \notin w && \text{(Maximal consistency of } w\text{)} \\ \Leftrightarrow\ & w \notin [\![\phi]\!]_{M_c} && \text{(Induction hypothesis)} \\ \Leftrightarrow\ & w \in [\![\neg\phi]\!]_{M_c} && \text{(Truth condition)} \end{aligned}$$

(1.3) $\chi \equiv \phi \wedge \psi$.

$$\begin{aligned} & \phi \wedge \psi \in w \\ \Leftrightarrow\ & \phi, \psi \in w && \text{(Maximal consistency of } w\text{)} \\ \Leftrightarrow\ & w \in [\![\phi]\!]_{M_c} \text{ and } w \in [\![\psi]\!]_{M_c} && \text{(Induction hypothesis)} \\ \Leftrightarrow\ & w \in [\![\phi \wedge \psi]\!]_{M_c} && \text{(Truth condition)} \end{aligned}$$

(1.4) $\chi \equiv \langle a\rangle\phi$.

$$\begin{aligned} & \langle a\rangle\phi \in w \\ \Rightarrow\ & \exists z \in W_c(\phi \in z \text{ and } \forall\psi \in z(\langle a\rangle\psi \in w)) && \text{(Action Lemma)} \\ \Rightarrow\ & \exists z(\phi \in z \text{ and } R^a wz) && \text{(Definition of } R^a\text{)} \\ \Rightarrow\ & \exists z(z \in [\![\phi]\!]_{M_c} \text{ and } R^a wz) && \text{(Induction hypothesis)} \\ \Rightarrow\ & w \in [\![\langle a\rangle\phi]\!]_{M_c} && \text{(Truth condition)} \end{aligned}$$

$$\begin{aligned} & w \in [\![\langle a\rangle\phi]\!]_{M_c} \\ \Leftrightarrow\ & \exists z \in W_c(R^a wz \text{ and } z \in [\![\phi]\!]_{M_c}) && \text{(Truth condition)} \\ \Leftrightarrow\ & \exists z \in W_c(R^a wz \text{ and } \phi \in z) && \text{(Induction hypothesis)} \\ \Rightarrow\ & \langle a\rangle\phi \in w && \text{(Definition of } R^a\text{)} \end{aligned}$$

(1.5) $\chi \equiv \phi \circ \psi$.

$$
\begin{array}{ll}
\phi \circ \psi \in w & \\
\Rightarrow \exists z(\phi \in z \text{ and } (\forall \rho \in z)((\rho \circ \psi) \in w)) & \text{(Update lemma)} \\
\Rightarrow \exists z(\phi \in z \text{ and } w \in cw(z, \{w' : \psi \in w'\})) & \text{(Definition of } cw) \\
\Rightarrow \exists z(z \in [\![\phi]\!]_{M_c} \text{ and } w \in cw(z, [\![\psi]\!]_{M_c})) & \text{(Induction hypothesis)} \\
\Rightarrow w \in [\![\phi \circ \psi]\!]_{M_c} & \text{(Truth condition)}
\end{array}
$$

$$
\begin{array}{ll}
w \in [\![\phi \circ \psi]\!]_{M_c} & \\
\Leftrightarrow \exists z(z \in [\![\phi]\!]_{M_c} \text{ and } w \in cw(z, [\![\psi]\!]_{M_c})) & \text{(Truth condition)} \\
\Leftrightarrow \exists z(\phi \in z \text{ and } w \in cw(z, \{w' : \psi \in w'\})) & \text{(Induction hypothesis)} \\
\Rightarrow \phi \circ \psi \in w & \text{(Definition of } cw)
\end{array}
$$

(1.6) $\chi \equiv \phi \boldsymbol{P} \psi$.

$$
\begin{array}{ll}
\phi \boldsymbol{P} \psi \in w & \\
\Leftrightarrow cw(w, \{w' : \phi \wedge \neg\psi \in w'\}) \succ & \\
\quad cw(w, \{w' : \psi \wedge \neg\phi \in w'\}) & \text{(Definition of } \succ) \\
\Leftrightarrow cw(w, \{w' : \phi \in w'\} \cap \{w' : \neg\psi \in w'\}) \succ & \\
\quad cw(w, \{w' : \psi \in w'\} \cap \{w' : \neg\phi \in w'\}) & \text{(Meta-reasoning)} \\
\Leftrightarrow cw(w, [\![\phi]\!]_{M_c} \cap [\![\neg\psi]\!]_{M_c}) \succ & \\
\quad cw(w, [\![\psi]\!]_{M_c} \cap [\![\neg\phi]\!]_{M_c}) & \text{(Induction hypothesis)} \\
\Leftrightarrow cw(w, [\![\phi \wedge \neg\psi]\!]_{M_c}) \succ cw(w, [\![\psi \wedge \neg\phi]\!]_{M_c}) & \text{(Truth condition)} \\
\Leftrightarrow w \in [\![\phi \boldsymbol{P} \psi]\!]_{M_c} & \text{(Truth condition)}
\end{array}
$$

This concludes the proof of the Truth Lemma. We now show that M_c is an ALX model. So, we have to show that cw satisfies (CS1), (CS2), and (CSC). Moreover, we have to show that $\succ$ satisfies the transitivity and normality conditions.

(CS1) $w \in cw(j, [\![\psi]\!]_{M_c}) \Rightarrow w \in [\![\psi]\!]_{M_c}$.

$$
\begin{array}{ll}
w \in cw(j, [\![\psi]\!]_{M_c}) & \\
\Leftrightarrow \forall \rho(\rho \in j \Rightarrow \rho \circ \psi \in w) & \text{(Definition of } cw) \\
\Rightarrow \exists \rho(\rho \in j \text{ and } \rho \circ \psi \in w) & (j \text{ is not an empty set}) \\
\Rightarrow \psi \in w & \text{(U1)} \\
\Leftrightarrow w \in [\![\psi]\!]_{M_c} & \text{(Truth Lemma)}
\end{array}
$$

(CS2) $j \in [\![\psi]\!]_{M_c} \Rightarrow cw(j, [\![\psi]\!]_{M_c}) = \{j\}$. We must show that: (a) $j \in [\![\psi]\!]_{M_c} \Rightarrow j \in cw(j, [\![\psi]\!]_{M_c})$, and (b) $j \in [\![\psi]\!]_{M_c}$ and $j' \in cw(j, [\![\psi]\!]_{M_c}) \Rightarrow j = j'$.

For (a), we have:

$$
\begin{array}{ll}
j \in [\![\psi]\!]_{M_c} & \\
\Leftrightarrow \psi \in j & \text{(Truth Lemma)} \\
\Rightarrow \forall \rho(\rho \in j \Rightarrow (\rho \wedge \psi) \in j) & \text{(Maximal consistency of } j) \\
\Rightarrow \forall \rho(\rho \in j \Rightarrow (\rho \circ \psi) \in j) & \text{(U2)} \\
\Rightarrow j \in cw(j, [\![\psi]\!]_{M_c}) & \text{(Definition of } cw).
\end{array}
$$

For (b), suppose that $j \in [\![\psi]\!]_{M_c}$ and $j' \in cw(j, [\![\psi]\!]_{M_c})$, we first show that $j \subseteq j'$. Then by the maximal consistency of both j and j', we have $j = j'$. To show that $j \subseteq j'$, we proceed by reductio ad absurdum and show that $\rho \in j$ and $\rho \notin j' \Rightarrow$ *False* for arbitrary ρ.

$$
\begin{array}{ll}
\rho \in j \text{ and } \rho \notin j' & \\
\Leftrightarrow \rho \in j \text{ and } \neg\rho \in j' & \text{(Maximal consistency of } j') \\
\Rightarrow \rho \wedge \psi \in j \text{ and } \neg\rho \in j' & (j \in [\![\psi]\!] \text{ and maximal consistency of } j) \\
\Rightarrow ((\rho \wedge \psi) \circ \psi) \in j' \text{ and } \neg\rho \in j' & (j' \in cw(j, [\![\psi]\!]_{M_c}) \text{ and definition of } cw) \\
\Rightarrow ((\rho \wedge \psi) \circ \psi) \in j' \text{ and} & \\
\quad \neg((\rho \wedge \psi) \circ \psi) \in j' & \text{(U5)} \\
\Rightarrow \mathit{False} & \text{(Maximal consistency of } j')
\end{array}
$$

(CSC) $cw(w, [\![\phi]\!]_{M_c}) \cap [\![\psi]\!]_{M_c} \subseteq cw(w, [\![\phi \wedge \psi]\!]_{M_c})$. For any $j \in cw(w, [\![\phi]\!]_{M_c}) \cap [\![\psi]\!]_{M_c}$, we have to show that $j \in cw(w, [\![\phi \wedge \psi]\!]_{M_c})$. That is, for any ρ, if $\rho \in w$, then $\rho \circ (\phi \wedge \psi) \in j$ by the definition of cw.

For any ρ:

$$
\begin{array}{ll}
\rho \in w \text{ and } j \in cw(w, [\![\phi]\!]_{M_c}) \cap [\![\psi]\!]_{M_c} & \\
\Rightarrow \rho \in w \text{ and } j \in cw(w, [\![\phi]\!]_{M_c}) \text{ and } \psi \in j & \text{(Truth Lemma)} \\
\Rightarrow \rho \circ \phi \in j \text{ and } \psi \in j & \text{(Definition of } cw) \\
\Rightarrow (\rho \circ \phi) \wedge \psi \in j & \text{(Consistency of } w) \\
\Rightarrow \rho \circ (\phi \wedge \psi) \in j & \text{(U6).}
\end{array}
$$

Therefore, $j \in cw(w, [\![\phi \wedge \psi]\!]_{M_c})$ by the definition of cw, so (CSC) holds.

(NORM) ($\emptyset \not\succ X$). We must show that $\emptyset \succ X \Rightarrow$ **False**.

$$
\begin{array}{ll}
\emptyset \succ X & \\
\Rightarrow \exists w \exists \phi \exists \psi (\phi \boldsymbol{P} \psi \in w \text{ and} & \\
\qquad\qquad cw(w, [\![\phi \wedge \neg\psi]\!]_{M_c}) = \emptyset \text{ and} & \\
\qquad\qquad cw(w, [\![\psi \wedge \neg\phi]\!]_{M_c}) = X \text{ and} & \\
\qquad\qquad cw(w, [\![\phi \wedge \neg\psi]\!]_{M_c}) \succ & \\
\qquad\qquad cw(w, [\![\psi \wedge \neg\phi]\!]_{M_c})) & \text{(Definition of } \succ) \\
\Rightarrow cw(w, [\![\bot]\!]_{M_c}) \succ cw(w, [\![\psi \wedge \neg\phi]\!]_{M_c}) & (cw(w, [\![\bot]\!]_{M_c}) = \emptyset \text{ by (CS!)}) \\
\Rightarrow cw(w, [\![\bot \wedge \neg(\psi \wedge \neg\phi)]\!]_{M_c}) \succ & \\
\quad cw(w, [\![(\psi \wedge \neg\phi) \wedge \neg\bot]\!]_{M_c}) & \text{(Meta-reasoning)} \\
\Rightarrow \bot \boldsymbol{P} (\psi \wedge \neg\phi) \in w & \text{(Definition of } cw) \\
\Rightarrow \textbf{False} & \text{(N)}
\end{array}
$$

(TRAN) $cw(w, X \cap \overline{Y}) \succ cw(w, Y \cap \overline{X})$ and $cw(w, Y \cap \overline{Z}) \succ cw(w, Z \cap \overline{Y}) \Rightarrow cw(w, X \cap \overline{Z}) \succ cw(w, Z \cap \overline{X})$.

$$
\begin{aligned}
& cw(w, X \cap \overline{Y}) \succ cw(w, Y \cap \overline{X}) \text{ and} \\
& cw(w, Y \cap \overline{Z}) \succ cw(w, Z \cap \overline{Y}) \\
\Rightarrow\ & \exists\phi\exists\psi\exists\chi(\phi \boldsymbol{P} \psi \in w \text{ and } \psi \boldsymbol{P} \chi \in w \text{ and} \\
& \qquad [\![\phi]\!]_{M_c} = X \text{ and } [\![\psi]\!]_{M_c} = Y \text{ and } [\![\chi]\!]_{M_c} = Z) && \text{(Definition of } \succ) \\
\Rightarrow\ & \exists\phi\exists\psi\exists\chi(\phi \boldsymbol{P} \chi \in w \text{ and } \psi \boldsymbol{P} \chi \in w \text{ and} \\
& \qquad [\![\phi]\!]_{M_c} = X \text{ and } [\![\psi]\!]_{M_c} = Y \text{ and } [\![\chi]\!]_{M_c} = Z) && \text{(TR)} \\
\Rightarrow\ & \exists\phi\exists\psi\exists \\
& \quad \chi(cw(w, [\![\phi \wedge \neg\chi]\!]_{M_c}) \succ \chi(cw(w, [\![\chi \wedge \neg\phi]\!]_{M_c}) \text{ and} \\
& \quad [\![\phi]\!]_{M_c} = X \text{ and } [\![\psi]\!]_{M_c} = Y \text{ and } [\![\chi]\!]_{M_c} = Z) && \text{(Definition of } \succ) \\
\Rightarrow\ & cw(w, X \cap \overline{Z}) \succ cw(w, Z \cap \overline{X}) && \text{(Meta-reasoning)}
\end{aligned}
$$

This concludes the proof that M_c is an ALX model. □

A.5. *The finite model property of ALX*

Definition 31 (*Subformula set*). A formula set Φ_ρ is said to be the subformula set of ρ iff Φ_ρ satisfies the following conditions:

- $\rho \in \Phi_\rho$,
- $\neg\phi \in \Phi_\rho \Rightarrow \phi \in \Phi_\rho$,
- $\phi \wedge \psi \in \Phi_\rho \Rightarrow \phi, \psi \in \Phi_\rho$,
- $\langle a \rangle \phi \in \Phi_\rho \Rightarrow \phi \in \Phi_\rho$,
- $\phi \circ \psi \in \Phi_\rho \Rightarrow \phi, \psi \in \Phi_\rho$,
- $\phi \boldsymbol{P} \psi \in \Phi_\rho \Rightarrow \phi, \psi \in \Phi_\rho$.

Claim 32. *For any formula ρ, the subformula set of ρ is closed under subformulas.*

In ALX, the truth condition of $\phi \boldsymbol{P} \psi$ depends on the conjunction expansion principle $cw(w, [\![\phi \wedge \neg\psi]\!]_M) \succ cw(w, [\![\psi \wedge \neg\phi]\!]_M)$. Because of this, we shall need an extended subformula set to handle the problem. We define the extended subformula set accordingly.

Definition 33 (*Extended subformula set*). Let Φ be a formula set which is closed under subformulas. The extended subformula set Φ^{++} is defined as the Boolean closure of Φ. Let Φ^+ be the set of single representatives for each propositional equivalence class of the formulas in Φ^{++}. In particular, we have $\Phi \subseteq \Phi^+$.

It is easy to see that Φ^+ is finite, since Φ is finite. Moreover, for any $\phi, \psi \in \Phi^+$, there exist formulas $\chi_1, chi_2 \in \Phi^+$ such that $\vdash (\chi_1 \leftrightarrow (\phi \wedge \neg\psi))$ and $\vdash (\chi_2 \leftrightarrow (\psi \wedge \neg\phi))$. In particular, we have $\bot \in \Phi^+$.

Definition 34 (*Equivalence relation on possible worlds*). Let M be an ALX model $\langle W, \succ, cw, R^a, V \rangle$ and Φ be a formula set which is closed under subformulas. For any $w, w' \in W$, we define

$$w \approx w' \text{ with respect to } M \text{ and } \Phi^+ \text{ iff } \forall \rho \in \Phi^+ (M, w \Vdash \rho \Leftrightarrow M, w' \Vdash \rho).$$

Definition 35 (*Equivalence class*). Let $\approx$ be an equivalence relation on possible worlds with respect to M and Φ^+ and w be a possible world, we define

$$[w] \overset{\text{def}}{\Longleftrightarrow} \{w' \in W : \ w \approx w'\}.$$

Definition 36 (*Filtration*). A filtration of $M = \langle W, \succ, cw, R^a, V\rangle$ through Φ^+ is any model $M^* = \langle W^*, \succ^*, cw^*, R^{a*}, V^*\rangle$ which satisfies the following conditions:

(1) W^* is a subset of W which consists of exactly one world from each equivalence class.

(2) $R^{a*}, cw^*, \succ^*$ satisfy the following suitability conditions:

(2.1) $\forall w, w' \in W^*((\exists u \in W)(R^a wu \text{ and } w' \approx u) \Rightarrow R^{a*}ww')$,

(2.2) $\forall w, w' \in W^*$
$$(R^{a*}ww' \Rightarrow (\forall\langle a\rangle\phi \in \Phi^+)(M, w' \Vdash \phi \Rightarrow M, w \Vdash \langle a\rangle\phi)),$$

(2.3) $\forall w, w' \in W^*$
$$((\forall\psi \in \Phi^+)((\exists u \in W)\ (w \in cw(u, [\![\psi]\!]_M) \text{ and } w' \approx u) \Rightarrow w \in cw^*(w', [\![\psi]\!]_{M^*}))),$$

(2.4) $\forall w, w' \in W^*$
$$\begin{aligned}&((\forall\psi \in \Phi^+)\\ &\ (w \in cw^*(w', [\![\psi]\!]_{M^*}) \Rightarrow\\ &\ (\forall\phi \in \Phi^+)\\ &\quad ((M, w' \Vdash \phi \wedge \psi \Rightarrow M, w \Vdash (\phi \wedge \psi) \circ \psi) \text{ and}\\ &\quad (M, w' \Vdash \phi \wedge \neg\psi \Rightarrow M, w \Vdash (\phi \wedge \neg\psi) \circ \psi) \text{ and}\\ &\quad (M, w' \Vdash (\neg\phi \wedge \psi) \Rightarrow M, w \Vdash (\neg\phi \wedge \psi) \circ \psi) \text{ and}\\ &\quad (M, w' \Vdash (\neg\phi \wedge \neg\psi) \Rightarrow M, w \Vdash (\neg\phi \wedge \neg\psi) \circ \psi)))),\end{aligned}$$

(2.5) $\forall w, \in W^*(\forall\phi, \psi \in \Phi^+)$
$$(cw^*(w, [\![\phi \wedge \neg\psi]\!]_{M^*}) \succ^* cw^*(w, [\![\neg\phi \wedge \psi]\!]_{M^*}) \Leftrightarrow M, w \Vdash (\phi \boldsymbol{P} \psi)).$$

(3) $V^*(p_i) = V(p_i)$ for any $p_i \in \Phi^+$.

Theorem 37 (Filtration Theorem). *Let $M = \langle W, \succ, cw, R^a, V\rangle$ be any ALX model, Φ be any formula set which is closed under subformulas and $M^* = \langle W^*, \succ^*, cw^*, R^{a*}, V^*\rangle$ be any filtration of M through Φ^+, then for any $\chi \in \Phi^+$ and any $w \in W^*$ $(M, w \Vdash \chi \Leftrightarrow M^*, w \Vdash \chi)$.*

Proof. We prove the theorem by induction on the complexity of Φ^+. For any $\chi \in \Phi$:

(1) $\chi \equiv p_i$.

$$\begin{aligned}&M, w \Vdash p_i\\ \Leftrightarrow\ & w \in V(p_i) && \text{(Truth condition)}\\ \Leftrightarrow\ & w \in V^*(p_i) && \text{(Definition of } V^*)\\ \Leftrightarrow\ & M^*, w \Vdash p_i && \text{(Truth condition)}\end{aligned}$$

(2) $\chi \equiv \neg\phi$. We know that $\phi \in \Phi^+$,

$$
\begin{aligned}
& M, w \Vdash \neg\phi \\
\Leftrightarrow\ & M, w \nVdash \phi && \text{(Truth condition)} \\
\Leftrightarrow\ & M^*, w \nVdash \phi && \text{(Induction hypothesis)} \\
\Leftrightarrow\ & M^*, w \Vdash \neg\phi && \text{(Truth condition)}
\end{aligned}
$$

(3) $\chi \equiv \phi \wedge \psi$. We know that $\phi, \psi \in \Phi^+$,

$$
\begin{aligned}
& M, w \Vdash \phi \wedge \psi \\
\Leftrightarrow\ & M, w \Vdash \phi \text{ and } M, w \Vdash \psi && \text{(Truth condition)} \\
\Leftrightarrow\ & M^*, w \Vdash \phi \text{ and } M^*, w \Vdash \psi && \text{(Induction hypothesis)} \\
\Leftrightarrow\ & M^*, w \Vdash \phi \wedge \psi && \text{(Truth condition)}
\end{aligned}
$$

(4) $\chi \equiv \langle a \rangle \phi$. We know that $\phi \in \Phi^+$,

$$
\begin{aligned}
& M, w \Vdash \langle a \rangle \phi \\
\Rightarrow\ & \exists u \in W(R^a wu \text{ and } M, u \Vdash \phi) && \text{(Truth condition)} \\
\Rightarrow\ & \exists w' \in W^*(R^a wu \text{ and } w' \approx u \text{ and } M, u \Vdash \phi) && \text{(Definition of } W^*) \\
\Rightarrow\ & \exists w' \in W^*(R^a wu \text{ and } w' \approx u \text{ and } M, w' \Vdash \phi) && \text{(Definition of } \approx) \\
\Rightarrow\ & \exists w' \in W^*(R^{a*} ww' \text{ and } M, w' \Vdash \phi) && (2.1) \\
\Rightarrow\ & \exists w' \in W^*(R^{a*} ww' \text{ and } M^*, w' \Vdash \phi) && \text{(Induction hypothesis)} \\
\Rightarrow\ & M^*, w \Vdash \langle a \rangle \phi && \text{(Truth condition)}
\end{aligned}
$$

$$
\begin{aligned}
& M^*, w \Vdash \langle a \rangle \phi \\
\Leftrightarrow\ & \exists w' \in W^*(R^{a*} ww' \text{ and } w' \in [\![\phi]\!]_{M^*}) && \text{(Truth condition)} \\
\Leftrightarrow\ & \exists w' \in W^*(R^{a*} ww' \text{ and } w' \in [\![\phi]\!]_{M}) && \text{(Induction hypothesis)} \\
\Rightarrow\ & M, w \Vdash \langle a \rangle \phi && (2.2)
\end{aligned}
$$

(5) $\chi \equiv \phi \circ \psi$. We know that $\phi, \psi \in \Phi^+$,

$$
\begin{aligned}
& M, w \Vdash \phi \circ \psi \\
\Rightarrow\ & \exists u(M, u \Vdash \phi \text{ and } w \in cw(u, [\![\psi]\!]_M)) && \text{(Truth condition)} \\
\Rightarrow\ & \exists w' \in W^*(M, u \Vdash \phi \text{ and } w' \approx u \text{ and} \\
& w \in cw(u, [\![\psi]\!]_M)) && \text{(Definition of } W^*) \\
\Rightarrow\ & \exists w' \in W^*(M, w' \Vdash \phi \text{ and } w \in cw(u, [\![\psi]\!]_M)) && \text{(Definition of } \approx) \\
\Rightarrow\ & \exists w' \in W^*(M, w' \Vdash \phi \text{ and } w \in cw^*(w', [\![\psi]\!]_{M^*})) && (2.3) \\
\Rightarrow\ & \exists w' \in W^*(M^*, w' \Vdash \phi \text{ and } w \in cw^*(w', [\![\psi]\!]_{M^*})) && \text{(Induction hypothesis)} \\
\Rightarrow\ & M^*, w \Vdash \phi \circ \psi && \text{(Truth condition)}
\end{aligned}
$$

$$
\begin{aligned}
& M^*, w \Vdash \phi \circ \psi \\
\Leftrightarrow\ & \exists w' \in W^*(M^*, w' \Vdash \phi \text{ and } w \in cw^*(w', [\![\psi]\!]_{M^*})) && \text{(Truth condition)} \\
\Leftrightarrow\ & \exists w' \in W^*(M, w' \Vdash \phi \text{ and } w \in cw^*(w', [\![\psi]\!]_{M^*})) && \text{(Induction hypothesis)}
\end{aligned}
$$

Case 1: $M, w' \Vdash \psi$.

$$\begin{aligned}
& M, w' \Vdash \psi \text{ and } M, w' \Vdash \phi \text{ and } w \in cw^*(w', [\![\psi]\!]_{M^*})) \\
\Rightarrow\ & M, w' \Vdash (\phi \wedge \psi) \text{ and } w \in cw^*(w', [\![\psi]\!]_{M^*})) && \text{(Truth condition)} \\
\Rightarrow\ & M, w \Vdash (\phi \wedge \psi) \circ \psi && (2.4) \\
\Rightarrow\ & M, w \Vdash \phi \circ \psi && \text{(MONU)}
\end{aligned}$$

Case 2: $M, w' \Vdash \neg\psi$.

$$\begin{aligned}
& M, w' \Vdash \neg\psi \text{ and } M, w' \Vdash \phi \text{ and } w \in cw^*(w', [\![\psi]\!]_{M^*})) \\
\Rightarrow\ & M, w' \Vdash (\phi \wedge (\neg\psi)) \text{ and } w \in cw^*(w', [\![\psi]\!]_{M^*})) && \text{(Truth condition)} \\
\Rightarrow\ & M, w \Vdash (\phi \wedge (\neg\psi)) \circ \psi && (2.4) \\
\Rightarrow\ & M, w \Vdash \phi \circ \psi && \text{(MONU)}
\end{aligned}$$

(6) $\chi \equiv \phi \boldsymbol{P} \psi$. We know that $\phi, \psi \in \Phi^+$,

$$\begin{aligned}
& M, w \Vdash \phi \boldsymbol{P} \psi \\
\Leftrightarrow\ & cw^*(w, [\![\phi \wedge \neg\psi]\!]_M) \succ^* cw^*(w, [\![\psi \wedge \neg\phi]\!]_{M^*}) && (2.5) \\
\Leftrightarrow\ & M^*, w \Vdash \phi \boldsymbol{P} \psi && \text{(Truth condition)}
\end{aligned}$$

For any χ such that $\chi \in \Phi^+$ but $\chi \notin \Phi$, we know the following facts:

(7) $\chi \equiv \neg\phi$ and $\phi \in \Phi$. So $\phi \in \Phi^+$.

$$\begin{aligned}
& M, w \Vdash \neg\phi \\
\Leftrightarrow\ & M, w \nVdash \phi && \text{(Truth condition)} \\
\Leftrightarrow\ & M^*, w \nVdash \phi && \text{(Induction hypothesis)} \\
\Leftrightarrow\ & M^*, w \Vdash \neg\phi && \text{(Truth condition)}
\end{aligned}$$

(8) $\chi \equiv \phi \wedge \psi$ and $\phi, \psi \in \Phi$.

$$\begin{aligned}
& M, w \Vdash \phi \wedge \psi \\
\Leftrightarrow\ & M, w \Vdash \phi \text{ and } M, w \Vdash \psi && \text{(Truth condition)} \\
\Leftrightarrow\ & M^*, w \Vdash \phi \text{ and } M^*, w \Vdash \psi && \text{(Induction hypothesis)} \\
\Leftrightarrow\ & M^*, w \Vdash \phi \wedge \psi && \text{(Truth condition)}
\end{aligned}$$

Therefore, for any $\chi \in \Phi^+$, we have $M, w \Vdash \chi \leftrightarrow M^*, w \Vdash \chi$. □

Corollary 38 (Filtration Corollary). *Let $M = \langle W, \succ, cw, R^a, V\rangle$ be any ALX model, Φ be any formula set which is closed under subformulas and $M^* = \langle W^*, \succ^*, cw^*, R^{a*}, V^*\rangle$ be any filtration of M through Φ^+, then for any $\phi, \psi \in \Phi^+$ and $w \in W^*$:*

(a) $M, w \Vdash \neg\phi \Leftrightarrow M^*, w \Vdash \neg\phi$,
(b) $M, w \Vdash \phi \wedge \psi \Leftrightarrow M^*, w \Vdash \phi \wedge \psi$,
(c) $M, w \Vdash \phi \wedge \neg\psi \Leftrightarrow M^*, w \Vdash \phi \wedge \neg\psi$.

Proof. Straightforward. □

Theorem 39 (Invalidity Theorem). *Suppose that a formula χ is invalid in a model M, then χ is invalid in every filtration of M through Φ_χ^+.*

Proof. Since χ is invalid in a model $M = \langle W, \succ, cw, R^a, V\rangle$, there is some $w \in W$ such that $M, w \Vdash \neg\chi$. Suppose that $M^* = \langle W^*, cw^*, \succ^*, R^{a*}, V^*\rangle$ is a filtration of M through Φ_χ^+. By the definition of W^*, there is some $w^* \in W^*$ such that $w \approx w^*$ with respect to M and Φ_χ^+. Obviously, $\chi \in \Phi_\chi^+$, therefore $M, w^* \Vdash \neg\chi$. By Corollary 38(a), $M^*, w^* \Vdash \neg\chi$ and so χ is invalid in M^*. □

Theorem 10. *ALX has the finite model property.*

Proof. For arbitrary χ, suppose that $\nvdash_{\mathrm{ALX}} \chi$, then there exists a model $M = \langle W, \succ, cw, R^a, V\rangle$ and a world $w \in W$ such that $M, w \nVdash \chi$. Let Φ_χ be the subformula set of χ. We know that Φ_χ is finite. Moreover, Φ_χ^+ also is finite, by the definition of Φ_χ^+.

Now, we construct a filtration $M^* = \langle W^*, \succ^*, cw^*, R^{a*}, V^*\rangle$ of M through Φ_χ^+ as follows:

(1) For W^*, we first construct the equivalence class [ ] on W as:

$$[w] \overset{\text{def}}{\Longleftrightarrow} \{w' : \forall\rho \in \Phi_\chi^+ \forall w' \in W(M, w \Vdash \rho \Leftrightarrow M, w' \Vdash \rho)\}.$$

From each class $[w]$, we select exactly one world $w' \in [w]$ to represent this class. Now let W^* be the set of all representing worlds.

From the definition of the equivalence class [ ], we know that for any class $[w1]$ and any class $[w2]$, if $[w1] \neq [w2]$, then there exists $\rho \in \Phi_\chi^+$ such that either $M, w1 \Vdash \rho$ and $M, w2 \nVdash \rho$, or $M, w1 \nVdash \rho$ and $M, w2 \Vdash \rho$. Because Φ_χ^+ is finite, there are only finitely many formulas ρ by which we can distinguish two different classes. Therefore, there are only finitely many equivalence classes, namely, at most $2^{\mathrm{Card}(\Phi_\chi^+)}$. So W^* is finite.

(2) For V^*, we define

$$V^*(p_i) \overset{\text{def}}{\Longleftrightarrow} V(p_i) \quad \text{if } p_i \in \Phi_\chi^+.$$

(3) For R^{a*}, we define that, for any $w, w' \in W^*$,

$$\langle w, w'\rangle \in R^{a*} \quad \text{iff} \quad (\forall\langle a\rangle\phi \in \Phi_\chi^+)(M, w' \Vdash \phi \Rightarrow M, w \Vdash \langle a\rangle\phi).$$

(4) For cw^*, we define that, for any $w, w' \in W^*$ and any $\psi \in \Phi_\chi^+$,

$$\begin{aligned} &w \in cw^*(w', [\![\psi]\!]_{M^*}) \quad \text{iff} \\ &((\forall\phi \in \Phi_\chi^+)(\ (M, w' \Vdash \phi \wedge \psi \Rightarrow M, w \Vdash M, w \Vdash (\phi \wedge \psi) \circ \psi) \text{ and} \\ &\qquad (M, w' \Vdash \phi \wedge \neg\psi \Rightarrow M, w \Vdash (\phi \wedge \neg\psi) \circ \psi) \text{ and} \\ &\qquad (M, w' \Vdash (\neg\phi \wedge \psi) \Rightarrow M, w \Vdash (\neg\phi \wedge \psi) \circ \psi) \text{ and} \\ &\qquad (M, w' \Vdash (\neg\phi \wedge \neg\psi) \Rightarrow M, w \Vdash (\neg\phi \wedge \neg\psi) \circ \psi))). \end{aligned}$$

(5) For $\succ^*$, we define that, for any $w \in W^*$ and any $\phi, \psi \in \Phi_\chi^+$,

$$cw^*(w, [\![\phi \wedge \neg\psi]\!]_{M^*}) \succ^* cw^*(w, [\![\neg\phi \wedge \psi]\!]_{M^*}) \quad \text{iff} \quad M, w \Vdash \phi \boldsymbol{P} \psi.$$

Now, we have to show that M^* satisfies the conditions of a filtration of M through Φ_χ^+. From the construction of W^*, we know that W^* is a subset of W. Moreover, W^* consists of exactly one world from each equivalence class with respect to M and Φ_χ^+. Therefore, the condition for W^* is satisfied.

From the above definition of V^*, R^{a*}, cw^*, $\succ^*$, the suitability conditions (2.2), (2.4), (2.5) of Definition 36 and the condition for V^* are obviously satisfied.

To show that (2.1) is satisfied, we have to show that

$$\forall w, w' \in W^*(\exists u \in W)(w' \approx u \text{ and } R^a wu)) \Rightarrow R^{a*}ww').$$

Suppose that $\exists u \in W(w' \approx u$ and $R^a wu)$ and for any $\langle a\rangle\phi \in \Phi_\chi^+$:

$$\begin{aligned}
&\langle a\rangle\phi \in \Phi_\chi^+ \text{ and } M, w' \Vdash \phi \text{ and } R^a wu \\
\Rightarrow\ &\langle a\rangle\phi \in \Phi_\chi^+ \text{ and } M, u \Vdash \phi \text{ and } R^a wu && \text{(Definition of } \approx) \\
\Rightarrow\ &\langle a\rangle\phi \in \Phi_\chi^+ \text{ and } M, w \Vdash \langle a\rangle\phi && \text{(Truth condition)}
\end{aligned}$$

Therefore, according to the definition of R^{a*}, we have $R^{a*}ww'$, so (2.1) holds.

To show that (2.3) is satisfied, we have to show that

$$\begin{aligned}
&(\forall w, w' \in W^*)(\forall \psi \in \Phi_\chi^+) \\
&\quad((\exists u \in W)(w' \approx u \text{ and } w \in cw(u, [\![\psi]\!]_M)) \Rightarrow w \in cw^*(w', [\![\psi]\!]_{M^*})).
\end{aligned}$$

For any $w, w' \in W^*$ and any $\psi \in \Phi_\chi^+$, suppose that

$$(\exists u \in W)(w' \approx u \text{ and } w \in cw(u, [\![\psi]\!]_M))$$

and for any $\phi \in \Phi_\chi^+$:

(1) Assume that $M, w' \Vdash (\phi \wedge \psi)$, then:

$$\begin{aligned}
&M, w' \Vdash (\phi \wedge \psi) \text{ and } w' \approx u \text{ and } w \in cw(u, [\![\psi]\!]_M) \\
\Rightarrow\ &M, u \Vdash (\phi \wedge \psi) \text{ and } w \in cw(u, [\![\psi]\!]_M) && \text{(Definition of } \approx) \\
\Rightarrow\ &M, w \Vdash (\phi \wedge \psi) \circ \psi) && \text{(Truth condition)}
\end{aligned}$$

(2) Assume that $M, w' \Vdash (\phi \wedge \neg\psi)$, then:

$$\begin{aligned}
&M, w' \Vdash (\phi \wedge \neg\psi) \text{ and } w' \approx u \text{ and } w \in cw(u, [\![\psi]\!]_M) \\
\Rightarrow\ &M, u \Vdash (\phi \wedge \neg\psi) \text{ and } w \in cw(u, [\![\psi]\!]_M) && \text{(Definition of } \approx) \\
\Rightarrow\ &M, w \Vdash (\phi \wedge \neg\psi) \circ \psi) && \text{(Truth condition)}
\end{aligned}$$

(3) Assume that $M, w' \Vdash (\neg\phi \wedge \psi)$, then:

$$\begin{aligned}
&M, w' \Vdash (\neg\phi \wedge \psi) \text{ and } w' \approx u \text{ and } w \in cw(u, [\![\psi]\!]_M) \\
\Rightarrow\ &M, u \Vdash (\neg\phi \wedge \psi) \text{ and } w \in cw(u, [\![\psi]\!]_M) && \text{(Definition of } \approx) \\
\Rightarrow\ &M, w \Vdash (\neg\phi \wedge \psi) \circ \psi) && \text{(Truth condition)}
\end{aligned}$$

(4) Assume that $M, w' \Vdash (\neg\phi \wedge \neg\psi)$, then:

$$\begin{aligned}
& M, w' \Vdash (\neg\phi \wedge \neg\psi) \text{ and } w' \approx u \text{ and } w \in cw(u, [\![\psi]\!]_M) \\
\Rightarrow\ & M, u \Vdash (\neg\phi \wedge \neg\psi) \text{ and } w \in cw(u, [\![\psi]\!]_M) && \text{(Definition of } \approx) \\
\Rightarrow\ & M, w \Vdash (\neg\phi \wedge \neg\psi) \circ \psi) && \text{(Truth condition)}
\end{aligned}$$

Therefore, according to the definition of cw^* above, we have $w \in cw^*(w', [\![\psi]\!]_{M^*})$. So (2.3) holds.

We know now that M^* is indeed a filtration of M through Φ_χ^+. Moreover, we know that M^* is a finite model. By the above theorem, we know that there exists a $w \in W^*$ such that $M^*, w \nVdash \chi$. Therefore, condition (1) of the finite model property (Definition 9) is satisfied.

In order to show that condition (2) of the finite model property is also satisfied, we have to show that M^* is an ALX model. That is to say, we have to show that cw^* satisfies (CS1)–(CSC) and $\succ^*$ satisfies the transitivity and the normality.

For any $w, w' \in W^*$ and any $\psi \in \Phi_\chi^+$,

(CS1) $w \in cw^*(w', [\![\psi]\!]_{M^*}) \Rightarrow w \in [\![\psi]\!]_{M^*}$.

$$\begin{aligned}
& w \in cw^*(w', [\![\psi]\!]_{M^*}) \\
\Leftrightarrow\ & (\forall \phi \in \Phi_\chi^+) \\
& \quad ((M, w' \Vdash (\phi \wedge \psi) \Rightarrow M, w \Vdash (\phi \wedge \psi) \circ \psi) \text{ and} \\
& \quad (M, w' \Vdash (\phi \wedge \neg\psi) \Rightarrow M, w \Vdash (\phi \wedge \neg\psi) \circ \psi) \text{ and} \\
& \quad (M, w' \Vdash (\neg\phi \wedge \psi) \Rightarrow M, w \Vdash (\neg\phi \wedge \psi) \circ \psi) \text{ and} \\
& \quad (M, w' \Vdash (\neg\phi \wedge \neg\psi) \Rightarrow M, w \Vdash (\neg\phi \wedge \neg\psi) \circ \psi)) && \text{(Definition of } cw^*)
\end{aligned}$$

Case 1: $M, w' \Vdash (\phi \wedge \psi)$.

$$\begin{aligned}
& M, w' \Vdash (\phi \wedge \psi) \\
\Rightarrow\ & M, w \Vdash (\phi \wedge \psi) \circ \psi && \text{(Definition of } cw^*) \\
\Rightarrow\ & M, w \Vdash \psi && (M \text{ is an ALX model and (U1)}) \\
\Rightarrow\ & M^*, w \Vdash \psi && \text{(Filtration Theorem)} \\
\Rightarrow\ & w \in [\![\psi]\!]_{M^*} && \text{(Definition of } [\![\ ]\!]_{M^*})
\end{aligned}$$

The other cases $(\phi \wedge \neg\psi, \neg\phi \wedge \psi, \neg\phi \wedge \neg\psi)$ are proved similarly. Therefore, (CS1) is satisfied.

(CS2) $w \in [\![\psi]\!]_{M^*} \Rightarrow cw^*(w, [\![\psi]\!]_{M^*}) = \{w\}$. We must show that:

(a) $w \in [\![\psi]\!]_{M^*} \Rightarrow w \in cw^*(w, [\![\psi]\!]_{M^*})$,

(b) $w \in [\![\psi]\!]_{M^*}$ and $w' \in cw^*(w, [\![\psi]\!]_{M^*}) \Rightarrow w \equiv w'$,

where $w \equiv w'$ means that w and w' represent the same equivalence class with respect to M^* and Φ_χ^+, namely,

$$\forall \rho \in \Phi_\chi^+ (M^*, w \Vdash \rho \Leftrightarrow M^*, w \Vdash \rho).$$

For (a), we will show that $w \in [\![\psi]\!]_{M^*}$ and $w \notin cw^*(w, [\![\psi]\!]_{M^*}) \Rightarrow$ *False*.

$$
\begin{array}{ll}
& w \in [\![\psi]\!]_{M^*} \text{ and } w \notin cw^*(w, [\![\psi]\!]_{M^*}) \\
\Rightarrow & w \in [\![\psi]\!]_{M} \text{ and } w \notin cw^*(w, [\![\psi]\!]_{M^*}) \qquad \text{(Filtration Theorem)} \\
\Rightarrow & w \in [\![\psi]\!]_{M} \text{ and } (\exists \phi \in \Phi_{\chi}^{+}) \\
& \quad ((M, w \Vdash (\phi \wedge \psi) \text{ and } M, w \nVdash (\phi \wedge \psi) \circ \psi) \\
& \quad \text{or } (M, w \Vdash (\phi \wedge \neg\psi) \text{ and} \\
& \qquad M, w \nVdash (\phi \wedge \neg\psi) \circ \psi) \\
& \quad \text{or } (M, w \Vdash (\neg\phi \wedge \psi) \text{ and} \\
& \qquad M, w \nVdash (\neg\phi \wedge \psi) \circ \psi) \\
& \quad \text{or } (M, w \Vdash (\neg\phi \wedge \neg\psi) \text{ and} \\
& \qquad M, w \nVdash (\neg\phi \wedge \neg\psi) \circ \psi) \qquad \text{(Definition of } cw^*) \\
\Rightarrow & w \in [\![\psi]\!]_{M} \text{ and } (\exists \phi \in \Phi_{\chi}^{+}) \\
& \quad ((M, w \Vdash (\phi \wedge \psi) \text{ and} \qquad M, w \nVdash (\phi \wedge \psi) \circ \psi) \\
& \quad \text{or } (M, w \Vdash (\neg\psi) \text{ and } M, w \nVdash (\phi \wedge \neg\psi) \circ \psi) \\
& \quad \text{or } (M, w \Vdash \neg\phi \wedge \psi \text{ and} \\
& \qquad M, w \nVdash (\neg\phi \wedge \psi) \circ \psi) \\
& \quad \text{or } (M, w \Vdash (\neg\psi) \text{ and} \\
& \qquad M, w \nVdash (\neg\phi \wedge \neg\psi) \circ \psi) \qquad \text{(MONU)} \\
\Rightarrow & w \in [\![\psi]\!]_{M} \text{ and } (\exists \phi \in \Phi_{\chi}^{+}) \\
& \quad ((M, w \Vdash (\phi \wedge \psi) \text{ and } M, w \nVdash (\phi \wedge \psi) \circ \psi) \\
& \quad \text{or } (M, w \Vdash (\neg\phi \wedge \psi) \text{ and} \\
& \qquad M, w \nVdash (\neg\phi \wedge \psi) \circ \psi) \qquad \text{(Meta-reasoning)} \\
\Rightarrow & \exists \phi \in \Phi_{\chi}^{+} \\
& \quad ((M, w \Vdash ((\phi \wedge \psi) \wedge \psi) \text{ and} \\
& \qquad M, w \nVdash (\phi \wedge \psi) \circ \psi) \\
& \quad \text{or } (M, w \Vdash ((\neg\phi \wedge \psi) \wedge \psi) \text{ and} \qquad (M, w \Vdash \psi, \\
& \qquad M, w \nVdash (\neg\phi \wedge \psi) \circ \psi) \qquad \text{and Truth condition)} \\
\Rightarrow & \exists \phi \in \Phi_{\chi}^{+} \\
& \quad ((M, w \Vdash ((\phi \wedge \psi) \circ \psi) \text{ and} \\
& \qquad M, w \nVdash (\phi \wedge \psi) \circ \psi) \\
& \quad \text{or } (M, w \Vdash ((\neg\phi \wedge \psi) \circ \psi) \text{ and} \qquad (M \text{ is an ALX model,} \\
& \qquad M, w \nVdash (\neg\phi \wedge \psi) \circ \psi) \qquad \text{and (U2))} \\
\Rightarrow & \mathit{False}
\end{array}
$$

For (b), suppose that $w \in [\![\psi]\!]_{M^*}$ and $w' \in cw^*(w, [\![\psi]\!]_{M^*})$, for any $\phi \in \Phi_{\chi}^{+}$, we have to show that

$$M^*, w \Vdash \rho \Leftrightarrow M^*, w' \Vdash \rho.$$

($\Rightarrow$) We show that $M^*, w \Vdash \rho$ and $M^*, w' \nVdash \rho \Rightarrow$ *False*.

$$
\begin{array}{ll}
 & M^*, w \Vdash \rho \text{ and } M^*, w' \nVdash \rho \\
\Rightarrow & M, w \Vdash \rho \text{ and } M^*, w' \nVdash \rho \qquad \text{(Filtration Theorem)} \\
\Rightarrow & M, w \Vdash \rho \text{ and } M^*, w' \Vdash \neg\rho \qquad \text{(Truth condition)} \\
\Rightarrow & M, w \Vdash \rho \text{ and } M, w' \Vdash \neg\rho \qquad \text{(Filtration Corollary (a))} \\
\Rightarrow & M, w \Vdash \rho \text{ and } M, w' \Vdash \neg((\rho \wedge \psi) \circ \psi) \qquad (M \text{ is an ALX model and (U5)}) \\
\Rightarrow & M, w \Vdash (\rho \wedge \psi) \text{ and} \\
 & M, w' \Vdash \neg((\rho \wedge \psi) \circ \psi) \qquad (w \in [\![\psi]\!]_{M^*}) \\
\Rightarrow & M, w' \Vdash (\rho \wedge \psi) \circ \psi \text{ and} \\
 & M, w' \Vdash \neg((\rho \wedge \psi) \circ \psi) \qquad \text{(Definition of } cw^*) \\
\Rightarrow & \textit{False}
\end{array}
$$

($\Leftarrow$) We show that $M^*, w' \Vdash \rho$ and $M^*, w \nVdash \rho \Rightarrow$ *False*.

$$
\begin{array}{ll}
 & M^*, w' \Vdash \rho \text{ and } M^*, w \nVdash \rho \\
\Rightarrow & M, w' \Vdash \rho \text{ and } M^*, w \nVdash \rho \qquad \text{(Filtration Theorem)} \\
\Rightarrow & M, w' \Vdash \rho \text{ and } M^*, w \Vdash \neg\rho \qquad \text{(Truth condition)} \\
\Rightarrow & M, w' \Vdash \rho \text{ and } M, w \Vdash \neg\rho \qquad \text{(Filtration Corollary (a))} \\
\Rightarrow & M, w' \Vdash \rho \text{ and } M^*, w \Vdash \psi \text{ and } M, w \Vdash \neg\rho \qquad (w \in [\![\psi]\!]_{M^*}) \\
\Rightarrow & M, w' \Vdash \rho \text{ and } M, w \Vdash \psi \text{ and } M, w \Vdash \neg\rho \qquad \text{(Filtration Theorem)} \\
\Rightarrow & M, w' \Vdash \rho \text{ and } M, w \Vdash (\neg\rho \wedge \psi) \qquad \text{(Truth condition)} \\
\Rightarrow & M, w' \Vdash \rho \text{ and } M, w' \Vdash (\neg\rho \wedge \psi) \circ \psi \qquad \text{(Definition of } cw^*) \\
\Rightarrow & M, w' \Vdash \neg(\neg\rho) \text{ and } M, w' \Vdash (\neg\rho \wedge \psi) \circ \psi \qquad \text{(Meta-reasoning)} \\
\Rightarrow & M, w' \Vdash \neg(((\neg\rho) \wedge \psi) \circ \psi) \text{ and} \qquad (M \text{ is an ALX model} \\
 & M, w' \Vdash (\neg\rho \wedge \psi) \circ \psi \qquad \text{and (U5))} \\
\Rightarrow & \textit{False}
\end{array}
$$

(CSC) $j \in cw^*(w, [\![\phi]\!]_{M^*}) \cap [\![\psi]\!]_{M^*} \Rightarrow j \in cw^*(w, [\![\phi \wedge \psi]\!]_{M^*})$.

$$
\begin{array}{ll}
 & j \in cw^*(w, [\![\phi]\!]_{M^*}) \cap [\![\psi]\!]_{M^*} \\
\Leftrightarrow & (\forall \rho \in \Phi_\chi^+) \\
 & \quad ((M, w \Vdash (\rho \wedge \phi) \Rightarrow M, j \Vdash (\rho \wedge \phi) \circ \phi) \text{ and} \\
 & \quad (M, w \Vdash (\rho \wedge \neg\phi) \Rightarrow M, j \Vdash (\rho \wedge \neg\phi) \circ \phi) \text{ and} \\
 & \quad (M, w \Vdash (\neg\rho \wedge \phi) \Rightarrow M, j \Vdash (\neg\rho \wedge \phi) \circ \phi) \text{ and} \\
 & \quad (M, w \Vdash (\neg\rho \wedge \neg\phi) \Rightarrow \\
 & \qquad M, j \Vdash (\neg\rho \wedge \neg\phi) \circ \phi) \text{ and} \\
 & \quad M^*, j \Vdash \psi) \qquad \text{(Definition of } cw^*) \\
\Rightarrow & M, j \Vdash \psi \qquad \text{(Filtration Lemma)}
\end{array}
$$

For any $\rho \in \Phi_X^+$:
Case 1: $M, w \Vdash (\rho \wedge (\phi \wedge \psi))$.

$$
\begin{array}{ll}
 & M, w \Vdash (\rho \wedge (\phi \wedge \psi)) \\
\Rightarrow & M, w \Vdash (\rho \wedge \phi) \qquad \text{(Meta-reasoning)} \\
\Rightarrow & M, j \Vdash (\rho \wedge \phi) \circ \phi \qquad (j \in cw^*(w, [\![\phi]\!]_{M^*})) \\
\Rightarrow & M, j \Vdash (\rho \wedge (\phi \wedge \psi)) \circ (\phi \wedge \psi) \qquad (M, j \Vdash \psi \text{ and (U8°)})
\end{array}
$$

Case 2: $M, w \Vdash \rho \wedge \neg(\phi \wedge \psi) \Rightarrow M, w \Vdash (\rho \wedge \neg\phi) \vee (\rho \wedge \neg\psi)$.
Case 2.1:

$$
\begin{array}{ll}
 & M, w \Vdash (\rho \wedge \neg\phi) \\
\Rightarrow & M, j \Vdash (\rho \wedge \neg\phi) \circ \phi \qquad (j \in cw^*(w, [\![\phi]\!]_{M^*})) \\
\Rightarrow & M, j \Vdash (\rho \wedge \neg\phi) \circ (\phi \wedge \psi) \qquad (M, j \Vdash \psi \text{ and (U6)}) \\
\Rightarrow & M, j \Vdash (\rho \wedge \neg\phi \vee \rho \wedge \neg\psi) \circ (\phi \wedge \psi) \qquad \text{(MONU)} \\
\Rightarrow & M, j \Vdash (\rho \wedge \neg(\phi \wedge \psi)) \circ (\phi \wedge \psi) \qquad \text{(Meta-reasoning)}
\end{array}
$$

Case 2.2: $M, w \Vdash \rho \wedge \neg\psi$.
Case 2.2.1:

$$
\begin{array}{ll}
 & M, w \Vdash \phi \\
\Rightarrow & M^*, w \Vdash \phi \qquad \text{(Filtration Lemma)} \\
\Rightarrow & \{w\} = cw^*(w, [\![\phi]\!]_{M^*}) \qquad \text{(CS2)} \\
\Rightarrow & w = j \qquad (j \in cw^*(w, [\![\phi]\!]_{M^*})) \\
\Rightarrow & M, j \Vdash \neg\psi \qquad (M, w \Vdash \neg\psi) \\
\Rightarrow & \mathit{False} \qquad (M, j \Vdash \psi)
\end{array}
$$

Case 2.2.2:

$$
\begin{array}{ll}
 & M, w \Vdash \neg\phi \\
\Rightarrow & M, w \Vdash (\rho \wedge \neg\phi) \qquad \text{(Meta-reasoning)} \\
\Rightarrow & M, j \Vdash (\rho \wedge \neg\phi) \circ \phi \qquad (j \in cw^*(w, [\![\phi]\!]_{M^*})) \\
\Rightarrow & M, j \Vdash (\rho \wedge \neg\phi) \circ (\phi \wedge \psi) \qquad \text{(U6)} \\
\Rightarrow & M, j \Vdash (\rho \wedge \neg\phi \vee \rho \wedge \neg\psi) \circ (\phi \wedge \psi) \qquad \text{(MONU)} \\
\Rightarrow & M, j \Vdash (\rho \wedge \neg(\phi \wedge \psi)) \circ (\phi \wedge \psi) \qquad \text{(Meta-reasoning)}
\end{array}
$$

Case 3:

$$
\begin{array}{ll}
 & M, w \Vdash \neg\rho \wedge (\phi \wedge \psi) \\
\Rightarrow & M, w \Vdash \neg\rho \wedge \phi \qquad \text{(Meta-reasoning)} \\
\Rightarrow & M, j \Vdash (\neg\rho \wedge \phi) \circ \phi \qquad (j \in cw^*(w, [\![\phi]\!]_{M^*})) \\
\Rightarrow & M, j \Vdash (\neg\rho \wedge (\phi \wedge \psi)) \circ \phi \qquad (M, j \Vdash \psi \text{ and (U8°)}) \\
\Rightarrow & M, j \Vdash (\neg\rho \wedge (\phi \wedge \psi)) \circ (\phi \wedge \psi) \qquad \text{(U4)}
\end{array}
$$

Case 4: $M, w \Vdash (\neg\rho \wedge \neg(\phi \wedge \psi)) \Rightarrow M, w \Vdash (\neg\rho \wedge \neg\phi) \vee (\rho \wedge \neg\psi)$.
Case 4.1:

$$
\begin{array}{ll}
M, w \Vdash (\neg\rho \wedge \neg\phi) & \\
\Rightarrow M, j \Vdash (\neg\rho \wedge \neg\phi) \circ \phi & (j \in cw^*(w, [\![\phi]\!]_{M^*})) \\
\Rightarrow M, j \Vdash (\neg\rho \wedge \neg\phi) \circ (\phi \wedge \psi) & (M, j \Vdash \psi \text{ and (U6)}) \\
\Rightarrow M, j \Vdash (\neg\rho \wedge \neg\phi \vee \rho \wedge \neg\psi) \circ (\phi \wedge \psi) & (\text{MONU}) \\
\Rightarrow M, j \Vdash (\neg\rho \wedge \neg(\phi \wedge \psi)) \circ (\phi \wedge \psi) & (\text{Meta-reasoning})
\end{array}
$$

Case 4.2: $M, w \Vdash \neg\rho \wedge \neg\psi$.
Case 4.2.1:

$$
\begin{array}{ll}
M, w \Vdash \phi & \\
\Rightarrow M^*, w \Vdash \phi & (\text{Filtration Lemma}) \\
\Rightarrow \{w\} = cw^*(w, [\![\phi]\!]_{M^*}) & (\text{CS2}) \\
\Rightarrow w = j & (j \in cw^*(w, [\![\phi]\!]_{M^*})) \\
\Rightarrow M, j \Vdash \neg\psi & (M, w \Vdash \neg\psi) \\
\Rightarrow \mathit{False} & (M, j \Vdash \psi)
\end{array}
$$

Case 4.2.2:

$$
\begin{array}{ll}
M, w \Vdash \neg\phi & \\
\Rightarrow M, w \Vdash (\neg\rho \wedge \neg\phi) & (\text{Meta-reasoning}) \\
\Rightarrow M, j \Vdash (\neg\rho \wedge \neg\phi) \circ \phi & (j \in cw^*(w, [\![\phi]\!]_{M^*})) \\
\Rightarrow M, j \Vdash (\neg\rho \wedge \neg\phi) \circ (\phi \wedge \psi) & (\text{U6}) \\
\Rightarrow M, j \Vdash (\neg\rho \wedge \neg\phi \vee \rho \wedge \neg\psi) \circ (\phi \wedge \psi) & (\text{MONU}) \\
\Rightarrow M, j \Vdash (\neg\rho \wedge \neg(\phi \wedge \psi)) \circ (\phi \wedge \psi) & (\text{Meta-reasoning})
\end{array}
$$

Therefore, by the results of Cases 1–4 and the definition of cw^*, we have that $j \in cw^*(w, [\![\phi \wedge \psi]\!]_{M^*})$. So (CSC) is satisfied.

(NORM) ($\emptyset \not\succ^* X$). We must show that $\emptyset \succ^* X \Rightarrow$ **False**.

$$
\begin{array}{ll}
\emptyset \succ^* X & \\
\Rightarrow (\exists \phi, \psi \in \Phi_X^+) & \\
\quad (M, w \Vdash \phi \boldsymbol{P} \psi \text{ and} & \\
\quad\quad cw^*(w, [\![\phi \wedge \neg\psi]\!]_{M^*}) = \emptyset \text{ and} & \\
\quad\quad cw^*(w, [\![\psi \wedge \neg\phi]\!]_{M^*}) = X \text{ and} & \\
\quad\quad cw^*(w, [\![\phi \wedge \neg\psi]\!]_{M^*}) \succ^* & \\
\quad\quad cw^*(w, [\![\psi \wedge \neg\phi]\!]_{M^*})) & (\text{Definition of } \succ) \\
\Rightarrow cw^*(w, [\![\bot]\!]_{M^*}) \succ^* & \\
\quad cw^*(w, [\![\psi \wedge \neg\phi]\!]_{M^*}) & (cw^*(w, [\![\bot]\!]_{M^*}) = \emptyset \text{ by (CS1)})
\end{array}
$$

$$
\begin{array}{ll}
\Rightarrow cw^*(w, [\![\bot]\!]_{M^*}) \succ^* & \\
\quad cw^*(w, [\![\psi \wedge \neg\phi]\!]_{M^*}) \text{ and} & \\
\quad \bot \in \Phi_X^+ \text{ and} & \\
\quad \exists\rho \in \Phi_X^+([\![\rho]\!]_{M^*} = [\![\psi \wedge \neg\phi]\!]_{M^*}) & \text{(Definition of } \Phi_X^+) \\
\Rightarrow cw^*(w, [\![\bot \wedge \neg\rho]\!]_{M^*}) \succ^* & \\
\quad cw^*(w, [\![\rho \wedge \neg\bot]\!]_{M^*}) & \text{(Meta-reasoning)} \\
\Rightarrow M, w \Vdash \bot \boldsymbol{P} \rho & \text{(Definition of } cw^*) \\
\Rightarrow \textbf{False} & \text{(N)}
\end{array}
$$

The proof for the second part of (NORM), $(X \not\succ^* \emptyset)$, is similar.

(TRAN) $cw^*(w, X \cap \overline{Y}) \succ^* cw^*(w, Y \cap \overline{X})$ and $cw^*(w, Y \cap \overline{Z}) \succ^* cw^*(w, Z \cap \overline{Y}) \Rightarrow cw^*(w, X \cap \overline{Z}) \succ^* cw^*(w, Z \cap \overline{X})$.

$$
\begin{array}{ll}
\quad cw^*(w, X \cap \overline{Y}) \succ^* cw^*(w, Y \cap \overline{X}) \text{ and} & \\
\quad cw^*(w, Y \cap \overline{Z}) \succ^* cw^*(w, Z \cap \overline{Y}) & \\
\Rightarrow \exists\phi\exists\psi\exists\rho(X = [\![\phi]\!]_{M*} \text{ and } Y = [\![\psi]\!]_{M*} \text{ and} & \\
\qquad\qquad Z = [\![\rho]\!]_{M*} \text{ and } M, w \Vdash (\phi \boldsymbol{P} \psi) \text{ and} & \\
\qquad\qquad M, w \Vdash (\psi \boldsymbol{P} \rho) \text{ and } (\phi, \psi, \rho \in \Phi_X^+)) & \text{(Definition of } cw^*) \\
\Rightarrow M, w \Vdash \phi \boldsymbol{P} \rho \text{ and } (\phi, \rho \in \Phi_X^+) & \text{(TR)} \\
\Rightarrow cw^*(w, [\![\phi \wedge \neg\rho]\!]_{M^*}) \succ^* cw^*(w, [\![\rho \wedge \neg\phi]\!]_{M^*}) & \text{(Definition of } \succ^*) \\
\Rightarrow cw^*(w, X \cap \overline{Z}) \succ^* cw^*(w, Z \cap \overline{X}) & \text{(Definitions of } X, Y, Z)
\end{array}
$$

As a consequence, M^* is an ALX model. Because of the soundness of ALX logic, we know that for any ρ, $\vdash_{\text{ALX}} \rho \Rightarrow \forall w(M^*, w \Vdash \rho)$. This means that ALX also satisfies condition (2) of the finite model property (Definition 9). So ALX has the finite model property. □

Acknowledgement

The authors gratefully acknowledge very helpful suggestions from Maarten Marx, Peter van Emde Boas, Scip Garling, Jaap Kamps, Jean-Jules Meyer and Yao-Hua Tan. This research was supported by a PIONIER grant from the Dutch National Science Foundation (# PGS 50-334).

References

[1] H. Blumer, *Symbolic Interactionism: Perspective and Methods* (Englewood Cliffs, Prentice-Hall, NJ, 1969).

[2] A.L. Brown, S. Mantha and T. Wakayama, Preferences as normative knowledge: towards declarative obligations, in: J.-J.C. Meyer and R.J. Wieringa, eds., *Proceedings DEON'91*, Amsterdam (1991) 142–163.

[3] K. Carley, Efficiency in a garbage can: implications for crisis management, in: J.G. March and R. Weissinger-Baylon, eds., *Ambiguity and Command* (Pitman, Marshfield, MA, 1986) 165–194.
[4] R. Chisholm and E. Sosa, On the logic of "intrinsically better", *Amer. Philos. Q.* **3** (1966) 244–249.
[5] P.R. Cohen and H.J. Levesque, Persistence, intention and commitment, in: M.P. Georgeff and A.L. Lansky, eds., *Proceedings 1986 Workshop on Reasoning about Actions and Plans* (Morgan Kaufmann, San Mateo, CA, 1987) 297–340.
[6] P.R. Cohen and H.J. Levesque, Intention is choice with commitment, *Artif. Intell.* **42** (1990) 213–261.
[7] S. Danielsson, *Preference and Obligation* (Filosofischa föreningen, Uppsala, 1968).
[8] T. Dean and M.P. Wellman, On the value of goals, in: *Proceedings Rochester Planning Workshop*, Rochester, NY (1989).
[9] P.F. Drucker, The theory of business, *Harvard Business Rev.* (Sept.-Oct. 1994) 95–107.
[10] S. French, *Decision Theory, an Introduction to the Mathematics of Rationality* (Ellis Horwood, Chichester, England, 1988).
[11] L.T.F. Gamut, *Logic, Language, and Meaning* (The University of Chicago Press, Chicago, IL, 1990).
[12] A. Giddens, *Central Problems in Social Theory: Action, Structures, and Contradiction in Social Analysis* (University of California Press, Berkeley, CA, 1979).
[13] M.L. Ginsberg, Counterfactuals, *Artif. Intell.* **30** (1986) 35–79.
[14] M.L. Ginsberg and D.E. Smith, Reasoning about action I: a possible worlds approach, *Artif. Intell.* **35** (1988) 165–195; also in: M. Ginsberg, ed., *Readings in Nonmonotonic Reasoning* (Morgan Kaufmann, Los Altos, CA, 1987) 433–463.
[15] G. Grahne, Updates and counterfactuals, in: J. Allen, R. Fikes and E. Sandewall, eds., *Proceedings Second International Conference on Principles of Knowledge Representation and Reasoning* (Morgan Kaufmann, San Mateo, CA, 1991) 269–276.
[16] J. Habermas, *The Theory of Communicative Action* (Beacon Press, Boston, MA, 1984).
[17] S. Halldén, *On the Logic of Better*, Library of Theoria **2** (Lund, 1957).
[18] S. Halldén, Preference logic and theory choice, *Synthese* **16** (1966) 307–320.
[19] S. Halldén, *The Foundations of Decision Logic* (CWK Gleerup, Lund, 1980).
[20] J.Y. Halpern and Y. Moses, A guide to completeness and complexity for modal logics of knowledge and belief, *Artif. Intell.* **54** (1992) 319–379.
[21] B. Hansson, Fundamental axioms for preference relations, *Synthese* **18** (1968) 423–442.
[22] S.O. Hansson, A new semantical approach to the logic of preference, *Erkenntnis* **31** (1989) 1–42.
[23] S.O. Hansson, Similarity semantics and minimal changes of belief, *Erkenntnis* **37** (1992) 401–429.
[24] D. Harel, Dynamic logic, in: D. Gabbay and F. Guenthner, eds., *Handbook of Philosophical Logic*, Vol. II (Reidel, Dordrecht, Netherlands, 1984) 497–604.
[25] K. Hirofumi and A. Mendelzon, On the difference between updating a knowledge base and revising it, in: J. Allen, R. Fikes and E. Sandewall, eds., *Proceedings Second International Conference on Principles of Knowledge Representation and Reasoning* (Morgan Kaufmann, San Mateo, CA, 1991) 387–394.
[26] Z. Huang, Logics for agents with bounded rationality, ILLC Dissertation series 1994-10, University of Amsterdam (1994).
[27] Z. Huang and M. Masuch, Reasoning about action: a comparative survey, CCSOM Research Report 91-37 (1991).
[28] Z. Huang, M. Masuch and L. Pólos, A preference logic for rational actions, in: R. Blanning and D. King, eds., *Artificial Intelligence in Organization Design, Modeling and Control*, Information Systems Series (IEEE Computer Society Press, forthcoming).
[29] Z. Huang, M. Masuch and L. Pólos, ALX, the x'th action logic for agents with bounded rationality, CCSOM Research Report 92-70a (1992).
[30] Z. Huang, M. Masuch and L. Pólos, ALX2: the quantifier ALX logic, CCSOM Research Report 93-99 (1993).
[31] G.E. Hughes and M.J. Cresswell, *A Companion to Modal Logic* (Methuen, New York, 1984).
[32] F. Jackson, ed., *Conditionals* (Oxford University Press, 1991).
[33] P. Jackson, On the semantics of counterfactuals, in: *Proceedings IJCAI-89*, Detroit, MI (1989).
[34] R.C. Jeffrey, *The Logic of Decision* (New York, 2nd ed., 1983).
[35] S. Kambhampati and S. Kedar, A unified framework for explanation-based generalization of partially ordered and partially instantiated plans, *Artif. Intell.* **67** (1994) 29–70.

[36] H. Katsuno and A.O. Mendelzon, Propositional knowledge base revision and minimal change, *Artif. Intell.* **52** (1991) 263–294.
[37] A. Kron and V. Milovanović, Preference and choice, *Theory Decision* **6** (1975) 185–196.
[38] D.K. Lewis, *Counterfactuals* (Blackwell, Oxford, 1973).
[39] N. Luhmann, *The Differentiation of Society* (Columbia University Press, New York, 1982).
[40] J. McCarthy and P. Hayes, Some philosophical problems from the standpoint of AI, in: B. Meltzer and D. Michie, eds., *Machine Intelligence* **4** (Edinburgh University Press, 1969).
[41] J.G. March, The technology of foolishness, in: J.G. March and J.P. Olsen, eds., *Ambiguity and Choice in Organizations* (Bergen, Norway, Universitetsforlaget, 1976) 69–81.
[42] J.G. March and J.P. Olsen, Garbage can models of decision making in organizations, in: J.G. March and R. Weissinger-Baylon, eds., *Ambiguity and Command* (Pitman, Marshfield, MA, 1986) 11–53.
[43] N. Martí-Oliet and J. Meseguer, Action and change in rewriting logic, SRI-CSL Tech. Report 94-07 (1994).
[44] M. Masuch, Formalization of Thompson's organization in action, CCSOM Research Report 91-32 (1991).
[45] M. Masuch, Introduction, in: M. Masuch and M. Warglien, eds., *Artificial Intelligence in Organization and Management, Models of Distributed Activity* (Elsevier-North Holland, Amsterdam, 1992) 1–19.
[46] M. Masuch and Z. Huang, A logical deconstruction of organizational action: formalizing Thompson's *Organizations in Action* into a multi-agent action logic, CCSOM Working Paper 94-120.
[47] J.-J.C. Meyer, Using programming concepts in deontic reasoning, in: R. Bartsch, J. van Benthem and P. van Emde Boas, eds., *Semantics and Contextual Expression* (Foris Publications, Dordrecht, Netherlands, 1989) 117–145.
[48] N.J. Moutafakis, *The Logic of Preference* (Reidel, Dordrecht, Netherlands, 1987).
[49] J.D. Mullen, Does the logic of preference rest on a mistake, *Metaphilosophy* **10** (1979) 247–255.
[50] D. Nute, Conditional logic, in: D. Gabby and F. Guenthner, eds., *Handbook of Philosophical Logic*, Vol. II (1986) 387–439.
[51] J.F. Padgett, Managing garbage can hierarchies, *Administrative Sci. Q.* **25** (1980) 583–604.
[52] T. Parsons, *The Structure of Social Action* (Free Press, Glencoe, IL, 1937).
[53] T. Parsons, *The Social System* (Routledge and Kegan Paul, London, 1951).
[54] M.E. Pollack, The uses of plans, *Artif. Intell.* **57** (1992) 43–68.
[55] J.L. Pollock, *Subjunctive Reasoning* (Reidel, Dordrecht, Netherlands, 1976).
[56] J.L. Pollock, A refined theory of counterfactuals, *J. Philos. Logic* **10** (1981) 239–266.
[57] L. Pólos, Updated situation semantics, *J. Symbolic Logic* **58** (1993) 1143–1144.
[58] L. Pólos and M. Masuch, Information states in situation semantics, in: L. Pólos and M. Masuch, eds., *Applied Logic: How, What and Why* (Kluwer Academic Publishers, Dordrecht, Netherlands, to appear) 177–218.
[59] A.S. Rao and M.P. Georgeff, Modeling rational agents within a BDI-architecture, in: J. Allen, R. Fikes and E. Sandewall, eds., *Proceedings Second International Conference on Principles of Knowledge Representation and Reasoning* (Morgan Kaufmann, San Mateo, CA, 1991) 473–484.
[60] N. Rescher, Semantic foundations for the logic of preference, in: N. Rescher, ed., *The Logic of Decision and Action* (University of Pittsburgh Press, Philadelphia, PA, 1967).
[61] A. Schutz, *The Phenomenology of the Social World* (Northwestern University Press, Evanston, IL, 1967).
[62] D. Scott, Towards a mathematical theory of computation, in: *Proceedings Fourth Annual Princeton Conference on Information Science and Systems* (1970) 169–176.
[63] D. Scott, Domains for denotational semantics, in: M. Nielsen and E.T. Schmidt, eds., Lecture Notes in Computer Science **140** (Springer-Verlag, Berlin, 1982) 577–613.
[64] K. Segerberg, The logic of deliberation action, *J. Philos. Logic* **11** (1982) 233–254.
[65] H.A. Simon, A behavioral model of rational choice, *Q. J. Economics* **69** (1955) 99–118.
[66] H.A. Simon, On the concept of organizational goal, *Administrative Sci. Q.* **9** (1964) 1–22.
[67] H.A. Simon, Bounded rationality, in: J. Eatwell et al., eds., *The New Palgrave* (Macmillan, London, 1987).
[68] R. Stalnaker, A theory of conditionals, in: *Studies in Logical Theory, Amer. Philos. Q.* **2** (1968) 98–122.
[69] J.D. Thompson, *Organizations in Action, Social Science Bases of Administrative Theory* (McGraw-Hill, New York, 1967).

[70] M.P. Wellman and J. Doyle, Preferential semantics for goals, in: *Proceedings AAAI-91*, Anaheim, CA (AAAI Press, 1991) 698–703.
[71] M. Winslett, Reasoning about action using a possible models approach, in: *Proceedings AAAI-88*, St. Paul, MN (1988) 89–93.
[72] G.H. von Wright, *The Logic of Preference* (Edinburgh, 1963).
[73] G.H. von Wright, The logic of preference reconsidered, *Theory Decision* **3** (1972) 140–169.

ELSEVIER

Artificial Intelligence 82 (1996) 129–155

Artificial Intelligence

On the nature and role of modal truth criteria in planning

Subbarao Kambhampati [a,*], Dana S. Nau [b]

[a] *Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287-5406, USA*

[b] *Department of Computer Science, Institute for Systems Research, and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA*

Received June 1993; revised November 1994

Abstract

Chapman's paper, "Planning for conjunctive goals", has been widely acknowledged for its contribution toward understanding the nature of partial-order planning, and it has been one of the bases of later work by others—but it is not free of problems. This paper addresses some problems involving modal truth and the modal truth criterion (MTC). Our results are as follows: (i) Even though modal duality is a fundamental property of classical modal logics, it does not hold for modal truth in Chapman's plans; i.e., "necessarily p" is not equivalent to "not possibly $\neg p$". (ii) Although the MTC for necessary truth is correct, the MTC for possible truth is incorrect: it provides necessary but *insufficient* conditions for ensuring possible truth. Furthermore, even though necessary truth can be determined in polynomial time, possible truth is NP-hard. (iii) If we rewrite the MTC to talk about modal *conditional* truth (i.e., modal truth conditional on executability) rather than modal truth, then both the MTC for necessary conditional truth and the MTC for possible conditional truth are correct; and both can be computed in polynomial time. (iv) The MTC plays a different role in plan generation than it does in checking the correctness of plans, and this has led to several misconceptions about the MTC. Several researchers have mistakenly attempted to simplify the MTC by eliminating the white-knight declobbering clause from it; and others have used Chapman's results to conjecture that partial-order planning will not scale up to more expressive action representations. We point out that these ideas are misconceptions, and explain why.

* Corresponding author. Fax: (602) 965-2751. E-mail: rao@asu.edu.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0004-3702(94)00095-6

1. Introduction

Chapman's paper, "Planning for conjunctive goals", [2] has been widely acknowledged as an important step towards formalizing partial-order planning,[1] and it has been one of the bases of later work by others (for example [7, 8, 10, 12, 18, 31, 36]). Unfortunately, however, Chapman's work is not free of problems, and this has led to confusion about the meaning of his results. Previous papers [7, 8, 19, 36] have pointed out several of these problems.

One of the fundamental concepts used by Chapman is the idea of modal truth in plans. We will discuss the details of this concept later—but a simple version of it is that if P is a partially ordered, partially instantiated plan and p is a ground literal, then p is *necessarily* (or *possibly*) true in P's final situation if for every (or some) totally ordered ground instance P' of P, p is true after executing P'. Chapman's modal truth criterion (MTC) purports to give necessary and sufficient conditions for ensuring that p is necessarily or possibly true. As we describe below, this paper addresses several problems with modal truth and the MTC.

Modal duality and the MTC

Chapman explicitly states and proves the MTC for necessary truth, and claims that by modal duality (i.e., the equivalence of "necessarily p" and "not possibly $\neg p$"), the MTC for possible truth is obtained via a simple rewording of the MTC for necessary truth. But in this paper, we show that although modal duality is a fundamental property of classical modal logics, it does *not* hold for modal truth in Chapman's plans. This has several consequences:

(i) The MTC for possible truth is not completely correct: it provides necessary but insufficient conditions for ensuring possible truth. Furthermore, although necessary truth in plans can be computed in polynomial time as pointed out by Chapman,[2] the same is not true for possible truth. Instead, the problem of computing possible truth in plans is NP-hard.[3]

(ii) We can define a concept called *modal conditional truth*, which is similar to modal truth but does not require that a plan be executable as modal truth does. Necessary conditional truth and possible conditional truth *are* duals of each other, and both can be computed in polynomial time. Furthermore, if we rewrite the MTC to talk about modal conditional truth rather than modal truth, then both the MTC for necessary conditional truth *and* the MTC for possible conditional truth are correct.

[1] *Partial-order planning* is planning by searching in the space of partially ordered, partially instantiated plans. We prefer not to use the more common term "nonlinear planning", because it appears to mean different things to different people.

[2] There are some difficulties with Chapman's proof of this, but these difficulties have been cleared up by Nebel and Bäckström [26].

[3] If modal duality held, then both necessary truth and possible would be at similar levels of complexity: either both would be polynomial, or one would be NP-hard and the other co-NP-hard. Section 3.2.2 discusses some formulations of planning in which this occurs.

The role of the MTC in plan generation

The MTC plays a different role in plan generation than it does in checking the correctness of plans. In particular, the MTC provides both necessary and sufficient conditions for necessary truth—but it is possible to write sound and complete partial-order planners which use only *sufficient* but not *necessary* conditions for necessary truth.

This has led to a number of misconceptions about the MTC. Several researchers have mistakenly attempted to simplify the MTC by eliminating the white-knight declobbering clause[4] from it. Others (including Chapman) (cf. [2,23,34]) have used Chapman's results to conjecture that partial-order planning will not scale up to more expressive action representations. In this paper, we explain why both of these notions are incorrect—and observe that at the root of the confusion lies the peculiar predicament of partial-order planners, which search in the space of partially ordered partially instantiated plans, but need completeness only in the space of totally ordered ground plans.

This paper is organized as follows. Section 2 contains basic definitions, and clarifications and corrections of some of Chapman's terminology. Section 3 presents results about modal duality, the complexity of modal truth, and the modal truth criterion, and compares and contrasts these results with Chapman's claims, as well as with other related work. Section 4 discusses and clarifies the misconceptions regarding the role of modal truth criterion in plan generation vs. verification of plan correctness. Section 5 contains concluding remarks. Complicated proofs appear in the appendix; simpler proofs are in the body of the paper.

2. Definitions

Below, we have tried to be as compatible as possible with Chapman's "TWEAK-style" plans, situations, and modal truth. However, at certain points, technical problems have forced us to adopt a different approach. At those points, we explain how our approach differs and why.

2.1. Basics

The planning language $\mathcal{L}$ is any function-free first-order language.[5] Since $\mathcal{L}$ is function-free, every term is either a variable symbol or a constant symbol, and thus every ground term is a constant symbol. We follow the usual convention of defining an *atom* to be a predicate symbol followed a list of terms, a *literal* to be an atom or its negation, and a *proposition* to be a 0-ary atom. *Thus, what Chapman calls a proposition, we call a literal.*

[4] For the benefit of those unfamiliar with this clause, our presentation of the MTC in Section 3.2.1 points it out explicitly.

[5] Conventional first-order languages contain only finitely many constant symbols, but Chapman requires his planning language to contain an infinite number of constant symbols. For compatibility with Chapman's work, we will assume that $\mathcal{L}$ contains infinitely many constant symbols—but in Section 3.2.2 we will discuss what happens if the number of constant symbols is finite.

A *state* is any finite collection of ground atoms of $\mathcal{L}$. If a state s contains a ground atom p, then p is true in s and $\neg p$ is false in s; otherwise p is false in s and $\neg p$ is true in s. Thus, a state is simply an Herbrand interpretation for the language $\mathcal{L}$, and hence each formula of first-order logic is either satisfied or not satisfied in s according to the usual first-order logic definition of satisfaction.

If T is a finite set of terms, then a *codesignation constraint* on T is a syntactic expression of the form '$t \approx u$' or '$t \not\approx u$', where $t, u \in T$. Let D be a set of codesignation constraints on T, and θ be a *ground substitution* over T (i.e., a substitution that assigns a ground term to each variable in T). Then θ *satisfies* D if $t\theta = u\theta$ for every syntactic expression '$t \approx u$' in D, and $t\theta \neq u\theta$ for every syntactic expression '$t \not\approx u$' in D. D is *consistent* if there is at least one ground substitution θ that satisfies D. If $t\theta = u\theta$ for every θ that satisfies D, then t *codesignates with* u.[6]

A *step* is a triple $a = (\text{name}(a), \text{pre}(a), \text{post}(a))$, where $\text{name}(a)$ is a constant symbol called a's *name*, and $\text{pre}(a)$ and $\text{post}(a)$ are collections of literals called a's *preconditions* and *postconditions*.[7] If A is a set of steps, then an *ordering constraint* on A is a syntactic expression of the form '$a \prec b$' (read as "*a precedes b*"), where $a, b \in A$. If O is a set of ordering constraints on A and $\prec$ is a total ordering on A, then $\prec$ *satisfies* O if for every syntactic expression '$a \prec b$' in O, $a \prec b$.

A *partially ordered, partially instantiated plan* (or more succinctly, a *plan*) is a 4-tuple $P = (s_0, A, D, O)$, where s_0 is a state called P's *initial* state, A is a set of steps, D is a set of codesignation constraints on the terms of P (i.e., the terms in s_0 and A), and O is a set of ordering constraints on the steps of A. P is *complete* if it is totally ordered and ground, i.e., if there is a unique total ordering $a_1 \prec a_2 \prec \cdots \prec a_n$ over A that satisfies O, and a unique ground substitution θ over the terms of P that satisfies D. If P is not complete, then it is *incomplete*. If P is complete, we will often write P informally as $a_1 \prec a_2 \prec \cdots \prec a_n$.

Suppose that P is a complete plan, and let k be the largest integer $\leqslant n$ for which there are states $s_1, s_2, \ldots, s_k$ such that the following properties are satisfied for $1 \leqslant i \leqslant k$:

(i) s_{i-1} satisfies a_i's preconditions; i.e., $p\theta$ is true in s_{i-1} for every literal $p \in \text{pre}(a_i)$.

(ii) s_i is the state produced by performing the step a_i in the state s_{i-1}; i.e., $s_i = (s_{i-1} - f_i) \cup t_i$, where t_i is the set of all ground atoms $p\theta$ such that $p \in \text{post}(a_i)$, and f_i is the set of all negated ground atoms $p\theta$ such that $p \in \text{post}(a_i)$.

Then for $1 \leqslant i \leqslant k$, a_i is *executable* in the *input* state s_{i-1}, *producing* the *output* state s_i. If $k = n$, then P is *executable*, and it *produces* the *final* state s_n.

A plan $P' = (s'_0, A', D', O')$ is a *constrainment* of a plan $P = (s_0, A, D, O)$ if $s'_0 = s_0$, $A' = A$, $O \subseteq O'$, and $D \subseteq D'$. A *completion* of P is any constrainment of

[6] If D is consistent, then this is equivalent to saying that t codesignates with u iff '$t \approx u$' is in D's transitive closure.

[7] Informally, we will use the terms a and $\text{name}(a)$ interchangeably. Chapman's definition of a step actually omits $\text{name}(a)$ completely—but as pointed out by McAllester and Rosenblitt [22], unless we give unique names to steps, it is impossible for a plan to contain two distinct steps that have the same preconditions and postconditions.

P that is complete.[8] P is *consistent* if it has at least one completion; otherwise P is *inconsistent*.

2.2. Situations, truth, and modality

Chapman defines a *situation* to be a collection of literals.[9] Given a literal p and a situation s, he defines p to be true if it codesignates with a literal in s, and false if it codesignates with the negation of a literal in s. Chapman also makes the following definitions [2, p. 338]:

> A plan has an *initial situation*, which is a set of [literals] describing the world at the time that the plan is to be executed, and a *final situation*, which describes the state of the world after the whole plan has been executed. Associated with each step in a plan its *input situation*, which is the set of [literals] that are true in the world just before it is executed, and its *output situation*, which is the set of [literals] that are true in the world just after it is executed. In a complete plan, the input situation of each step is the same as the output situation of the previous step. The final situation of a complete plan has the same set of [literals] in it as the output situation of the last step. ... A [literal] is denied in a situation if its negation is asserted there. It is illegal for a [literal] to be both denied and asserted in a situation.

This approach leads to several difficulties:

(i) As pointed out by Yang and Tenenberg [36], if a plan P is not complete, then its situations are ill-defined. For example, suppose P consists of two unordered steps a and b, such that a asserts p and denies q, and b asserts q and denies p. Then P's final situation is either $\{p\}$ or $\{q\}$, depending on which completion of P we choose.

(ii) If a situation contains literals that are not completely ground, then what those literals mean is problematic. For example, suppose a plan's initial situation contains the literal $p(x)$, where x is a variable symbol. This cannot mean $(\forall x)p(x)$, because Chapman's TWEAK planner may later constrain $x \not\approx y$ for some constant or variable y. It cannot mean $(\exists x)p(x)$, because TWEAK may later constrain $x \approx y$. Apparently, it means $p(x)$ for some undetermined x, and TWEAK gets to choose what x is. In other words, if the initial situation contains any variables, then TWEAK changes the meaning of the initial situation as it goes along.

Thus, rather than using Chapman's approach, we define plans using STRIPS-style states of the world, and then define situations in terms of states. The intent of our definitions is that if a plan is complete and can be executed at least far enough to reach the situation s, then s corresponds to some state t that arises while executing the plan; and what

[8] Chapman's definition of a completion does not make it entirely clear whether a completion of P should include only the steps in P, or allow other steps to be added. However, other statements in his paper make it clear that he means for a completion to include only the steps in P, so this is how we and most others (e.g., [18,36]) use the term.

[9] He calls them propositions—but as mentioned at the beginning of Section 2.1, we call them literals instead.

is true and false in s is precisely what is true and false in t.[10] Otherwise, *nothing* is true or false in s although certain things may be *conditionally* true or false (as defined below). These ideas are formalized below.

If P is a plan, then associated with each step a of P are two symbols $\mathsf{in}(a)$ and $\mathsf{out}(a)$, called a's *input* and *output* situations. Associated with P are symbols init and fin called the *initial* and *final* situations of P. All of these symbols must be distinct. Whenever $a \prec b$, we will also say that $x \prec y$, where x may be a or $\mathsf{in}(a)$ or $\mathsf{out}(a)$, and y may be b or $\mathsf{in}(b)$ or $\mathsf{out}(b)$.[11]

We now define what is *true* and *false* in a situation of a complete plan. Let P be a complete plan, and p be a ground literal. Then p is true in init if p is true in P's initial state, and p is true in fin if p is true in P's final state. If a is an executable step of P, then p is true in $\mathsf{in}(a_i)$ (or $\mathsf{out}(a_i)$) if p is true in a's input state (or output state, respectively). A ground literal p is false in a situation s iff $\neg p$ is true in s. Note that if P is not executable, then the law of the excluded middle does not apply, for p will be neither true nor false in P's final situation.

As a consequence of the above definitions, it follows that p is true in s (which we write as $\mathcal{M}(p, s)$) iff the following three conditions are satisfied:

- *Establishment*: Either p codesignates with a postcondition of some step a such that $a \prec s$, or else $p \in s_0$.
- *Nondeletion*: For all steps b between a (or s_0) and s, no postcondition of b codesignates with $\neg p$.[12]
- *Executability*: Every step that precedes s is executable.

A closely related concept is *conditional truth*, which is like ordinary truth except that it does not require executability: p is *conditionally true* in s (which we write symbolically as $\mathcal{C}(p, s)$) iff the establishment and nondeletion conditions hold.

We defined truth and conditional truth only for complete plans, because for incomplete plans, what is true or conditionally true will vary depending on which completion we choose. In incomplete plans, we instead need to talk about *modal* truth, which Chapman defines as follows [2, p. 336]:

> I will say "*necessarily p*" if p is true of all completions of an incomplete plan, and "*possibly p*" if p is true of some completion.

Above, Chapman apparently means p to be nearly any statement about a plan: examples in his paper include not only statements about specific literals and situations in the plan, but also statements about the entire plan (e.g., the statement [2, p. 341] that a plan "necessarily solves the problem"). However, unless we place some restrictions on the nature of p, this has some dubious results—for example, if P is an incomplete plan,

[10] From our definition of a state earlier, the truth value of every literal p is known in the state t, and hence in the situation s. This differs from Chapman's formulation of a situation, in which the truth value of p is unknown in s unless s explicitly contains something codesignating with p or $\neg p$.

[11] According to this definition, $\mathsf{out}(a)$ and $\mathsf{in}(b)$ are always distinct; hence we would say $\mathsf{out}(a) \prec \mathsf{in}(b)$ in some cases where Chapman would say $\mathsf{out}(a) = \mathsf{in}(b)$. However, this makes no significant difference in any of the results.

[12] This is basically the white-knight declobbering clause of the MTC (see Section 3.2.1), simplified to handle the special case where the plan is complete.

Table 1
Relationships among truth, modal truth and modal conditional truth of a literal p in the final situation of a plan P

Complete plans	Incomplete plans
Truth $\mathcal{M}(p, \mathsf{fin})$: P is executable, *and* produces p in fin. This requires that the establishment, nondeletion and executability conditions hold.	Necessary truth $\Box\mathcal{M}(p, \mathsf{fin})$: *every* completion of P is executable, *and* produces p in fin.
	Possible truth $\Diamond\mathcal{M}(p, \mathsf{fin})$: *some* completion of P is executable, *and* produces p in fin.
Conditional truth $\mathcal{C}(p, \mathsf{fin})$: if P's steps did not have preconditions, then P would have produced p in fin. This requires the establishment and nondeletion conditions, but *not* the executability condition.	Necessary conditional truth $\Box\mathcal{C}(p, \mathsf{fin})$: if P's steps did not have preconditions, then *every* completion of P *would* have produced p in fin.
	Possible conditional truth $\Diamond\mathcal{C}(p, \mathsf{fin})$: if P's steps did not have preconditions, then *some* completion of P would have produced p in fin.

then all completions of P are complete, and therefore P itself is necessarily complete. Therefore, for the formal results in the paper, we will use "necessarily" and "possibly" only in the following cases (although we will sometimes use them informally in a broader sense). If p is an atom, P is a plan, and s is a situation in P, then:

- p is *necessarily* (or *possibly*) true in s (written $\Box\mathcal{M}(p, s)$ and $\Diamond\mathcal{M}(p, s)$, respectively) iff $\mathcal{M}(p, s)$ in every (or some) completion of P;
- p is *necessarily* (or *possibly*) conditionally true in s (written $\Box\mathcal{C}(p, s)$ and $\Diamond\mathcal{C}(p, s)$, respectively) iff $\mathcal{C}(p, s)$ in every (or some) completion of P.

Table 1 summarizes the relationships among truth, modal truth and modal conditional truth.

We now define the following decision problems:

- NECESSARY TRUTH: Given a ground literal p and a plan P, is p necessarily true in P's final situation fin? Or equivalently, does every completion of P produce a final state in which p is true?
- POSSIBLE TRUTH: Given a ground literal p and a plan P, is p possibly true in P's final situation fin? Or equivalently, is there a completion of P that produces a final state in which p is true?
- NECESSARY CONDITIONAL TRUTH: Given a ground literal p and a plan P, is p necessarily conditionally true in P's final situation fin?
- POSSIBLE CONDITIONAL TRUTH: Given a ground literal p and a plan P, is p possibly conditionally true in P's final situation fin?

3. Modal duality and the complexity of modal truth

3.1. Our results

From the definitions of truth and conditional truth in the previous section, it is easy to see that in each completion of a plan P, a literal p is true in the final situation fin if and only if (1) p is conditionally true in fin, and (2) the completion itself is executable. The completion is executable if and only if for every action a and every precondition p_a of a, p_a is conditionally true in the situation $\mathsf{in}(a)$. Thus,

$$\mathcal{M}(p,\mathsf{fin}) \equiv \left[\mathcal{C}(p,\mathsf{fin}) \wedge \underbrace{\left[\bigwedge_{\forall a \in P, \forall p_a \in \mathsf{pre}(a)} \mathcal{C}(p_a,\mathsf{in}(a))\right]}_{\text{Executability of the completion}}\right]. \tag{1}$$

Since ensuring that p is necessarily true in P's final situation is equivalent to ensuring that p is true in the final situation of every completion of P, we have:

$$\Box\mathcal{M}(p,\mathsf{fin}) \equiv \Box\left[\mathcal{C}(p,\mathsf{fin}) \wedge \left[\bigwedge_{\forall a \in P, \forall p_a \in \mathsf{pre}(a)} \mathcal{C}(p_a,\mathsf{in}(a))\right]\right]. \tag{2}$$

Now, since modal necessity distributes over conjunctions (i.e., $\Box(p \wedge q) \equiv \Box(p) \wedge \Box(q)$), we can rewrite Eq. (2) as

$$\Box\mathcal{M}(p,\mathsf{fin}) \equiv \left[\Box\mathcal{C}(p,\mathsf{fin}) \wedge \underbrace{\left[\bigwedge_{\forall a \in P, \forall p_a \in \mathsf{pre}(a)} \Box\mathcal{C}(p_a,\mathsf{in}(a))\right]}_{\text{Executability of all completions}}\right]. \tag{3}$$

Thus we can determine whether p is necessarily true by checking to see whether it is necessarily conditionally true, and whether the preconditions of each step of the plan are necessarily conditionally true. This can be done in polynomial time using the same technique Chapman suggests for computing the MTC in [2]. In particular, computing the necessary conditional truth of a literal in a situation (which involves checking whether the MTC's establishment and declobbering clauses are consistent with the plan's ordering and codesignation/noncodesignation constraints) can be done in time polynomial ($O(n^3)$) in the plan length. Thus, since the total number of preconditions in a plan is of the order of number of actions in the plan, computing whether p is necessarily true can also be done in polynomial time.

Coming to the case of possible truth, by similar arguments we get:

$$\Diamond\mathcal{M}(p,\mathsf{fin}) \equiv \Diamond\left[\mathcal{C}(p,\mathsf{fin}) \wedge \left[\bigwedge_{\forall a \in P, \forall p_a \in \mathsf{pre}(a)} \mathcal{C}(p_a,\mathsf{in}(a))\right]\right]. \tag{4}$$

But possible truth does not distribute over conjunctions (i.e., in general, $\Diamond(p \wedge q) \not\equiv \Diamond(p) \wedge \Diamond(q)$), so there is no way to simplify Eq. (4) into component tests of computing possible conditional truth of individual literals. Thus, even though possible conditional truth in fin and necessary conditional truth in fin are duals of each other (i.e., $\Diamond\mathcal{C}(p,\mathsf{fin}) \equiv \neg\Box\neg\mathcal{C}(p,\mathsf{fin})$), possible truth in fin and necessary truth in fin are *not* duals of each other. More specifically:

Theorem 1. *There exist a ground literal p and a plan P such that in P's final situation* fin, $\Box\mathcal{M}(p,\mathsf{fin}) \not\equiv \neg\Diamond\neg\mathcal{M}(p,\mathsf{fin})$.

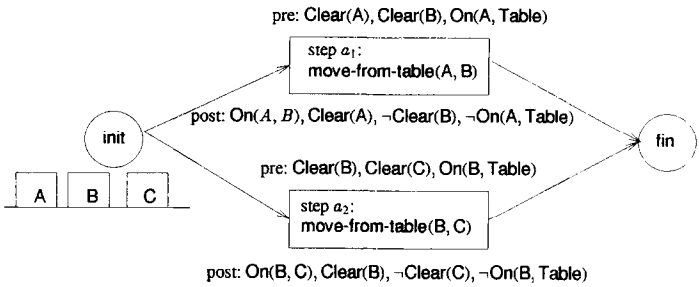


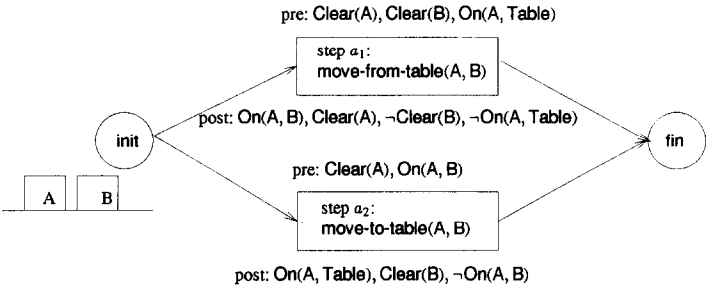
Fig. 1. A blocks-world plan illustrating the nonduality of necessary truth and possible truth in TWEAK-style plans. As shown in the drawing, the following atoms are true in the initial situation: On(A, Table), On(B, Table), On(C, Table), Clear(A), Clear(B), Clear(C). The operator move-from-table(x, y) moves block x from the table to block y. The preconditions and postconditions of each step are shown above and below the box that represents the step.

Proof. Consider the blocks-world plan shown in Fig. 1. This plan has only one executable completion, namely $a_2 \prec a_1$. This completion produces a final state in which A is on B and B is on C. Thus, no executable completion produces a final state where ¬On(A, B) is true, so ¬On(A, B) is not possibly true in the final situation fin. If possible truth and necessary truth were duals, then this would mean that On(A, B) is necessarily true in fin. However, On(A, B) is not necessarily true in fin, because the plan contains an *unexecutable* completion, namely $a_1 \prec a_2$. Thus possible truth and necessary truth are not duals. □

Thus, unlike necessary conditional truth and possible conditional truth, necessary truth and possible truth do not obey the modal duality that is obeyed by all classical modal logics [4, p. 62], and thus do not define a well-formed modal logic. It is easy to understand why this is so. The semantics of modal logics are based on Kripke structures (a.k.a. possible worlds). In this formulation, if p is a ground literal, then for every possible world, p must either be true or false in that world. For partially ordered plans, one might expect that each completion of the plan would give rise to a possible world. However, the modal truth of p in a situation of a plan requires that the plan's actions be executable in order to produce that situation. Thus, if a completion is not executable, then the truth of p is not defined in the corresponding possible world.[13]

Given a ground literal p and a plan P, p is possibly true in P's final situation if and only if there is an executable completion of P that produces a final state in which p is true, and this happens iff it is not the case that every executable completion of P produces a final state in which $\neg p$ is true. Thus, POSSIBLE TRUTH is the dual of the following problem:

[13] Although TWEAK plans cannot be modeled using the semantics of classical modal logics, they can be modeled in a variant of modal logics, called *first-order dynamic logic* [32]. Dynamic logic, which has been used to provide semantics for programs and plans, provides a clean way to separate executability/termination conditions from goal satisfaction conditions. More about this in Section 3.2.2.



Property	Atoms with that property
Necessarily true (produced by all completions)	*none*
Necessarily conditionally true (produced by all completions, if preconditions are stripped off)	Clear(A), On(B, Table)
Partially true (produced by all executable completions)	Clear(A), On(A, Table), Clear(B), On(B, Table)

Fig. 2. Example showing that partial truth is weaker than both necessary truth and necessary conditional truth. The table shows which atoms satisfy various truth conditions in fin.

- PARTIAL TRUTH: Given a ground literal p and a plan P, does every executable completion of P produce a final state in which p is true?[14]

Lemma A.1 of the appendix shows that PARTIAL TRUTH is NP-hard.

PARTIAL TRUTH is a weaker condition than both NECESSARY TRUTH and NECESSARY CONDITIONAL TRUTH.[15] The example in Fig. 2 illustrates this. There are some cases in which every executable completion of P produces a final state in which p is true, but p is neither necessarily true nor necessarily conditionally true in P's final situation.

Another way of understanding the problem with simplifying Eq. (4) is to note that if p is possibly conditionally true and that all the preconditions of the preceding actions are possibly conditionally true, this only implies that each of them is *individually* true in at least one completion—and this condition is *necessary* but *insufficient* for ensuring possible truth. We could check possible truth by checking to see whether all these conditions are *collectively* true in at least one completion of the plan, but since the number of completions of a plan is exponential in the number of actions of the plan, this would take exponential time. Furthermore, the following theorem (proved in the appendix) shows that unless P = NP, there is no polynomial-time approach for solving this problem.

[14] PARTIAL TRUTH corresponds closely to the notion of partial correctness, which was studied in connection with dynamic logic-based modeling of computer programming languages [29,32].

[15] It may seem at first glance that partial truth rather than necessary truth should form the basis of planning, since we are only interested in the executable completions of the plan. However, if a planner returned a plan whose conditions were all partially true rather than necessarily true, this would require the user of the plan to spend time trying to figure out which of the possibly exponential number of completions is actually executable.

Theorem 2. POSSIBLE TRUTH *is* NP-*hard.*

Thus, NECESSARY TRUTH and POSSIBLE TRUTH have different levels of complexity. If modal duality held, then this would not be so, for each would be reducible to the other's complement via an equivalence of the form $\Diamond\mathcal{M}(p,s) \equiv \neg\Box\neg\mathcal{M}(p,s)$. Thus it would follow [9, p. 29] that either POSSIBLE TRUTH would be polynomial like NECESSARY TRUTH, or else NECESSARY TRUTH would be co-NP-hard. In Section 3.2.2, we discuss some planning situations where this occurs.

3.2. Comparison with other work

3.2.1. The modal truth criterion

Chapman states the MTC as follows [2, p. 340]:

> *Modal truth criterion.* A [literal] p is necessarily true in a situation s iff two conditions hold:[16] there is a situation t equal or necessarily previous to s in which p is necessarily asserted; and for every step C possibly before s and every [literal] q possibly codesignating with p which C denies, there is a step W necessarily between C and s which asserts r, a [literal] such that r and p codesignate whenever p and q codesignate. The criterion for possible truth is exactly analogous, with all the modalities switched (read "necessary" for "possible" and vice versa).

If we take these words literally, then the definition of modal truth tells us that the plan must be modally executable. This is consistent with Chapman's definition of a situation (quoted in Section 2.2), from which it follows that a step's output situation (and hence what is true in that situation) is only defined if the step can be executed. However, a careful look at Chapman's proof of necessity and sufficiency of his MTC reveals that his proof deals with necessary *conditional* truth rather than necessary truth.[17] In proving that any literal with an establisher and no clobberer must be necessarily true, Chapman's proof refers to white-knight steps for every potential clobberer, [2, p. 370], without checking that the white knights are in fact executable.[18]

For the "necessary truth" version of the MTC, this does not affect the validity of Chapman's proof, since executability occurs naturally as a consequence of applying necessary conditional truth recursively to prerequisites of all preceding steps. The same, however, cannot be guaranteed for possible truth, since modal possibility does not commute over conjunctions—and thus Chapman's proof cannot be extended to possible truth. In particular, the following theorem shows that the "possible truth" version of the MTC sometimes fails:

[16] The second of these conditions is the "white-knight declobbering clause" that we refer to elsewhere. The steps C and W are often called a *clobberer* and a *white knight*, respectively. The situation t can be either the initial situation or the output situation of some step e, and in the latter case e is often called an *establisher*.

[17] Had Chapman explicitly noted this use of modal conditional truth in his proof, we believe he would have noticed the nonduality of necessary and possible truths.

[18] Note that in Chapman's terminology, the establisher is a *situation*, while clobberers and white knights are *steps*.

Theorem 3. *There is a plan P and a ground literal p such that in P's final situation, p is not possibly true but the MTC concludes otherwise.*

Proof. In the blocks-world plan in Fig. 2, consider the condition On(A, B), which holds in out(a_1). a_2 deletes On(A, B), but a_1 adds On(A, B), and a_1 can possibly come after a_2. Thus, the MTC would conclude that On(A, B) is possibly true in fin. In this case the MTC is incorrect, since there is no executable completion of the plan for which On(A, B) is true in the final state.[19] □

The MTC and modal conditional truth

The above discussion suggests an alternative interpretation of the MTC that sidesteps the problem: drop the executability requirement, and interpret the MTC as a statement about modal *conditional* truth rather than modal truth. This alternative interpretation is not as far-fetched as it might sound. To see this, note that Chapman defines the notion of truth of a literal in a situation as follows [2, p. 338]:

> A [literal] is true in a situation if it codesignates with a [literal] that is a member of the situation. A step asserts a [literal] in its output situation if the [literal] codesignates with a postcondition of the step.

Here, there is no explicit requirement that the step be executable. This suggests that the MTC does not require that *P* be modally executable, and thus suggests that Chapman was talking about modal conditional truth. This interpretation is also consistent with his "nondeterministic achievement procedure" [2, Fig. 7], where to make a literal necessarily true in a situation, he only ensures establishment and declobbering without explicitly stating that the establisher needs to be executable. (As explained above, for the case of necessary truth, executability follows from making every prerequisite of every action necessarily conditionally true.)

The "conditional truth" interpretation of the MTC gives a quasi-local flavor to planning, by separating the process of ensuring local establishment and declobbering from the process of ensuring executability, with the understanding that if all preconditions are necessarily established and declobbered, then the whole plan itself will be executable and correct. In fact, some latter rewrites of the MTC (e.g. [18,36]) use this interpretation to eliminate the notion of situations entirely, and state the MTC solely in terms of steps (operators) and their preconditions and postconditions.

Although a truth criterion for modal conditional truth does have utility in plan generation, it is of limited utility in projecting plans or partially ordered events. As mentioned in Section 2.2, the latter are more naturally related to modal truth.

3.2.2. Modal duality and universal executability

In Section 3.1, we observed that the main reason why necessary truth and possible truth are not duals in TWEAK-style plans is that such plans can contain unexecutable completions. Thus, one way to achieve duality between necessary truth and possible

[19] Note, however, that On(A, B) is possibly conditionally true in fin, because if the steps had no preconditions, then one of the completions would have produced On(A, B).

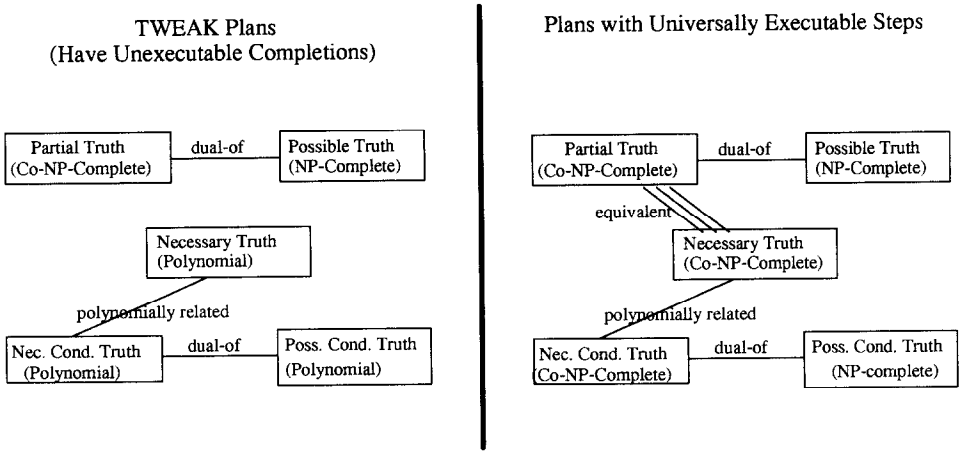


Fig. 3. Complexity relations among decision problems for plans containing actions that have executability preconditions, and plans containing universally executable actions.

truth is to restrict our attention to plans whose completions are always executable. One way to guarantee that plans will always be executable is to restrict the actions to have no preconditions, i.e., to consider only those plans P such that $\mathrm{pre}(a) = \emptyset$ for every step a of P. In this case, it is easy to see that Eqs. (2) and (4) in Section 3.1 will simplify respectively to:

$$\Box\mathcal{M}(p,s) \equiv \Box\mathcal{C}(p,s); \tag{5}$$

$$\Diamond\mathcal{M}(p,s) \equiv \Diamond\mathcal{C}(p,s). \tag{6}$$

In other words, for the set of plans composed entirely of precondition-less steps, modal truth and modal conditional truth are identical, necessary truth and possible truth are duals, and all are computable in polynomial time.

The above approach to achieving universal executability is clearly too restrictive, since it precludes modeling actions with any form of preconditions. But if we relax the restrictions of TWEAK-style action representation, there is a more reasonable way to guarantee universal executability: let an action a be executable even if its preconditions are not satisfied. If the preconditions are satisfied, then a will produce its postconditions; otherwise, a will simply have no effects. For plans that contain only this type of action, possible truth and necessary truth are duals of one another, computation of possible truth is NP-hard, and computation of necessary truth is co-NP-hard. Fig. 3 summarizes the complexity relations among the various decision problems. As discussed below, this approach has been used in different forms by several different researchers.

Propositional dynamic logic

To our knowledge, the above approach was first used in Rosenschein's work [32] on providing semantics to plans based on first-order propositional dynamic logic. Propositional Dynamic Logic (PDL) is a variant of modal logic, which was originally designed

to provide semantics to computer programs [29]. In PDL, the semantics of a program are described in terms of what will be necessarily and possibly true after the execution of that program. A program is said to be totally correct if (a) it halts, and (b) whenever it halts, certain goal propositions will be true in the resulting world. Programs that only satisfy condition (b) are said to be partially correct. (Note the similarity between partial correctness and PARTIAL TRUTH.) In using PDL to provide semantics to plans, Rosenschein guarantees universal executability of plans by starting with a loop-free subset of PDL, and restricting it further to allow only the so-called C-programs. C-programs restrict the use of conditionals in PDL to guarantee that the plan terminates irrespective of which branch of the conditional it takes.

Temporal projection

A very similar idea is used in Dean and Boddy's work on temporal projection [5]. Specifically, they use actions that have ground preconditions and effects. The effects of the actions are defined in terms of projection rules, which are of the form $\langle e, \phi, \alpha, \delta \rangle$, where e is an event with which the rule is associated, and ϕ is a set of antecedent conditions, which if true before e, will cause the α conditions to be added and the δ conditions to be deleted. Dean and Boddy are concerned with the following decision problem: given a partially ordered set of events A, does a condition C belong to Possible(e), where the latter is the set of conditions that hold immediately following the event e in some totally ordered ground instance (i.e., completion) of A.

In Dean and Boddy's formulation, A is executable even when a rule's preconditions don't hold (in which case the rule simply has no effect). Thus as discussed earlier, possible truth is equivalent to possible conditional truth, necessary truth is equivalent to necessary conditional truth, and possible truth and necessary truth are duals. Hence they are able to prove that in their formalism, determining possible truth is NP-hard and determining necessary truth is co-NP-hard.

Conditional steps

Chapman uses universally executable actions (he calls them conditional steps) in proving his intractability theorem for actions containing conditional effects. Specifically, Chapman defines a conditional step as follows [2, p. 371]:

> A conditional step is always applicable, but has two sets of postconditions, the if–true and the if–false postconditions. The if–true postconditions hold in the output situation if all the preconditions were satisfied in the input situation; otherwise the if–false postconditions hold.

Since these conditional steps are always applicable, a plan composed entirely of such steps will always be executable. Thus, just as in Dean and Boddy's formalism, determining necessary truth is co-NP-hard, as shown by Chapman in the proof of his Intractability Theorem.

Since the Intractability Theorem is based on planning operators that have conditional effects, it has been natural for planning researchers to interpret it to mean that the conditionality of these operators is what causes necessary truth to be intractable. However, this interpretation is misleading. The intractability result depends just as much on the

universal executability of Chapman's conditional steps as it does on their conditionality. Below we explain why.

Consider an incomplete plan P composed of ordinary "unconditional" steps as defined in Section 2, and let a be a step of P such that pre(a) and post(a) contain an unconstrained variable x. Then for the purposes of both planning and temporal projection, the effects of a are to some extent conditional. In particular, depending on what value we give to x, a will have different effects in different completions of P. However, computing necessary truth in such plans is still polynomial. Since Chapman's planning language has an infinite number of constant symbols, it follows that in the plan P we can *always* find a binding for x that makes a unexecutable. As a consequence, P will always have at least one unexecutable completion. Hence, determining necessary truth is trivial: nothing will be necessarily true in P's final situation.

Now, suppose we restrict our planning language $\mathcal{L}$ to contain only finitely many constant symbols (and thus only finitely many ground terms, since $\mathcal{L}$ is function-free). Then there will be some plans in which a is executable for every binding of x. In this case, as the following theorem shows, checking necessary truth will be co-NP-hard, even with unconditional steps.

Theorem 4. *If the language $\mathcal{L}$ contains only finitely many constant symbols, then* NECESSARY TRUTH *is co-*NP*-hard.*

Notice that this result is related to Chapman's observation [2, p. 356] that restricting the range of a variable to a finite set will defeat the MTC, and make constraint computations NP-complete.

Coherent plans

Nebel and Bäckström [26] have recently studied the computational complexity of plan validation and temporal projection. While our investigation was initially motivated by the apparent lack of modal duality in Chapman's MTC, Nebel and Bäckström's work is motivated by the apparent asymmetry between the complexity of plan validation as studied by Chapman, and temporal projection as studied by Dean and Boddy [5].

Although Nebel and Bäckström's results are related to ours, there are several significant differences. Rather than interpret the MTC in terms of modal conditional truth and use that to explain the asymmetry in the possible and necessary truth, as we have done in this paper, Nebel and Bäckström instead chose to restrict the MTC to apply only to plans whose completions are all executable (they call this property *coherence*). Furthermore, they restricted their plans to be ground (i.e., to contain no variables), and our plans do not have this restriction. We believe that the results in this paper complement theirs and together provide a coherent interpretation of the role of modal truth criteria in planning.

4. The role of the MTC in partial-order planning

Chapman's original motivation for the formulation of the MTC was to provide a formal basis for partial-order planning. Intuitively, since the MTC accounts for all the

scenarios in which a proposition p is necessarily true in a situation, we can make p necessarily true by simply adding constraints to the plan to *make* one of those scenarios true. While this use of the MTC provides a *sufficient* formal basis for partial-order planning, it turns out not to be *necessary*. More specifically, sound and complete partial-order planners:

- *do not need to* reason about the correctness of arbitrary partially ordered plans;
- *do not need to* consider only those goals that are not necessarily true for achievement;
- *do not need to* base their goal achievement procedure on a necessary and sufficient truth criterion for partially ordered plans.

These seemingly counter-intuitive facts are a result of the somewhat peculiar predicament of partial-order planners: they search in a space of plans that are partially ordered and partially instantiated, to find a totally ordered ground plan that solves the problem. This has lead to several confusions about the role of the MTC in planning. In this section, we will clarify the role played by the MTC in partial-order planning, and then address some misconceptions that resulted from misunderstandings in this regard.

4.1. Unnecessity of the MTC for partial-order planning

Many planning algorithms can be thought of as repeated iterations of the following steps: take a plan, evaluate it to see if it is a solution, and if it is not, then refine it further using a goal achievement procedure. For example, Chapman describes TWEAK, a planner based on his MTC, as follows:

> [The planner] enters a loop in which some goal *not yet achieved* is chosen and the [goal achievement] procedure is applied. ([2, p. 344]; emphasis ours)

> The goal achievement procedure is derived by interpreting the necessary truth criterion as a nondeterministic procedure. The criterion tells us all the ways a proposition could be necessarily true; the procedure chooses one of them and modifies the plan accordingly. [2, p. 341]

In the above, the MTC plays three separate roles: as a termination criterion (the planning stops when all the goals are necessarily correct), as a goal selection criterion (only conditions that are not necessarily true are selected for goal achievement), and as the basis for the nondeterministic goal achievement procedure.

Although the MTC is sufficient for serving these roles, it is not *required* for either of them—it is possible to provide a formal basis for partial-order planning without recourse to the MTC (cf. [28]). To see this, we must start with a clear understanding of the objectives of partial-order planning. For both partial-order and total-order planning, the objective is to find a ground operator sequence which when executed in the initial state produces a desired goal state. In the case of partial-order planning, the search is conducted in the space of partially ordered plans for efficiency of plan generation, and each partially ordered plan is simply a shorthand representation for the

set of all ground operator sequences that are consistent with the plan's constraints.[20] This means that one can do partial-order planning without having to reason about the "correctness" of partially ordered plans. Let us now re-examine how crucial the MTC is for termination, goal-selection and goal achievement procedures of a partial-order planner.

4.1.1. Termination

Since the objective of partial-order planning is to find plans that are ground and totally ordered, it follows that for sound and complete partial-order planning, it is sufficient for the termination condition to be capable of checking the correctness of totally ordered ground plans. Rather than using Chapman's MTC for this purpose, a termination condition such as the one given below can be used instead:

> *Eager termination.* Randomly generate a completion (ground linearization) of the current plan. If the completion solves the problem the planner is trying to solve, then terminate the planner and return the completion.

The eager termination criterion is always tractable, since a completion can be enumerated in polynomial time, and can be checked for correctness in polynomial time. Furthermore, a planner using this termination criterion will terminate on any incomplete plan on which a planner using MTC terminates. This is because whenever a plan satisfies the MTC, all of its completions (including the randomly generated one) are guaranteed to solve the problem the planner is trying to solve. Finally, the eager termination criterion may allow the planner to terminate earlier than the MTC would, because the MTC is not satisfied unless all completions of the current plan solve the problem.

4.1.2. Goal selection

Since the order in which goals are selected does not affect the completeness of partial-order planning [16, 22], MTC-based goal selection is just one of many possible goal selection strategies. Goal selection based on the MTC essentially boils down to preferring to work on preconditions of the plan that are not necessarily true, with the rationale that such a strategy may allow the planner to exploit any serendipitously satisfied goals by terminating without having to explicitly work on achieving them.[21] Replacing this strategy or complementing it with other goal selection strategies does not affect the soundness and completeness of the underlying planner.

4.1.3. Goal achievement

The last and perhaps most important role of the MTC in plan generation in TWEAK is as the basis for the goal achievement procedure. Although Chapman's interpretation of the MTC as a nondeterministic program provides a sufficient basis for goal achievement,

[20] See [16, 20] for an elaboration of this view.

[21] The occurrence of such a serendipitously satisfied goal has been referred to in [11] as an enabling-condition interaction.

it is once again not *necessary*. In fact, many partial-order planners, such as McAllester's SNLP [15,22,24], use a goal achievement procedure that corresponds to a modified MTC in which the white-knight declobbering clause is replaced with a much simpler demotion clause:

> A literal p is necessarily true in a situation s if p is necessarily asserted in a situation s' which necessarily precedes s, and p is necessarily not deleted by any step a that possibly comes between s' and s.

With this modification, the truth criterion is sufficient but unnecessary for ensuring the truth of a literal (see Section 4.2 for an example and related discussion). A planner using this modified truth criterion can still be complete, because for the special case of totally ordered plans, this criterion is equivalent to the MTC, and provides sufficient as well as necessary conditions for determining whether the plan solves the problem. To put it another way, for every plan P that is correct according to Chapman's MTC, there will be a constrainment P' of P that will be correct according to this modified truth criterion such that whenever TWEAK terminates with P, the planner using the modified truth criterion will terminate with the constrainment P'.

4.2. The role of white-knight declobbering

4.2.1. White-knight declobbering in checking plan correctness

Starting with the fact that one does not need the full power of Chapman's MTC in order to do sound and complete partial-order planning, some researchers have attempted to simplify the MTC by eliminating the white-knight clause from it. Unfortunately, such a simplification is erroneous. As Chapman's proof shows, something similar to the white-knight clause is still required if we want to state the necessary and sufficient conditions for the necessary truth of a literal in a given partially ordered plan (or equivalently, recognize the correctness of a given partially ordered plan) in *polynomial time*.

If we don't care about polynomial time, then we can simply enumerate all the completions of the plan, and verify that each completion is a correct totally ordered ground plan for solving the problem. But since the number of completions of a partially ordered plan is exponential in the size of the plan, this is very inefficient unless the plan is already totally ordered—in which case one can use a "nondeletion" condition similar to that we discussed in Section 2.2.

Although the necessity of the white-knight declobbering clause does depend on whether or not the plan is totally ordered, it does not depend on whether or not the plan is ground. (Chapman's use of a partially instantiated plan [2, Fig. 5, p. 339] to motivate white-knight declobbering seems to have caused this misimpression.) The following example, due to Mark Drummond [6], illustrates this point. Consider the ground partially ordered plan in Fig. 4, in which the literal p is required in the final situation fin, the steps b_1 and b_2 add p, the steps a_1 and a_2 delete p, and the steps a_1 and b_1 are unordered with respect to a_2 and b_2. We can see that p is true in the situation preceding fin in every completion of this plan. However, without the white-knight clause, the modal truth

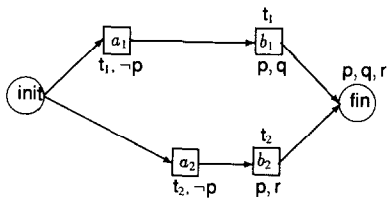


Fig. 4. A ground partially ordered plan for which the white-knight clause is needed to verify plan correctness (example due to Mark Drummond). Each step's name is in a box, with its preconditions and postconditions above and below the box. fin is the final situation, and init is the initial situation.

criterion would not be able to recognize this fact.[22]

Since the MTC can be used to determine efficiently (in polynomial time) whether all the completions of an arbitrary partially ordered plan are correct, it can also be used as a basis for removing any unnecessary orderings in a given plan in polynomial time [1, 19]: repeatedly remove some (nontransitive) ordering relation from the plan, and check if all its completions are still correct.[23] Such "order generalization" could be useful if one wants to execute steps of the plan in parallel in order to improve the execution time. It could also be useful when one wants to separate independent subparts of the plan to facilitate storage compactions in case-based approaches (cf. [17]).

4.2.2. *White-knight declobbering in goal achievement*

Although the white-knight declobbering clause is needed in the MTC, we pointed out in Section 4.1.3 that white-knight declobbering is not required to guide plan generation. In fact, Chapman's own implementation of TWEAK [2, p. 361], as well as many later partial-order planners such as SNLP [22] do not use white-knight declobbering clause in the goal achievement procedure. However, as Chapman remarks [2, p. 359], avoiding the white-knight declobbering clause during planning means that the planner may terminate with somewhat more constrained plans. For the example in Fig. 4, a planner such as SNLP that does not use the white-knight declobbering clause will find one of two alternate plans:

[22] *Historical Note*: Although the term "white knight" became popular after Chapman's work on TWEAK [2], Tate's Nonlin was the first planner to use a white-knight clause to specify weakest conditions for establishment and declobbering. Nonlin's Q&A procedure [33] says that a literal p is true at a step s in a partially ordered plan, if and only if (1) there exists a step n' such that $n' \prec n$, and n' asserts p, and (2) there does not exist a step n'' such that n'' deletes p, and (2.1) either n'' is unordered with respect to n or (2.2) there does not exist any step $n''' \prec n$ which deletes p without a subsequent node $w \succ n'''$ asserting it back again. According to this criterion, the plan in Fig. 4 is found to be correct. In Nonlin, this check is done by ensuring that (1) for every branch of the plan that is coming into s, the last node in the branch that gives a value to p must be asserting p and (2) no branch parallel to s contains a node that deletes p. Unlike TWEAK, Nonlin did not deal with partially instantiated plans (however, O-PLAN [3], a successor of Nonlin, does deal with such plans).

[23] Note that this involves removing existing orderings, without adding any new orderings. Bäckström [1] shows that if we also allow arbitrary addition and deletion of orderings, then the problem of finding the least constrained plan is NP-hard.

$$P_w^1 : \ a_1 \prec b_1 \prec a_2 \prec b_2;$$
$$P_w^2 : \ a_2 \prec b_2 \prec a_1 \prec b_1.$$

Note that both of these plans are constrainments of the original plan. However, since the objective of partial-order planning is only to find *a* ground operator sequence, terminating with these constrainments in itself is not a problem, unless there is a concomitant loss of efficiency in planning.[24]

This gives rise to the question of whether the use of white-knight declobbering will improve or reduce the efficiency of plan generation of a partial-order planner. The use of white-knight declobbering as part of the goal achievement procedure does tend to increase the redundancy in the search space (in particular, the same ground operator sequence may be considered in more than one search branch of the planner)—but as we discuss in [14, 15], whether or not such redundancy leads to inefficiency depends on the tradeoffs between the search space redundancy vs. level of commitment made by the planner. Thus using the white-knight declobbering clause does not *ipso facto* make planning inefficient, as has been conjectured by some researchers (cf. [13]). An important issue is whether the planner implements white-knight declobbering only through steps that already exist in the plan, or whether it also allows new steps to be introduced as white knights into the plan. In [15], we describe a planner called MP-I, which allows white-knight declobbering only via already existing steps. Our experiments show that this opportunistic declobbering leads to significant performance improvements in certain domains.

4.3. Plan generation with more expressive action representations

One unfortunate result of the misinterpretation of the role of the MTC in partial-order planning is the misconception that partial-order planning has more difficulty in scaling up to a more expressive action representation than does total-order planning. Not only did this belief slow down progress on planning with expressive action representations, it also inhibited some learning researchers from basing their work on partial-order planning frameworks [17, 23, 34].

As we noted earlier, planning can be seen as an iterative process, such that during each iteration the planner takes a plan, evaluates it to see if it is a solution, and refines

[24] A related question is whether the MTC, and in particular the white-knight declobbering clause, would be necessary in the goal achievement procedure if one wants to find optimal partially ordered plans (for example, to ensure optimal execution time, cf. [21]). At first glance, it might seem that we must search in the space of all partially ordered plans to find the optimal partially ordered plan, and thus the white-knight declobbering clause would be necessary. However, this is not strictly required, since the unnecessary ordering constraints can be removed from plans in polynomial time (Section 4.2.1). In particular, suppose we defined the cost of each plan P to be the execution time that would be needed for P if all unnecessary ordering constraints were removed. If this cost were used as part of an admissible search strategy in a planner like SNLP [22] or even a total-order planner, then the planner would terminate with some plan P_c such that P_c is a constrainment of an optimal partially ordered plan P_s. We could then derive P_s from P_c by once again removing all unnecessary ordering constraints from P_c.

it further (using the goal achievement procedure) if it is not. The argument about the disadvantages of partial-order planning for expressive action representation is based on the complexity of each of these iterations. It starts with Chapman's Intractability Theorem, which shows that if conditional steps are allowed, then necessary truth is NP-hard.[25] The argument goes that since a planner must compute necessary truth each time it evaluates and refines a plan, the amount of time taken per iteration will increase drastically in partial-order planning.

This reasoning is fallacious, since, as we noted earlier, determining necessary truth is *not required* in order to do partial-order planning. In particular, as we discussed in Section 4.1, a sound and complete partial-order planner will not have to compute necessary truth either for termination or for goal selection. Thus, the NP-hardness result is clearly irrelevant. Indeed, Pednault [27,28] provides a formal theory of partial-order planning in the presence of actions with conditional and quantified effects,[26] and his theory has served as the basis for a popular implementation called UCPOP [30], that takes only a polynomial amount of time per iteration.

Moreover, even if one were to compute necessary truth during goal selection and termination, it is possible to devise partial-order planners in which each plan that is generated is constrained in such a way that necessary truth can be evaluated in polynomial time. One extreme example of this would be a planner that generates only ground linear plans, but there are however other types of constrainments in which the plans are partially ordered—for example, unambiguous constrainments (cf. [24]), and safe constrainments (cf. [16,22])—which avoid the extreme of searching with totally ordered plans. All of these attempt to reduce the cost of plan evaluation and refinement by possibly increasing the search space size.

In order to determine the overall time complexity of the planner, what really counts not the time per iteration, but the tradeoff between the time per iteration and the size of the space searched (i.e., the number of iterations), since the time complexity is the product of these two factors. In [16,20], we systematically classify the types of operations (called "tractability refinements") used by various planners to ensure tractable plan evaluation and analyze the tradeoffs offered by them.

Finally, it is also wrong to believe that planning itself is more difficult if conditional operators are allowed. Erol et al. [8] have analyzed how the complexity of planning varies under a wide variety of conditions, including whether or not function symbols, negative preconditions, or delete lists (i.e., negative postconditions) are allowed, whether or not the predicates are propositional (i.e., 0-ary), and whether the planning operators are part of the input or fixed in advance. In all of these cases, the presence or absence of conditional operators made no difference in the complexity or decidability of planning.

[25] However, as we discussed in Section 3.2.2, the NP-hardness depends as much on the universal executability of these steps as it does on the conditionality of their effects.

[26] Instead of checking for necessary truth, Pednault's theory of planning concentrates on adding a sufficient number of constraints (including steps, orderings, bindings, and secondary preconditions) to ensure necessary truth in the resulting plan.

5. Concluding remarks

In this paper, we have discussed several misconceptions regarding the role of modal truth and the modal truth criterion (MTC) in planning. Along the way, we have also clarified and corrected several problems with Chapman's terminology.

First, we have presented the following results about modal truth and the modal truth criterion:

(i) Contrary to Chapman's statement, the principle of modal duality that is obeyed by all classical modal logics is not obeyed in TWEAK-style plans. The lack of duality between necessary truth and possible truth is related to the fact that modal truth of a literal in a situation of a plan requires that the plan's actions be executable in order to produce that situation. To achieve modal duality, one needs universally executable plans.

(ii) Even though necessary truth in plans can be determined in polynomial time as stated by Chapman, the same statement does not hold for possible truth. Instead, the problem of determining possible truth in plans is NP-hard. This is important because checking possible truth has several applications in plan projection [5] as well as plan generalization [19].

(iii) As stated by Chapman, the MTC is correct only as a criterion for necessary truth (not as a criterion for possible truth). However, if we reinterpret it as a criterion for modal *conditional* truth (i.e., modal truth conditional on plan executability), then it is correct as a criterion for both necessary conditional truth and possible conditional truth.

Second, we clarified the role of the MTC in plan generation vs. checking the correctness of a given plan, by emphasizing the peculiar predicament of partial-order planners: they search in the space of partially ordered partially instantiated plans, but need completeness only in the space of totally ordered and totally instantiated plans. We showed that misunderstandings in this regard have been the root of several of the confusions regarding the role of the MTC:

(i) Sound and complete partial-order planning is possible as long as the goal achievement procedure is based on a truth criterion that is consistent with the necessary and sufficient truth criterion for totally orderered plans.

(ii) Although the MTC provides a sufficient basis for partial-order planning, it is *not necessary* for sound and complete partial-order planning. Specifically, it is possible to devise sound and complete partial-order planners whose termination, goal selection and goal achievement procedures do not depend upon Chapman's MTC.

(iii) Although the white-knight declobbering clause of the MTC is needed in order to provide both necessary and sufficient conditions for ensuring truth of a literal in a partially ordered plan, white-knight declobbering is not required for partial-order planning. Several sound and complete partial-order planners use demotion instead.

(iv) Although Chapman proved that is NP-hard to verify necessary truth in plans whose steps have conditional effects, this does not necessarily imply (as has been conjectured elsewhere) that partial-order planners are any worse off than

total-order planners in dealing with actions that have conditional effects. There are two reasons for this: (1) partial-order planners do not have to compute necessary truth in order to be sound and complete, and (2) even if a partial-order planner does compute necessary truth, it can be sound and complete while only generating plans for which necessary truth can be computed in polynomial time.

Because of the wide impact of Chapman's paper, it is important to correct any misimpressions that may result from it. We hope readers will find this paper useful for that purpose.

Appendix A. Proofs

Lemma A.1. PARTIAL TRUTH *is co*-NP-*hard.*

Proof. The proof is by reduction from the complement of 3-SAT (satisfiability with three literals per clause). In particular, let $X = c_1 + c_2 + \cdots + c_m$ be a DNF formula over the Boolean variables $x_1, x_2, \ldots, x_n$, where each c_i is a conjunct of three literals $c_i = l_{i1} l_{i2} l_{i3}$. We encode X as a plan P_X^* and a ground atom sat(yes, yes, yes), such that every executable completion of P_X^* produces a final situation containing sat(yes, yes, yes) iff X is a tautology. P_X^* is the following plan (see Fig. A.1):

- *Initial state.* P_X^*'s initial state s_0 is the empty set.
- *Steps.* For each Boolean variable x_i, there are two steps, Set_i and Unset_i.
  - Set_i has no preconditions, and has the following postconditions:

    $$\neg\overline{\mathsf{x}}_i(\mathsf{yes}), \overline{\mathsf{x}}_i(\mathsf{no}), \neg\mathsf{x}_i(\mathsf{no}), \mathsf{x}_i(\mathsf{yes}).$$

  - Unset_i has no preconditions, and has the following postconditions:

    $$\overline{\mathsf{x}}_i(\mathsf{yes}), \neg\overline{\mathsf{x}}_i(\mathsf{no}), \mathsf{x}_i(\mathsf{no}), \neg\mathsf{x}_i(\mathsf{yes}).$$

    Here, yes and no are constant symbols; the interpretations of $\mathsf{x}_i(\mathsf{yes})$, $\overline{\mathsf{x}}_i(\mathsf{no})$, $\overline{\mathsf{x}}_i(\mathsf{yes})$, and $\mathsf{x}_i(\mathsf{no})$ are that the Boolean variable x_i is true, not false, false, and not true, respectively. Thus, the interpretations of Set_i and Unset_i are that they make x_i true and false, respectively.
  - There is a step Sep, which has no preconditions nor postconditions.[27] Its only purpose is to separate the steps Set_i and Unset_i (defined above) from the steps Con_i defined below.
  - For each conjunct $c_i = l_{i1} l_{i2} l_{i3}$ in X, there is a step Con_i. Corresponding to the literals in c_i, Con_i has preconditions L_{i1}, L_{i2}, L_{i3}, as follows. Each l_{ij} is either x_k or $\overline{x_k}$ for some x_k. If $l_{ij} = x_k$, then L_{ij} is $\mathsf{x}_k(v_{ij})$, where v_{ij} is a variable;

[27] To prove his Intractability Theorem, Chapman also uses steps that have no preconditions and postconditions. However, this raises the question of whether TWEAK can ever create a plan such as P_X^*. It is easy to modify P_X^* so that TWEAK will construct it; here's how. For each step a of P_X^*, add a new postcondition done(name(a)) (recall that name(a) is a constant symbol). For each ordering constraint '$a \prec b$' of P_X^*, give b a new precondition done(name(a)).

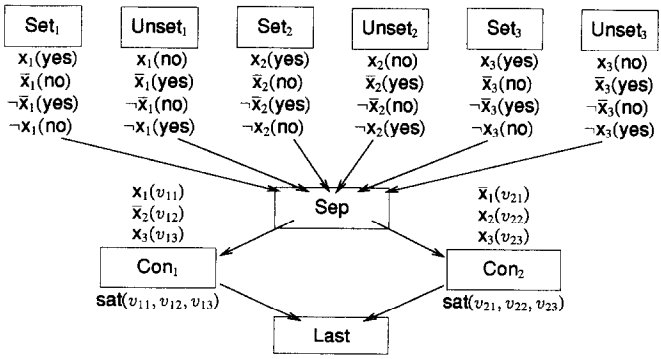


Fig. A.1. An example of the plan P_X^* in the case where $X = x_1\overline{x_2}x_3 + \overline{x_1}x_2x_3$. Each step's name is in a box, with its preconditions and postconditions above and below the box.

if $l_{ij} = \overline{x_k}$, then L_{ij} is $\overline{\mathsf{x}}_k(v_{ij})$. Con_i has one postcondition: $\mathsf{sat}(v_{i1}, v_{i2}, v_{i3})$. The interpretation of $\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ is that X is satisfied. For any other constant symbols u, v and w, $\mathsf{sat}(u, v, w)$ has no particular interpretation. Thus, the interpretation of Con_i is that if $c_i = l_{i1}l_{i2}l_{i3}$ is satisfied, then Con_i asserts that X is satisfied.

- There is one other step, Last, which has no preconditions and no postconditions. Last's purpose is to provide a final step in the plan.

- *Constraints.* O contains an ordering constraint '$\mathsf{Set}_i \prec \mathsf{Sep}$' for every Set_i, an ordering constraint '$\mathsf{Unset}_i \prec \mathsf{Sep}$' for every Unset_i, and ordering constraints '$\mathsf{Sep} \prec \mathsf{Con}_i$' and '$\mathsf{Con}_i \prec \mathsf{Last}$' for every Con_i. There are no other ordering constraints. There are no codesignation constraints; i.e., $D = \emptyset$.

Let P be any executable completion of P_X^*, and θ be the unique ground substitution that satisfies P's codesignation constraints. In P, Sep's input and output states are a set s of ground atoms corresponding to truth values for all the x_i. More specifically, $s = s_1 \cup s_2 \cup \cdots \cup s_n$, where each s_k is either $\{\overline{\mathsf{x}}_k(\mathsf{yes}), \mathsf{x}_k(\mathsf{no})\}$ (meaning x_k is false), or $\{\overline{\mathsf{x}}_k(\mathsf{yes}), \mathsf{x}_k(\mathsf{no})\}$ (meaning x_k is true).

The input state for each Con_i consists of some ground atoms of the form $\mathsf{sat}(u, v, w)$, plus the set s described above. Since Con_i is executable, each precondition L_{ij} of Con_i codesignates with an atom in Con_i's input state. In particular, since each L_{ij} is either $\mathsf{x}_k(v_{ij})$ or $\overline{\mathsf{x}}_k(v_{ij})$ for some k, it follows that $L_{ij}\theta \in s_k$. Thus, either $v_{ij}\theta = \mathsf{yes}$ or $v_{ij}\theta = \mathsf{no}$, depending on whether s_k corresponds to a truth value for x_k that makes l_{ij} true, or one that makes l_{ij} false. Con_i asserts $\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ iff s corresponds to a set of truth values that make l_{i1}, l_{i2}, and l_{i3} all true.

Thus, P produces a final state containing $\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ iff s corresponds to a set of truth values that makes at least one of the conjuncts $c_i = l_{i1}l_{i2}l_{i3}$ true. Since s may correspond to any assignment of truth values to $x_1, x_2, \ldots, x_n$, this means that all executable completions of P_X^* produce final states containing $\mathsf{sat}(\mathsf{yes}, \mathsf{yes}, \mathsf{yes})$ iff $X = c_1 + c_2 + \cdots + c_m$ is true for all assignments of truth values to $x_1, x_2, \ldots, x_n$. $\square$

Theorem 2. POSSIBLE TRUTH *is* NP-*hard.*

Proof. Let $Y = c_1 c_2 \ldots c_m$ be a CNF formula over the Boolean variables $y_1, y_2, \ldots, y_n$, with three literals in each disjunctive clause c_i. Let $X = \neg Y$. Using de Morgan's laws, in linear time we can express X as a DNF formula over $y_1, y_2, \ldots, y_n$, with three literals in each conjunct.

Suppose Y is unsatisfiable. Then X is a tautology, so from the proof of Lemma A.1, every executable completion of P_X^* produces a final state containing sat(yes, yes, yes). Thus, no executable completion of P_X^* produces a final state in which ¬sat(yes, yes, yes) is true, so ¬sat(yes, yes, yes) is not possibly true in P_X^*'s final situation.

Suppose Y is satisfiable. Then X is not a tautology, so from the proof of Lemma A.1, there is an executable completion P of P_X^* that produces a final state that does not contain sat(yes, yes, yes). Thus ¬sat(yes, yes, yes) is true in P's final state, so it is possibly true in P_X^*'s final situation. □

Remark A.2. The above proof makes use of the duality between satisfiability checking and tautology checking. However, it is also quite straightforward to prove the theorem without using this duality, by constructing a plan Q_Y^* and a ground atom sat(yes, ..., yes) such that that some completion of Q_Y^* produces sat(yes, ..., yes) iff Y is satisfiable. Such a proof appears in [25].

Theorem 4. *If the language $\mathcal{L}$ contains only finitely many constant symbols, then* NECESSARY TRUTH *is co*-NP-*hard.*

Proof. In the proof of Lemma A.1, suppose we specify that the only constant symbols in the language $\mathcal{L}$ are yes and no. Then every completion of P_X^* is executable, and thus sat(yes, yes, yes) is necessarily true in fin iff the formula X is a tautology.

Even if $\mathcal{L}$ contains finitely many additional constant symbols, we can still make sat(yes, yes, yes) necessarily true in fin iff the formula X is a tautology, by adding codesignation constraints to P_X^* of the form $v \not\equiv c$ for each constant symbol c other than yes or no, and each variable v appearing in the steps Con_1 and Con_2. Thus Lemma A.1 shows that if $\mathcal{L}$ contains only finitely many constant symbols, then NECESSARY TRUTH is co-NP-hard even with ordinary "unconditional" steps. □

Acknowledgements

We wish to thank Christer Bäckström, John Bresina, Mark Drummond, Kutluhan Erol, Jim Hendler, Craig Knoblock, Ed Pednault, Arunabha Sen, Austin Tate and V.S. Subrahmanian for their helpful criticisms and comments.

Kambhampati's research is supported in part by an NSF Research Initiation Award IRI-9210997, an NSF Young Investigator Award IRI-9457634, and ARPA/Rome Laboratory planning initiative under grants F30602-93-C-0039 and F30602-95-C-0247. Nau's research is supported in part by NSF grants IRI-9306580, NSF EEC 94-02384 and ARPA grant DATB63-95-C-0037. Any opinions, findings, and conclusions or recommendations

expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or ARPA.

References

[1] C. Bäckström, Finding least-constrained plans and optimal parallel executions is harder than we thought, in: *Proceedings 2nd European Workshop on Planning* (1993).
[2] D. Chapman, Planning for conjunctive goals, *Artif. Intell.* **32** (1987) 333–379.
[3] K. Currie and A. Tate, O-Plan: the open planning architecture, *Artif. Intell.* **52** (1991) 49–86.
[4] E. Davis, *Representations of Commonsense Knowledge* (Morgan Kaufmann, San Mateo, CA).
[5] T. Dean and M. Boddy, Reasoning about partially ordered events, *Artif. Intell.* **36** (1988) 375–399.
[6] M. Drummond, Private communication, 1991.
[7] K. Erol, D. Nau and V.S. Subrahmanian, When is planning decidable?, in: *Proceedings First International Conference AI Planning Systems* (1992) 222–227.
[8] K. Erol, D.S. Nau and V.S. Subrahmanian, Complexity, decidability and undecidability results for domain-independent planning, *Artif. Intell.* **76** (1995) 75–88.
[9] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, New York, 1979).
[10] M.L. Ginsberg, What is a modal truth criterion?, Unpublished manuscript (1990).
[11] N. Gupta and D.S. Nau, On the complexity of blocks-world planning, *Artif. Intell.* **56** (1992) 223–254.
[12] S. Hanks and D.S. Weld, Systematic adaptation for case-based planning, in: *Proceedings First International Conference AI Planning Systems* (1992) 96–105.
[13] E. Jacopin, C. Le Pape and J.F. Puget, A theoretical analysis of the "uselessness" of white-knights, Tech. Rept. 92/27, Institut Blaise Pascal (1992).
[14] S. Kambhampati, On the utility of systematicity: Understanding tradeoffs between redundancy and commitment in partial ordering planning, in: *Proceedings IJCAI-93*, Chambery, France (1993).
[15] S. Kambhampati, Multi-contributor causal structures for planning: a formalization and evaluation, *Artif. Intell.* **69** (1994) 235–278.
[16] S. Kambhampati, Refinement search as a unifying framework for analyzing planning algorithms, in: *Proceedings 4th International Conference on Principles of Knowledge Representation and Reasoning*, Bonn (1994).
[17] S. Kambhampati and J. Chen, Relative utility of EBG based plan reuse in total ordering vs. partial ordering planning, in: *Proceedings AAAI-93*, Washington, DC (1993).
[18] S. Kambhampati and S. Kedar, Explanation-based generalization of partially ordered plans, in: *Proceedings AAAI-91*, Anaheim, CA (1991) 679–685.
[19] S. Kambhampati and S. Kedar, A unified framework for explanation-based generalization of partially ordered and partially instantiated plans, *Artif. Intell.* **67** (1994) 29–70.
[20] S. Kambhampati, C.A. Knoblock and Q. Yang, Planning as refinement search: a unified framework for evaluating design tradeoffs in partial order planning, *Artif. Intell.* **76** (1995) 167–238.
[21] C.A. Knoblock, Generating parallel execution plans with a partial-order planner, in: *Proceedings AIPS-94* (1994).
[22] D. McAllester and D. Rosenblitt, Systematic nonlinear planning, in: *Proceedings AAAI-91*, Anaheim, CA (1991) 634–639.
[23] D. McDermott, Regression planning, *Int. J. Intelligent Systems* **6** (1991) 357–416.
[24] S. Minton, M. Drummond, J. Bresina and A. Philips, Total order vs. partial order planning: factors influencing performance, in: *Proceedings Third International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA (1992).
[25] D. Nau, On the complexity of possible truth, in: *Proceedings AAAI Spring Symposium*, Stanford, CA (1993).
[26] B. Nebel and C. Bäckström, On the computational complexity of temporal projection, planning and plan validation, *Artif. Intell.* **66** (1994) 125–160.
[27] E.P.D. Pednault, Synthesizing plans that contain actions with context-dependent effects, *Comput. Intell.* **4** (1988) 356–372.

[28] E.P.D. Pednault, Generalizing nonlinear planning to handle complex goals and actions with context-dependent effects, in: *Proceedings IJCAI-91*, Sydney, Australia (1991).

[29] V. Pratt, Semantical considerations on Floyd–Hoare logic, in: *Proceedings 17th Annual Symposium on Foundations of Computer Science*, Houston, TX (1977) 109–121.

[30] J.S. Penberthy and D.S. Weld, UCPOP: a sound, complete, partial order planner for ADL, in: *Proceedings Third International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA (1992).

[31] M.A. Peot, Conditional nonlinear planning, in: *Proceedings First International Conference on AI Planning Systems* (1992) 189–197.

[32] S. Rosenschein, Plan synthesis: a logical perspective, in: *Proceedings IJCAI-81*, Vancouver, BC (1981) 331–337.

[33] A. Tate, Generating project networks, in: *Proceedings IJCAI-77*, Cambridge, MA (1977) 888–893.

[34] M.M. Veloso, M.A. Perez and J.G. Carbonell, Nonlinear planning with parallel resource allocation, in: *Proceedings Workshop on Innovative Approaches to Planning, Scheduling and Control* (1990) 207–212.

[35] M.M. Veloso, Learning by analogical reasoning in general problem solving, Ph.D. thesis (CMU-CS-92-174), Carnegie-Mellon University, Pittsburgh, PA (1992).

[36] Q. Yang and J.D. Tenenberg, Abtweak: abstracting a nonlinear, least commitment planner, in: *Proceedings AAAI-90*, Boston, MA (1990) 204–209.

ELSEVIER

Artificial Intelligence 82 (1996) 157–179

Artificial Intelligence

Noise modelling and evaluating learning from examples

Ray J. Hickey*

Faculty of Informatics, University of Ulster at Coleraine, Ulster, Co. Londonderry BT52 1SA, N. Ireland, UK

Received July 1993; revised July 1994

Abstract

The means of evaluating, using artificial data, algorithms, such as ID3, which learn concepts from examples is enhanced and referred to as the method of artificial universes. The central notions are that of a class model and its associated representations in which a class attribute is treated as a dependent variable with description attributes functioning as the independent variables. The nature of noise in the model is discussed and modelled using information-theoretic ideas especially that of majorisation. The notion of an irrelevant attribute is also considered. The ideas are illustrated through the construction of a small universe which is then altered to increase noise. Learning curves for ID3 used on data generated from these universes are estimated from trials. These show that increasing noise has a detrimental effect on learning.

1. Introduction

Supervised learning of concepts from classified examples remains a problem of major interest. Amongst the many approaches that have been taken to this task are: induction of decision trees (Quinlan [24]), or high-level rules (Clark and Niblett [8]); instance-based learning (see Aha [2] and Cost and Salzberg [9]); artificial neural networks (Rumelhart [26]); genetic classifiers (see Booker, Goldberg and Holland [4]); Bayesian classifiers (see Cheeseman [6] and Langley, Iba and Thompson [17]).

Typically the examples presented to the algorithm are representative in some sense of a set of possible examples and often constitute a very small subset. Each example consists of a description in an appropriate representation language (often just a simple attribute–value formalism) together with the assigned class or concept offered by the teacher. The learning task is therefore one of induction of a general concept description from the particular cases provided.

* Telephone: +44-(0)1265-44141. E-mail: rj.hickey@ulst.ac.uk.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0004-3702(94)00094-8

In real-world settings the task is complicated by the presence of noise of various forms such as errors in recording attribute values or in classification by the teacher. This not only makes the work of the learning algorithm more difficult but also complicates the evaluation of its performance.

Performance of an algorithm or comparison of several algorithms is usually assessed by one of the following means:

(1) *Empirical analysis using real data.* From a database of examples (such as one of the standard sets kept in the "Machine Learning Repository" (Murphy and Aha [21])) subsets are drawn at random and used for learning. Further subsets are then drawn and used for evaluation of the classification performance of the learned description.

(2) *Empirical analysis using artificial data.* In order to simulate the effects of noise, data is generated according to a given prescription and then aspects of the example description or the class are altered using a mechanism involving known probabilities. Irrelevant attributes may be introduced.

(3) *Average-case analysis.* Examples are generated according to a known probabilistic prescription. The expected value behaviour of the learning algorithm is then derived.

(4) *PAC analysis.* The probably approximately correct (PAC) theory of Valiant [31] provides a theoretical basis for assessing performance. A concept is said to be PAC-learnable by an algorithm if with probability $1-\delta$ the learned concept description has a probability $1-\varepsilon$ of classifying almost correctly on subsequent trials. Although the analysis is probabilistic, no underlying distribution for examples is assumed and it is therefore "worst case" over all possible distributions.

In the first three methods a learning curve showing performance (typically classification accuracy) against number of training examples can be derived. For methods (1) and (2) this curve is estimated from data over many trials. Kibler and Langley [16] argue that method (2) provides greater opportunity for systematic investigation than method (1) particularly with regard to the controlled administration of noise; in method (1) the naturally occurring noise cannot be quantified satisfactorily. Amongst the most well-known artificial data sets containing an element of noise is the LED domain of Breiman et al. [5] where components of an LED display for digits are inverted with a small probability. Aha [1] has introduced a variation of the empirical approach by taking a database and altering it in a random fashion to produce a variant for experimentation (called a case) which retains the essential characteristics of the original.

In method (3) the theoretical learning curve can, mathematics permitting, be derived from the underlying setup (into which noise may be introduced); see Pazzani and Sarrett [22] or Langley et al. [17]. Unfortunately this approach seems feasible only for simple algorithms. Analysis of more sophisticated algorithms such as ID3 (Quinlan [24, 25]) or backpropagation (Rumelhart [26]) would appear to be too difficult.

Unlike methods (1)–(3), the PAC approach offers very little insight into the performance of an algorithm in typical circumstances and is overly pessimistic

about its capabilities. Pazzani and Sarrett [22] show learning curves derived from the average case and PAC approaches. For noise modelling in the PAC approach see Valiant [32], Angluin and Laird [3] or Sakakibara [27].

1.1. Modelling noise and extending the use of artificial data: the method of artificial universes

Although, as indicated above, the introduction of noise in a systematic way can be undertaken in methods (2), (3) and (4), in practice the modelling has often been ad hoc and lacking in any underlying theory. The resulting artificial domains are then not sufficiently realistic with regard to complexity of noise. Also it may not be clear how much noise overall has been introduced.

The purpose of this work is to enhance the capabilities of method (2) by introducing a unified theory of noise which makes it possible to measure and modify with ease the amount of noise in an artificial system. Amongst the benefits of this approach are:

(1) a simple means of producing data sets which possess the required amount of noise and which provide challenging tasks for a wide range of algorithms;
(2) greater clarity concerning the relative importance of different sources of noise;
(3) greater insight into the nature of noise and information and how these may be explicated.

The approach involves specifying a complete probabilistic model for the attributes used in the example description and the class. This will be referred to as an artificial universe. The class model of the universe will declare relationships between descriptions and class distributions—as distinct from individual classes. The class model can be represented in many different forms from an exhaustive table to a comparatively small set of very general rules.

An artificial universe can be used to generate examples for use by any of the types of learning algorithm mentioned above and, after learning has taken place, can provide a true indication of the performance of a learned concept description. Since the latter is a random variable its expected value behaviour can be approximated from a series of trials involving generated examples.

Noise will be modelled using the class distributions in the class model. The amount of noise will be manipulated principally using the majorisation relation which is concerned with the relative degree of inequality amongst elements in a real vector, in this case a vector of probabilities constituting a distribution.

Although what is proposed here is merely an extension or elaboration of method (2) above it will be referred to as the *method of artificial universes*. Specifying a complete probability model for the generation of data is hardly a new idea. Many of the artificial domains in the literature such as the LED domain [5] are such models. General-purpose generators offering a limited noise modelling capability have also been developed; see, for example Lounis and Bisson [18]. In essence the contribution here is to provide a simple yet general means of

prescribing a particular amount of noise and in a way which is largely independent of its physical source.

The ideas in this paper were outlined by Hickey [14] where the use of the method was illustrated through the construction of a small universe and subsequent experimentation using ID3 on generated examples. The paper concentrates on the underlying theory although a similar set of experiments to those in [14] is performed but with a much larger number of trials.

1.2. Plan of the paper

The definition of an artificial universe and its class model are given in Section 2 and a running example is introduced. The representation of the class model is also discussed. The modelling of different types of noise is addressed in Section 3. Two types of irrelevant attribute (pure noise and redundant) are defined.

The role of majorisation in explicating noise and its relationship to other information-theoretic ideas are reviewed in Section 4. This is applied to produce information statistics for a universe.

In Section 5 the means of assessing the performance of a deterministic classifier, acquired through learning, is discussed. Experimental results obtained from using ID3 on data generated from universes with varying degrees of noise are reported.

2. Artificial universes

An object or situation to be classified is described using attributes (referred to as the description or condition attributes) which will be labelled $a, b, c, \ldots$. These attributes may be discrete (usually finite) or continuous. The description attribute set together with the values that attributes may take is called the *description schema*. A vector of values $(a_1, b_1, \ldots)$ where there is one value from each attribute in the schema is called a *description vector*. More generally a partially instantiated description vector is called a *condition* or *complex*. The classes that may be assigned to an object or situation will be labelled *class*1, *class*2, ... and will be regarded as values of an attribute called *class* (the class attribute).

An artificial universe together with the notion of a class model and its representation are then defined as follows:

Definition 2.1. An (*artificial*) *universe* consists of a description schema, a class attribute and the joint distribution of the class and description attributes.

Definition 2.2. The function which maps a description vector to the distribution of the class attribute conditional on that vector is called the *class model* of the universe.

Definition 2.3. Any statement specifying the class model is said to be a *representation* of the class model.

The class model is analogous to a statistical model, such as is used in regression analysis: *class* is treated as the dependent variable with the description schema providing the independent variables. Likewise the class distributions play the role that error distributions play in a statistical model. It is important to note that it is the *observed* schema vectors and class and their probabilistic relationship that are being modelled. The artificial universe makes no statement about whether, for example, the class associated with a particular description is the correct one. This point will be further elaborated in the next section when the physical sources of noise are discussed.

A representation is usually a set of rules of the form:

if ⟨complex⟩ then ⟨*class* distribution⟩

where the complexes form a partition of the set of description vectors. The class model and joint distribution of the description attributes are sufficient to completely define the universe.

To illustrate these ideas a small universe, called universe 1, will be built. This is a variation on the universe defined in [14]: it has the same description schema and joint distribution of the description attributes however the number of classes has been increased from three to eight. The schema involving description attributes a, b, c, d, and e is:

a:	a_1, a_2
b:	b_1, b_2, b_3, b_4
c:	c_1, c_2, c_3
d:	d_1, d_2, d_3
e:	e_1, e_2
class:	$1, 2, 3, 4, 5, 6, 7, 8$

The rule set specifying the class model is shown in Table 1. Note that d and e do not appear. The joint distribution of the description attributes is provided in Table 2. Here (a, b), c and e are defined to be mutually independent; d is dependent on a and c. The probability of any description occurring is then obtained as:

$$\begin{aligned} &P(a = v_1, b = v_2, c = v_3, d = v_4, e = v_5) \\ &\quad = P(a = v_1, b = v_2) \cdot P(c = v_3) \cdot P(d = v_4 \mid a = v_1, c = v_3) \cdot P(e = v_5) \,. \end{aligned} \tag{1}$$

The right-hand side of (1) is an example of a *generating expression* for a description vector.

The rule set in Table 1 offers a fairly compressed representation involving general complexes. Clearly if all attributes in the description schema are finite discrete it is possible to tabulate the mapping between individual description

Table 1
A representation of the class model for universe 1

Rule number	Description	Class distribution
1	$a = a_1$ and $b = b_1$	(0.5, 0.5, 0, 0, 0, 0, 0, 0)
2	$a = a_2$ and $b = b_1$ and $c = c_1$	(0.55, 0, 0, 0.45, 0, 0, 0, 0)
3	$a = a_2$ and $b = b_1$ and $c = c_2$	(0, 0, 0.6, 0, 0, 0, 0, 0.4)
4	$a = a_2$ and $b = b_1$ and $c = c_3$	(0, 0, 0, 0, 0, 0, 0.4, 0.6)
5	$b = b_2$ and $c = c_1$	(0, 0, 0, 0.3, 0, 0, 0.7, 0)
6	$b = b_2$ and $c = c_2$	(0, 0, 0, 0, 0, 0.55, 0.45, 0)
7	$b = b_2$ and $c = c_3$	(0, 0, 0.4, 0, 0.6, 0, 0, 0)
8	$a = a_1$ and $b = b_3$	(0.3, 0.7, 0, 0, 0, 0, 0, 0)
9	$a = a_2$ and $b = b_3$ and $c = c_1$	(0, 0, 0, 0.6, 0, 0.4, 0, 0)
10	$a = a_2$ and $b = b_3$ and $c = c_2$	(0, 0, 0.6, 0, 0, 0.4, 0, 0)
11	$a = a_2$ and $b = b_3$ and $c = c_3$	(0, 0, 0, 0, 0, 1, 0, 0)
12	$b = b_4$ and $c = c_1$	(0, 0, 0, 0.35, 0, 0.65, 0, 0)
13	$a = a_1$ and $b = b_4$ and $c = c_2$	(0, 0, 0, 0, 0.5, 0, 0, 0.5)
14	$a = a_2$ and $b = b_4$ and $c = c_2$	(0.4, 0, 0, 0.6, 0, 0, 0, 0)
15	$a = a_1$ and $b = b_4$ and $c = c_3$	(0, 0.6, 0, 0, 0, 0.4, 0, 0)
16	$a = a_2$ and $b = b_4$ and $c = c_3$	(0, 0, 0, 0, 0, 0.5, 0.5, 0)

vectors and their class distributions; this will be referred to as the *enumerated* representation.

One of the motivations for this work is to simulate in an artificial setting some of the complications of the real world. Making the description attributes dependent on one another in some way is one aspect of this. In other approaches to artificial data modelling mutual independence is often assumed.

Table 2
Probabilities for the joint distribution of the description attributes in universe 1; (a, b), c and e are mutually independent; d is dependent on (a, c)

		b			
		b_1	b_2	b_3	b_4
a	a_1	0.05	0.02	0.4	0
	a_2	0.15	0.03	0.25	0.1

c			e	
c_1	c_2	c_3	e_1	e_2
0.6	0.05	0.35	0.3	0.7

		d		
		d_1	d_2	d_3
Conditional values of a and c	a_1 and (c_1 or c_3)	0.4	0.2	0.4
	a_1 and c_2	0.2	0.5	0.3
	a_2 and c_1	0.1	0.2	0.7
	a_2 and c_2	0.25	0.7	0.05
	a_2 and c_3	0.3	0.3	0.4

Once a universe is specified as above, examples can be generated from it as follows:

(1) Generate a description vector from the joint distribution of the description attributes using the generating expression.
(2) Look up the generated vector in the class model to find the class distribution.
(3) Generate a class from the class distribution.
(4) The description vector and the class thus obtained constitute the example.

It may appear that the use of a rule set to define the class model is somehow biased towards learning algorithms which induce high-level rules. This is not the case however. It is just a convenient way to produce a specification for a universe. Moreover the choice of one representation rather than another, for example the use of general rules versus the enumerated representation, is of no consequence for the generation of examples.

The marginal, i.e. unconditional, distribution of class (referred to as the *default* distribution) can be obtained from the universe specification. For universe 1 this is:

$$(0.1215, 0.1510, 0.0400, 0.2115, 0.0445, 0.2675, 0.1270, 0.0370)\,. \tag{2}$$

The majority class, i.e. most probable, in the default distribution is called the *default* class. Here it is class 6 with a 26.75% chance of occurring.

2.1. Sub-universes

There are universes within universes. These will be called *sub-universes*. If several attributes in the example description are removed this results in a sub-universe with a reduced description schema. This will be referred to as the *marginal* sub-universe w.r.t. the remaining description attributes. There are also *conditional* sub-universes relative to a given complex having one or more attributes uninstantiated. These latter attributes can be used to formulate a sub-inverse conditional on the complex. Such a complex may involve partial instantiation of one or more attributes, e.g. $b = b_1$ or b_2. In such a case the partially instantiated attributes are included in the schema for the sub-universe with their values being restricted to those specified in the instantiation.

Class models and generating expressions can be constructed for sub-universes from the specification of the universe itself.

3. Noise and the class model

In learning from examples, noise is anything which obscures the relationship between description and class. There are three major physical sources of noise:

(1) insufficiency of the description schema,
(2) corruption of attribute values in the example description,
(3) erroneous classification of training examples.

If there are non-degenerate distributions appearing in the class model this can be attributed to the presence of one or more of these noise sources. The user of the artificial universe method can adopt one of two attitudes to the origin of noise. Firstly she can ignore it, i.e. say nothing about it, and just investigate how the noise affects learning, perhaps varying the degree of noise in the manner to be described below. Secondly she can declare that the noise originates from some of the sources above. It would be difficult, though, to specify how much came from each source without building a more elaborate model. For the most part, however, the origin of noise does not affect the analysis of learning and the first approach is often all that is required. To see this consider the sources separately and how they might be catered for within the class model.

The class distributions in the class model describe the uncertainty in the association between observed description vector and the class it is assigned and can therefore be taken to model the first of these, the insufficiency of the description schema (what the statisticians call residual error).

These distributions can also be used to model—or rather eliminate the separate need for—attribute noise. To explicitly allow for corruption of attribute values a distinction can be made between true attributes (which are not observed) and actual attributes which are observed. Usually, as, for example, in the LED domain an actual attribute takes values from the same set as its corresponding true attribute according to a given probability distribution (called the *corruption distribution*) which, in the most general case, is conditional upon the true description vector.

The definition of a universe can be augmented to provide the joint distribution of both *true* and *actual* attributes together with the class attribute. From this can be obtained a true and an actual class model (class distributions conditional on the true and actual attributes respectively).

If it is assumed that the mechanism which corrupts example descriptions is at work both when examples are being obtained for learning and also when the learned concept description is being tested (on fresh unclassified examples) then the true class model has no role to play: the actual class model can be used to generate examples and to assess the learned concept.

It may be argued that the purpose of introducing separate true and actual attributes is that it allows the affect of attribute noise to be determined. Schaffer [28], for example, increases the probability of corruption in the LED domain and investigates the consequences for learning. In general, though, it is difficult to state the overall effect on noise resulting from a particular set of corrupting distributions. It does not follow, paradoxical though it may seem, that corrupting the attribute distributions necessarily increases noise. In fact it is possible to define a universe in which the true class model has distributions which are non-degenerate (so that class is uncertain) whereas the actual class model has each of its distributions degenerate at a single class!

Errors in example classification do, however, require more elaborate modelling. Here the class distributions reflect teacher miss-classification (and perhaps other sources too), so that some of the classes present in the examples are actually the wrong class. After learning, however, the acquired concept description is used

to make classifications. Thus the classification error mechanism is not present. Evaluation of learning would use the true class distributions—not those used to generate the examples. Two universes are therefore required: an actual universe (used for example generation) and a true universe. It appears from the literature that there is little awareness of this aspect of classification noise since the data for testing is usually generated according to the same prescription as that for learning.

3.1. Irrelevant attributes

In addition to noise there are often "irrelevant" attributes present in the schema, i.e. those that contribute little or nothing to classification. There are two cases to distinguish here, namely those of redundancy and pure noise.

Definition 3.1. An attribute having the property that there is a representation of the class model in which it does not appear is said to be *redundant* relative to those attributes which do appear.

Definition 3.2. An attribute, a, which satisfies

$$P(class = c \mid S, A) = P(class = c \mid S)$$

for all c, S and A, where S is a subset of the vector description space of all the description attributes excluding a and A is a subset of the values taken by a, is called a *pure noise* attribute. A pure noise attribute is said to be *uninformative* (about class); all other attributes are said to be *informative*. A universe in which at least one attribute is informative is said to be *informative*, otherwise it is said to be *uninformative*.

The redundancy of an attribute may be relative to a particular subset of attributes, for example, when it is functionally dependent on those in the subset. Pure noise implies redundancy but not vice versa: the omission of an attribute from a representation of the universe does not guarantee that it is pure noise as will be seen below.

In universe 1, d is redundant (it is not instantiated in any of the rules in the given representation of the class model) but it is not pure noise. The class distribution for the condition $b = b_3$ and $c = c_1$ is:

$$(0.113, 0.263, 0, 0.375, 0, 0.250, 0, 0)\,,$$

indicating that class 4 is the most likely. If, however, $d = d_1$ is also instantiated the class distribution changes to

$$(0.212, 0.494, 0, 0.176, 0, 0.118, 0, 0)\,,$$

which makes class 2 the most likely.

Some authors, for example Pazzani and Sarrett [22], use the term "irrelevant" to mean redundancy as defined here. Since redundant attributes may contain useful information about class the use of the term "irrelevant" is unfortunate. Only a pure noise attribute is truly irrelevant in the sense that there are no circumstances in which knowledge of its value can influence classification.

The definition for pure noise appears similar to that of conditional independence of class and the pure noise attribute given all the other description attributes—abbreviated in this discussion to "conditional independence" (see Pearl [23] for a general discussion of conditional independence). In the latter definition, however, the conditioning description vector must be fully instantiated whereas that is not the case for pure noise. Thus pure noise implies conditional independence but not conversely. The specification of a class model in which an attribute does not appear, as was made for universe 1, is equivalent to the assertion of conditional independence for that attribute (so that redundancy is essentially conditional independence).

The implementation of a pure noise attribute can be achieved with the further requirement (in addition to omission from the concept description) that it be independent of the vector of all the other description attributes.[1] In universe 1, e is pure noise. Additional pure noise attributes can be added very easily to a universe already defined simply by specifying a marginal distribution for each one and adding a corresponding term to the generating expression for a description vector.

By taking S to be the empty set in Definition 3.2 it follows that class and a pure noise attribute are independent.

Clearly a universe is uninformative if and only if all the class distributions in the class model are identical. In this case they are all equal to the default distribution for class. In any representation of the class model, the complexes that appear in the rules define uninformative conditional sub-inverses. The notion of informativeness of an attribute is quite separate from that of noise in the universe class distributions. If an attribute is informative then there will be occasions when knowledge of its value will be of some use to classification regardless of how much noise is in the universe although, of course, the extent of the latter limits just how informative an attribute can be.

It is also possible for a universe with no noise in the class distributions to have several pure noise attributes. The assessment of how informative individual attributes are will be pursued further in the next section when noise in the universe is further explicated using information-theoretic concepts.

4. Assessing the degree of noise in the class distributions

The question remains as to how to manipulate the class distribution to achieve a required amount of noise. In statistical modelling of residual error, it is common practice to employ normal distributions with the variance parameter indicating the

[1] Let a' denote the vector of all description attributes excluding a. The assertion: "independence of a and a' together with the conditional independence of *class* and a given a' implies that a is pure noise" follows easily from elementary probability theory. This sufficient condition for pure noise is also readily seen to be equivalent to the independence of $(class, a')$ and a.

extent of noise. For class distributions where there are at most a small number of discrete (often nominal) classes this is not appropriate. What is needed is a means of explicating the degree of noise using only the probabilities in the class distribution and not the classes themselves. For example, in universe 1 the class distribution for the first rule for the class model (Table 1) is (0.5, 0.5, 0, 0, 0, 0, 0, 0). Is there more noise in this distribution than that of the next rule which is (0.55, 0, 0, 0.45, 0, 0, 0, 0)? Such a means of comparison is provided by the majorisation relation applied to probability vectors. Majorisation underpins much of the theory of measurement of information and uncertainty.

4.1. Majorisation and noise

Measures of uncertainty such as Shannon's entropy function

$$entropy(P) = -\sum_{i=1}^{n} p_i \ln p_i ,$$

where $P = (p_1, \ldots, p_n)$ and $\sum_{i=1}^{n} p_i = 1$ is the probability distribution on the n possible classes, provide an assessment of noise based on probabilities only.

Entropy, though, is only one of many possible measures of uncertainty. The well-known additivity of information property of entropy which renders it unique up to positive multiples (Shannon and Weaver [30]) and which has meaning in communication theory is not relevant here. Another popular measure is provided by the Gini index of diversity:

$$gini(P) = 1 - \sum_{i=1}^{n} p_i^2 .$$

Information and uncertainty are usually regarded as dual, i.e. the greater the information the lesser the uncertainty, and will be treated as such here. Noise will be identified with uncertainty.

Uncertainty measures are usually defined to be strictly Schur-concave (see Hickey [11, 12]) that is they respect the pre-ordering afforded by the majorisation relation between probability distributions.

Majorisation as developed in the mathematical theory of inequalities provides a pre-ordering amongst vectors of real numbers having the same total of their elements (but not necessarily having the same number of elements) interpreted as: "the elements in this vector are less equal than those in that vector". It is a pre-ordering rather than a partial ordering because it lacks antisymmetry: vectors which differ only in permutation of elements majorise each other but are not necessarily identical.

Applied to discrete probability distributions or, indeed, relative frequency distributions, strict majorisation explicates the notion "less noisy than", i.e. the distribution whose probabilities are less equal is the less noisy or, equivalently, the more informative. Here the background to majorisation will be outlined for probability distributions only. For the general theory of majorisation see [19].

Majorisation can be defined using the notion of an *equalising transfer* which involves re-distributing some (larger) probability from one event to another having a smaller probability (which may be zero) in such a way as to render the two probabilities involved more equal. Formally, if in $P = (p_1, \ldots, p_n)$, there are i and j such that $p_i > p_j$ then some of the excess in p_i over p_j is transferred to p_j such that a new distribution, P', is created with all except the ith and jth probabilities unchanged and the latter being replaced by p'_i and p'_j where

$$p'_i = cp_i + (1-c)p_j\,, \qquad p'_j = (1-c)p_i + cp_j \tag{3}$$

for some c, $0 \leqslant c \leqslant 1$. This leads to a definition of majorisation:

Definition 4.1. If discrete distributions P and Q are such that Q can be transformed into P by a finite number of equalising transfers then Q is said to *majorise* P (or P *de-majorises* Q) and this is written $P \preccurlyeq Q$. If $P \preccurlyeq Q$ and Q is not a permutation of P then Q is said to *strictly majorise* P (or P *strictly de-majorises* Q) and this is written $P \prec Q$.

It follows that majorisation is transitive. The case $c = 0$ corresponds to swapping the ith and jth probabilities. Thus if Q is a permutation of P then $P \preccurlyeq Q$ and $Q \preccurlyeq P$. The expression in (3) can be reformulated as

$$P = QS\,, \tag{4}$$

where S is the doubly stochastic matrix with

$$s_{kk} = 1\,, \quad k \neq i, j\,,$$
$$s_{ii} = c\,, \qquad s_{ji} = 1-c\,, \qquad s_{ij} = 1-c\,, \qquad s_{jj} = c\,,$$

and all other elements zero.

It is shown in [19] that $P \preccurlyeq Q$ if and only if (4) holds for some (more complex) doubly stochastic matrix S.

If $P \prec Q$ then P will be regarded as *more uncertain* or *noisier* or *less informative* than Q. With the majorisation interpretation of noise, low noise is synonymous with high concentration of probability on a small number of events whereas high noise corresponds to a spread of probability across a large number of events.

The first two class distributions in universe 1 are related by majorisation:

$$(0.5, 0.5, 0, 0, 0, 0, 0, 0) \preccurlyeq (0.55, 0, 0, 0.45, 0, 0, 0, 0)$$

since a single transfer of 0.05 from 0.55 to 0.45 produces a permutation of the left-hand side.

A more useful reformulation of the definition of majorisation can be given in terms of partial sums of the sorted probabilities in each vector (see [19]). Let $(p_{[1]}, \ldots, p_{[n]})$ denote the decreasing rearrangement of P, i.e. $p_{[1]} \geqslant \cdots \geqslant p_{[n]}$, then $P \preccurlyeq Q$ holds if and only if

$$q_{[1]} + \cdots + q_{[k]} \geqslant p_{[1]} + \cdots + p_{[k]} \tag{5}$$

for all k, $1 \leqslant k \leqslant n$. This is the most suitable form for computation.

The following useful properties of majorisation ($P \preccurlyeq Q$) are immediate consequences of (5):

(1) $\max(P) \leq \max(Q)$ where max is the maximum function.
(2) The size of the support of P, i.e. the number of non-zero probabilities, must be at least as great as that of Q.
(3) The smallest non-zero probability in P must be at least as large as that in Q.

It can also be seen from (5) that majorisation is defined as "modulo zero probabilities": distributions $(0.3, 0.7)$ and $(0.3, 0.7, 0, 0, 0)$ are equally noisy. (Since, in this application, noise will usually be modelled over a fixed number of classes this point is of little consequence.) Amongst n classes the uniform distribution $(1/n, \ldots, 1, n)$ is the most noisy and is majorised by every other distribution on n classes. At the other end of the scale any degenerate distribution is least noisy in the sense that it majorises every other distribution.

If classes in a universe are combined, the resulting class distribution in the class model majorises the original since combining probabilities amounts to the reverse of an equality transfer, i.e. an *inequality* transfer.

A concrete example of the use of majorisation is provided by the commonly used device of introducing noise through inversion of classes with a known probability. This is the mechanism in the classification noise process of Angluin and Laird [3]. In a two-class problem suppose $Q = (q_1, q_2)$ is a class distribution. Suppose further that although a class is generated according to Q it is then inverted with probability α. This results in a new class distribution $P = (p_1, p_2)$ where

$$p_1 = (1-\alpha)q_1 + \alpha q_2 , \qquad p_2 = \alpha q_1 + (1-\alpha)q_2 , \tag{6}$$

which is just (3). Thus $P \preccurlyeq Q$. More generally, in an n-class problem, the inversion probability, α, can be split equally amongst the remaining $n-1$ classes. This generalizes (6) to (4) where S has diagonal elements $(1-\alpha)$ and off-diagonal elements $\alpha/(n-1)$ and so again $P \preccurlyeq Q$.

4.2. *Increasing noise in universe* 1

To illustrate the use of majorisation, universe 1 will be made noisier (thereby creating universe 2) by de-majorising each class distribution. This will be achieved by leaving the majority class probability in each distribution unchanged and spreading the remaining probability more evenly over the other classes instead of it being concentrated on a single class as is the case in universe 1. This is the strategy that was employed in [14]. The joint distribution of the description attributes will not be altered.

The new class distributions are shown in Table 3 alongside those for universe 1. Notice that only the distribution for rule 11 is unchanged—there is no residual probability to re-distribute here. In the other cases the complement of the majority class probability has been spread fairly evenly across most of the other classes.

Table 3
Class distributions for universes 1 and 2

Rule number	Universe 1	Universe 2
1	(0.5, 0.5, 0, 0, 0, 0, 0, 0)	(0.5, 0.1, 0.1, 0, 0, 0.1, 0.1, 0.1)
2	(0.55, 0, 0, 0.45, 0, 0, 0, 0)	(0.55, 0.05, 0.1, 0.1, 0.05, 0, 0.1, 0.05)
3	(0, 0, 0.6, 0, 0, 0, 0, 0.4)	(0.1, 0.1, 0.6, 0, 0.05, 0.05, 0.05, 0.05)
4	(0, 0, 0, 0, 0, 0, 0.4, 0.6)	(0.1, 0.1, 0.05, 0.05, 0, 0.05, 0.05, 0.6)
5	(0, 0, 0, 0.3, 0, 0, 0.7, 0)	(0.05, 0.05, 0.05, 0.05, 0, 0.05, 0.7, 0.05)
6	(0, 0, 0, 0, 0, 0.55, 0.45, 0)	(0, 0.05, 0.05, 0.05, 0.05, 0.55, 0.15, 0.1)
7	(0, 0, 0.4, 0, 0.6, 0, 0, 0)	(0.1, 0.05, 0.05, 0.1, 0.6, 0, 0.05, 0.05)
8	(0.3, 0.7, 0, 0, 0, 0, 0, 0)	(0.1, 0.7, 0.05, 0.05, 0, 0.05, 0.05, 0)
9	(0, 0, 0, 0.6, 0, 0.4, 0, 0)	(0.1, 0.05, 0.05, 0.6, 0, 0.1, 0.05, 0.05)
10	(0, 0, 0.6, 0, 0, 0.4, 0, 0)	(0.05, 0.05, 0.6, 0.05, 0.1, 0.05, 0.05, 0.05)
11	(0, 0, 0, 0, 0, 1, 0, 0)	(0, 0, 0, 0, 0, 1, 0, 0)
12	(0, 0, 0, 0.35, 0, 0.65, 0, 0)	(0.05, 0.05, 0, 0.05, 0, 0.65, 0.1, 0.1)
13	(0, 0, 0, 0, 0.5, 0, 0, 0.5)	(0.1, 0.1, 0.05, 0.05, 0.5, 0.1, 0.05, 0.05)
14	(0.4, 0, 0, 0.6, 0, 0, 0, 0)	(0.1, 0.05, 0.05, 0.6, 0.05, 0.05, 0.05, 0.05)
15	(0, 0.6, 0, 0, 0, 0.4, 0, 0)	(0.05, 0.6, 0.05, 0.05, 0.05, 0.05, 0.05, 0.1)
16	(0, 0, 0, 0, 0, 0.5, 0.5, 0)	(0.1, 0.1, 0.1, 0.1, 0, 0.05, 0.5, 0.05)

Universe 2 has a different default class distribution. It is

$$(0.1364, 0.1699, 0.0594, 0.1378, 0.0531, 0.2124, 0.1534, 0.0778)\,, \tag{7}$$

whereas that for universe 1, given in (2), is:

$$(0.1215, 0.1510, 0.0400, 0.2115, 0.0445, 0.2675, 0.1270, 0.0370)\,.$$

The default class is still class 6 but its probability has dropped by over 5%. It is not generally the case, though, that increasing noise will reduce the probability of the default class. In the next section, universe 2 will be altered slightly to produce an increase in default class probability over that of universe 1.

4.3. Measures of information and uncertainty

As a step towards defining these measures a function which is monotone w.r.t. majorisation is needed:

Definition 4.2. A continuous real-valued function, ϕ, on the space of finite discrete probability distributions is *Schur-convex* if $\phi(P) \leqslant \phi(Q)$ whenever $P \preccurlyeq Q$. If $\phi(P) < \phi(Q)$ whenever $P \prec Q$ then ϕ is *strictly Schur-convex*. If ϕ is (strictly) Schur-convex then $-\phi$ is (*strictly*) *Schur-concave*.

Schur-convex functions are necessarily symmetric. Ordinary convexity together with symmetry implies Schur-convexity (see [19]). Although not essential in an arbitrary Schur-convex function, convexity has an important meaning for information measures as it guarantees (and is in fact equivalent to) the property, familiar from entropy, that expected conditional information is always at least as

great as that of an unconditional distribution. Expressed in terms of random variables X and Y this property is:

$$\phi(X \mid Y) \geqslant \phi(X) . \tag{8}$$

If ϕ in (8) is strictly convex then equality occurs if and only if X and Y are independent. Accordingly, the following definition is made:

Definition 4.3. A real-valued continuous function, ϕ, on the space of discrete probability distributions is a *measure of information* (*uncertainty*) if it is symmetric and strictly convex (concave). If the function lacks strictness, the measure is said to be *weak*.

Entropy and the Gini index are both symmetric and strictly concave and are therefore measures of uncertainty. The maximum function, $\max(P)$, is a weak information measure: its convexity is not strict. The commonly-used error or miss-classification function, $error(P) = 1 - \max(P)$ is a weak measure of uncertainty.

The development given above can be extended to continuous distributions where the idea of majorisation carries over in a natural way (see [13]). The observation that strict convexity was desirable in a measure of information was also made by Breiman et al. [5] in the context of selection measures for the CART learning algorithm.

All information measures render the same ordering of informativeness between two distributions which are related by majorisation (in fact majorisation can be defined in terms of this property [19]). On the other hand if two distributions are not related by majorisation then it is always possible to find two information measures which will order them differently. Thus majorisation between distributions is a stronger condition than "has greater information" as assessed by a real-valued measure. The latter inequality, without majorisation holding, could be just an artefact of the measure used and not indicative of any material difference in information. It is for this reason that it is generally better to manipulate the amount of noise using majorisation itself rather than, say, by increasing entropy.

4.4. Information in a universe

Measures of information are useful, however, for providing overall summaries of the information content of a universe, that is, information about class.

Definition 4.4. The *rule information* in a universe with respect to an information measure is its expectation over the class distributions in the class model of the universe. The *default information* is that of the default distribution with respect to the measure. The *information gain* is:

$$|\text{rule information} - \text{default information}|.$$

A dual definition holds for uncertainty measures. By virtue of the strict convexity in Definition 4.3 the information gain is zero if and only if the universe is uninformative.

The rule information assesses the contribution of the attributes to identifying class. The benefit of the attributes is, however, relative to the default information as provided by the information gain. Care must be taken when comparing universes in terms of their noise content using rule information. Suppose in a universe all class distributions are permutations of one distribution, P, but not all identical. The uninformative universe with all class distributions equal to P has the same rule information but zero information gain.

When the information measure is the max function, the default and rule informations have interpretations as classification rates:

Definition 4.5. When the information measure is max, the default and rule informations are called the *default* and *universe classification rates* (*DCR* and *UCR*) respectively.

Increasing the noise in a class distribution in any representation of the universe can never increase the rule information (and will decrease it unless the information measure is weak). In particular the UCR can only decrease. The information gain, though, may increase or decrease.

Table 4 shows information statistics, using entropy and max, for universes 1 and 2. It is clear that the description attributes facilitate the identification of class as there is substantial information over the default. Also shown is the information about class provided by each attribute on its own. For both universes, b is the most informative attribute as judged by both entropy and max. No single attribute, however, can classify satisfactorily on its own. Because universe 2 was

Table 4
Information statistics relating to entropy and max for universes 1 and 2; ranks of each attribute are shown in brackets

Information	Universe 1				Universe 2			
	Entropy		Max		Entropy		Max	
Default	1.8742		0.2675		1.9847		0.2124	
Rules	0.5976		0.6540		1.1849		0.6540	
Gain	1.2766		0.3865		0.7998		0.4416	
Attributes:								
a	1.6728	(3)	0.3622	(2)	1.8853	(2)	0.2922	(2)
b	1.3936	(1)	0.4250	(1)	1.6761	(1)	0.3649	(1)
c	1.6420	(2)	0.3395	(3)	1.8921	(3)	0.2316	(3)
d	1.8526	(4)	0.2675	(4)	1.9728	(4)	0.2151	(4)
e	1.8742	(5)	0.2675	(5)	1.9847	(5)	0.2124	(5)

constructed from universe 1 by leaving the majority probability in each class distribution unaltered, its UCR of 65.4% is the same as that for universe 1.

Even a small universe such as this can provide a substantial task for learning algorithms. Holte [15] has observed that many of the real data sets in the "Machine Learning Repository" [21] appear to possess a single attribute which is very informative with the others contributing little. With the method of artificial universes it is comparatively simple to build a small manageable universe and yet in which several attributes are needed for effective classification.

5. Evaluating learning

The result of concept learning from examples will usually be a mechanism for classification, i.e. a means of deciding, given *any* description vector, what the corresponding class is. Some learning algorithms may produce a non-deterministic classification, i.e. offer a choice of possible classes with an indication of the degree of uncertainty attached to each. For example a trained neural network may give the strengths of its output units or a decision tree may have relative frequencies for observed classes in its leaves. Some rule induction algorithms such as CN2 (Clark and Boswell [7]) induce overlapping rules. The user of such a system can decide on a rule for selection of a particular class thus creating a deterministic classifier. It will be assumed here that classifiers produce a definite class on each occasion.

Definition 5.1. A (*deterministic*) *classifier* is a mapping from the set of description vectors to the set of classes. A statement of this mapping is called a *representation* of the classifier.

The universe itself provides a best classifier, i.e. one which has the optimal probability of correct classification.

Definition 5.2. A *best classifier* associated with a universe maps each description vector to a *majority class* in its associated class distribution in the class model, i.e. one which has maximum probability in this distribution. A *default classifier* for a universe assigns each description vector a majority class in the default distribution.

Because of ties in the class distribution, the best and the default classifiers may not be unique.

Concept learning can be characterised as estimation of the best classifier from examples. With some forms of learning from examples, though, such as neural networks and instance-based learning, the classifier obtained is implicit rather than explicit. To obtain an extensional form of the classifier all possible description vectors must be supplied to it and the resulting classifications noted. For small universes with discrete attributes this does not present a problem. For

very large cases or if continuous attributes are involved this would present difficulties. One possibility here is to estimate the classifier using a second stage of learning. A large sample of description vectors are supplied to the classifier for classification. The resulting (noiseless) example set is then passed to a learning algorithm which generates rules. The rule set obtained is an approximation to the true classifier.

Evaluating an extensionally available classifier is straightforward. The classifier provides a set of deterministic rules of the form

if ⟨complex⟩ then ⟨class⟩

from which, using the universe specification, the classification rate can be calculated.

Definition 5.3. The probability that an induced classification rule correctly classifies a randomly selected example from the universe which satisfies its condition is called the *actual classification probability* of the rule. The expected value of the actual classification probability in a classifier is its *actual classification rate* (*ACR*).

The ACR of a classifier can never exceed the UCR (Definition 4.5) of the universe. It can, though, if the classifier is sufficiently bad, be less than the DCR, which is the rate for the default classifier. The UCR is the classification rate for the best classifier.

An ACR associated with a classifier learned from data randomly generated from a universe is a random variable. The expected ACR over example sets of a particular size is a useful indicator of the effectiveness of the learning algorithm. The relationship between this expected ACR and size of example set is the *learning curve*. The expected ACR for an example set size, while difficult to compute for algorithms such as ID3, can be estimated from a number of trials of learning.

5.1. Experiments with ID3 on universes 1 *and* 2

To examine the effect of increased noise on learning with ID3, a number of trials were performed using universes 1 and 2. Each trial consisted of generating an example set of a particular size, inducing a tree with ID3, producing the deterministic classifier obtained by adopting the majority class in each leaf of the tree (a mild form of pruning) and computing, from the probabilities in the universe, the ACR of the classifier.

To estimate expected ACRs and the learning curve, a number of trials were carried out for a range of example set sizes from 5 to 5000. Because of the high variability in ACR obtained from small sizes, a large number of replications were carried out for these (4000 for size 5) with the number of replications decreasing as the size increased (down to 25 for a size of 5000).

The results of these experiments are shown in Table 5 and the learning curves

Table 5
Estimates of expected ACR, expressed as a percentage, for rules sets induced by ID3 from example sets generated by universes 1 and 2; estimated standard errors are also given as percentages

Size	No. of trials	Universe 1	Universe 2
		Est. expected ACR (Est. standard error) DCR = 26.75%, UCR = 65.4%	Est. expected ACR (Est. standard error) DCR = 21.24%, UCR = 65.4%
5	4000	29.2 (0.12)	23.5 (0.11)
8	4000	34.8 (0.13)	27.8 (0.13)
10	2000	37.5 (0.18)	30.0 (0.18)
12	2000	39.8 (0.17)	31.7 (0.18)
15	2000	42.1 (0.16)	33.6 (0.18)
20	1000	45.1 (0.20)	35.7 (0.24)
30	1000	48.2 (0.16)	39.1 (0.21)
50	500	51.0 (0.16)	42.5 (0.24)
75	500	52.4 (0.12)	44.4 (0.18)
100	200	53.6 (0.16)	46.6 (0.25)
200	100	56.0 (0.19)	51.2 (0.29)
300	100	57.3 (0.20)	53.8 (0.23)
500	100	59.1 (0.13)	57.7 (0.14)
750	100	60.7 (0.09)	60.4 (0.11)
1000	100	61.6 (0.10)	61.6 (0.08)
1250	100	62.2 (0.09)	62.5 (0.07)
1500	50	62.4 (0.11)	63.1 (0.07)
2000	50	63.1 (0.09)	63.9 (0.06)
3000	50	63.9 (0.08)	64.6 (0.04)
5000	25	64.5 (0.06)	64.9 (0.04)

up to size 100 are displayed in Fig. 1. The expected ACR for universe 1 is consistently above that for universe 2 until about size 750 showing that the increased noise in universe 2 presents considerable difficulties for ID3. For larger example set sizes, universe 2 appears to be slightly better with the differences being statistically significant for sizes of 1500 and above. The explanation for this

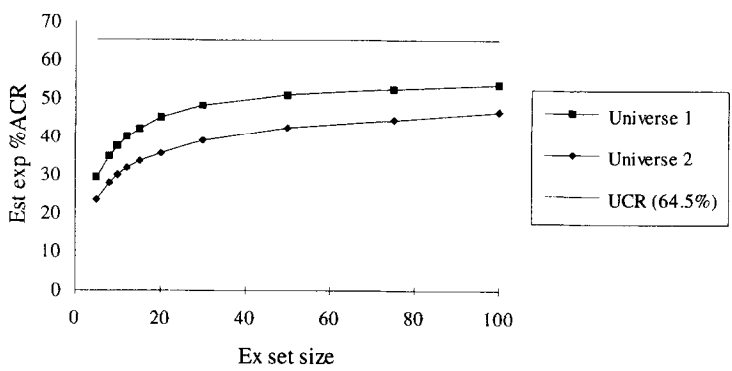


Fig. 1. ID3 learning curves for example sets of up to size 100 from universes 1 and 2.

Table 6
Class distributions for universe 3

Rule number	Class distribution
1	(0.5, 0.1, 0.05, 0, 0, 0.2, 0.05, 0.1)
2	(0.55, 0.05, 0.1, 0.05, 0.05, 0.15, 0, 0.05)
3	(0.05, 0.05, 0.6, 0, 0.05, 0.15, 0.05, 0.05)
4	(0.05, 0.05, 0.05, 0.05, 0, 0.15, 0.05, 0.6)
5	(0.05, 0.05, 0.05, 0.05, 0, 0.05, 0.7, 0.05)
6	(0, 0.05, 0.05, 0.05, 0.05, 0.55, 0.15, 0.1)
7	(0.05, 0.05, 0.05, 0, 0.6, 0.15, 0.05, 0.05)
8	(0.05, 0.7, 0.05, 0.05, 0, 0.1, 0.05, 0)
9	(0.05, 0.05, 0.05, 0.6, 0, 0.15, 0.05, 0.05)
10	(0.05, 0.05, 0.6, 0.05, 0.05, 0.1, 0.05, 0.05)
11	(0, 0, 0, 0, 0, 1, 0, 0)
12	(0.05, 0.05, 0, 0.05, 0, 0.65, 0.1, 0.1)
13	(0.1, 0.05, 0.05, 0.05, 0.5, 0.15, 0.05, 0.05)
14	(0.05, 0.05, 0.05, 0.6, 0.05, 0.1, 0.05, 0.05)
15	(0.05, 0.6, 0.05, 0.05, 0.05, 0.1, 0.05, 0.05)
16	(0.05, 0.1, 0.05, 0.05, 0, 0.2, 0.5, 0.05)

may be that the single minority class in universe 1 is emerging on occasions as the majority in the data—something that is unlikely to happen in universe 2 with its small residual probabilities.

It might be argued that the better expected ACRs obtained for universe 1 for smaller sample sizes are due to a superior default classification rate (DCR): from Table 4 the DCR for universe 1 is 26.75% whereas for universe 2 it is 21.24% so that universe 1 has a considerable head start on universe 2. To investigate this, universe 2 was modified slightly to produce a DCR close to that of universe 1. This was achieved by transferring a small amount of probability to the majority class (class 6) in the class distribution of most rules.

The new distributions which, together with the joint distributions of the description attributes, define universe 3 are shown in Table 6. The information statistics for universe 3 are shown in Table 7. The default distribution for universe

Table 7
Information statistics for universe 3; ranks of each attribute are shown in brackets

Information	Entropy	Max
Default	1.9453	0.2705
Rules	1.1692	0.6540
Gain	0.7761	0.3835
Attributes:		
a	1.8391 (2)	0.3293 (2)
b	1.6217 (1)	0.3825 (1)
c	1.8412 (3)	0.2736 (3)
d	1.9326 (4)	0.2705 (4)
e	1.9453 (5)	0.2705 (5)

Table 8
Estimates of expected ACR, expressed as a percentage, for rules sets induced by ID3 from example sets generated by universes 1 and 3; estimated standard errors are also given as percentages

Size	No. of trials	Universe 1	Universe 3
		Est. expected ACR (Est. standard error) DCR = 26.75%, UCR = 65.4%	Est. expected ACR (Est. standard error) DCR = 27.05%, UCR = 65.4%
5	4000	29.2 (0.12)	24.8 (0.12)
8	4000	34.8 (0.13)	29.2 (0.13)
10	2000	37.5 (0.18)	31.3 (0.18)
12	2000	39.8 (0.17)	33.0 (0.18)
15	2000	42.1 (0.16)	35.0 (0.17)
20	1000	45.1 (0.20)	36.7 (0.23)
30	1000	48.2 (0.16)	40.3 (0.19)
50	500	51.0 (0.16)	43.5 (0.22)
75	500	52.4 (0.12)	45.7 (0.19)
100	200	53.6 (0.16)	47.4 (0.22)
200	100	56.0 (0.19)	51.8 (0.24)
300	100	57.3 (0.20)	55.3 (0.16)
500	100	59.1 (0.13)	58.0 (0.14)
750	100	60.7 (0.09)	60.5 (0.10)
1000	100	61.6 (0.10)	62.0 (0.08)
1250	100	62.2 (0.09)	62.7 (0.07)
1500	50	62.4 (0.11)	63.3 (0.08)
2000	50	63.1 (0.09)	64.0 (0.07)
3000	50	63.9 (0.08)	64.6 (0.03)
5000	25	64.5 (0.06)	65.0 (0.02)

3 is

$$(0.1129, 0.1666, 0.0551, 0.1245, 0.0525, 0.2705, 0.1419, 0.0760)$$

and thus the DCR is 27.05%, slightly above that for universe 1 which is 26.75%.

Trials similar to those described above for universes 1 and 2 were carried out to estimate the ID3 learning curve for universe 3. The results are shown in Table 8 (alongside those for universe 1 for comparison). It can be seen that the pattern for universe 3 follows that for universe 2: ACRs are well below those for universe 1 for sample sizes up to 750 and then universe 3 creeps ahead by a small but statistically significant amount. Thus the results for universe 3 are comparable with those from universe 2 to which it is similar in all regards except for DCR. The lower ACRs for universes 2 and 3, therefore, appear to be caused by the de-majorising of the class distributions.

6. Conclusion

The method of artificial universes has the class model as its central notion. By focusing on the class distributions of the model as the means of describing noise

together with the inclusion of redundant and irrelevant attributes, the process of setting up learning tasks of varying degrees of difficulty is greatly facilitated. Smallish universes with 5 to 10 attributes can provide quite challenging tasks yet remain comprehensible to the experimenter.

The majorisation relation affords a simple practical way of manipulating noise levels and through its link with information and uncertainty measures gives a unified account of several seemingly different indicators.

It would appear from experimental results that, at least for ID3, increasing noise by de-majorising class distributions leads to poorer learning curves.

References

[1] D.W. Aha, Generalising from case studies: a case study, in: D. Sleeman and P. Edwards, eds., *Proceedings Ninth International Conference on Machine Learning* (Morgan Kaufmann, San Mateo, CA, 1992) 1–10.

[2] D.W. Aha, D. Kibler and M. Albert, Instance-based learning algorithms, *Mach. Learning* **6** (1991) 37–66.

[3] D. Angluin and P. Laird, Learning from noisy examples, *Mach. Learning* **2** (1988) 343–370.

[4] L.B. Booker, D.A. Goldberg and J.H. Holland, Classifier systems and genetic algorithms, *Artif. Intell.* **40** (1989) 235–282.

[5] L. Breiman, J. Friedman, R. Olshen and C. Stone, *Classification and Regression Trees* (Wadsworth, Belmont, CA, 1984).

[6] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor and D. Freeman, AUTOCLASS: a Bayesian classification system, in: *Proceedings Fifth International Conference on Machine Learning* (Morgan Kaufmann, San Mateo, CA, 1988) 54–64.

[7] P. Clark and R. Boswell, Rule induction with CN2: some recent improvements, in: Y. Kodratoff, ed., *EWSL-91* (Springer-Verlag, Berlin, 1991) 151–163.

[8] P. Clark and T. Niblett, The CN2 induction algorithm, *Mach. Learning* **3** (1989) 261–283.

[9] S. Cost and S. Salzberg, A weighted nearest neighbour algorithm for learning with symbolic features, *Mach. Learning* **10** (1993) 57–78.

[10] D. Haussler, Quantifying inductive bias: AI learning algorithms and Valiant's learning framework, *Artif. Intell.* **36** (1986) 177–221.

[11] R.J. Hickey, A note on the measurement of randomness, *J. Appl. Prob.* **19** (1982) 229–232.

[12] R.J. Hickey, Majorisation, randomness and some discrete distributions, *J. Appl. Prob.* **20** (1983) 897–902.

[13] R.J. Hickey, Continuous majorisation and randomness, *J. Appl. Prob.* **21** (1984) 924–929.

[14] R.J. Hickey, Artificial universes: towards a systematic approach to evaluating algorithms which learn from examples, in: D. Sleeman and P. Edwards, eds., *Proceedings Ninth International Conference on Machine Learning* (Morgan Kaufman, San Mateo, CA, 1992) 196–205.

[15] R.C. Holte, Very simple classification rules perform well on most commonly used datasets, *Mach. Learning* **11** (1993) 63–91.

[16] D. Kibler and P. Langley, Machine learning as an experimental science, in: D. Sleeman, ed., *Proceedings Third European Working Session on Learning* (Pitman, Glasgow, Scotland, 1988) 81–92.

[17] P. Langley, W. Iba and K. Thompson, An analysis of Bayesian classifiers, in: *Proceedings AAAI-92*, San Jose, CA (MIT Press, Cambridge, MA, 1992) 223–228.

[18] H Lounis and G.M. Bisson, Evaluation of learning systems: an artificial databased approach, in: Y. Kodratoff, ed., *EWSL-91* (Springer-Verlag, Berlin, 1991) 463–481.

[19] A.W. Marshall and I. Olkin, *Inequalities: The Theory of Majorisation and its Applications* (Academic Press, New York, 1979).

[20] J. Mingers, An empirical comparison of pruning methods for decision tree induction, *Mach. Learning* **4** (1989) 227–243.
[21] P.M. Murphy and D.W. Aha, UCI repository of machine learning databases, Maintained at the Department of Information and Computer Science, University of California, Irvine, CA (1992).
[22] M.J. Pazzini and W. Sarrett, A framework for average case analysis of conjunctive learning algorithms, *Mach. Learning* **9** (1992) 349–372.
[23] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, San Mateo, CA, 1988).
[24] J.R. Quinlan, Induction of decision trees, *Mach. Learning* **1** (1986) 81–106.
[25] J.R. Quinlan, Simplifying decision trees, *Int. J. Man-Mach. Stud.* **27** (1987) 221–234.
[26] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland and the PDP Research Group, eds., *Parallel Distributed Processing* **1** (MIT Press, Cambridge, MA, 1986) 318–362.
[27] Y. Sakakibara, Noise-tolerant occam algorithms and their applications to learning decision trees, *Mach. Learning* **11** (1993) 37–62.
[28] C. Schaffer, Sparse data and the effect of overfitting avoidance in decision tree induction, in: *Proceedings AAAI-92*, San Jose, CA (MIT Press, Cambridge, MA, 1992) 147–152.
[29] C. Schaffer, Overfitting avoidance as bias, *Mach. Learning* **10** (1993) 153–178.
[30] C.E. Shannon and W. Weaver, *Mathematics of Communication Theory* (University of Illinois Press, Urbana, IL, 1964).
[31] L.G. Valiant, A theory of the learnable, *Commun. ACM* **27** (1984) 1134–1142.
[32] L.G. Valiant, Learning conjunctions of disjunctions, in: *Proceedings IJCAI-85*, Los Angeles, CA (Morgan Kaufmann, San Mateo, CA, 1985) 560–566.

ELSEVIER

Artificial Intelligence 82 (1996) 181–213

Artificial Intelligence

Optimal composition of real-time systems

Shlomo Zilberstein *, Stuart Russell

Computer Science Division, University of California, Berkeley, CA 94720, USA

Received November 1993; revised October 1994

Abstract

Real-time systems are designed for environments in which the utility of actions is strongly time-dependent. Recent work by Dean, Horvitz and others has shown that *anytime algorithms* are a useful tool for real-time system design, since they allow computation time to be traded for decision quality. In order to construct complex systems, however, we need to be able to compose larger systems from smaller, reusable anytime modules. This paper addresses two basic problems associated with composition: how to ensure the *interruptibility* of the composed system; and how to allocate computation time optimally among the components. The first problem is solved by a simple and general construction that incurs only a small, constant penalty. The second is solved by an off-line compilation process. We show that the general compilation problem is NP-complete. However, efficient local compilation techniques, working on a single program structure at a time, yield globally optimal allocations for a large class of programs. We illustrate these results with two simple applications.

1. Introduction

This paper describes work on a fundamental problem in computer science and artificial intelligence, namely the construction of systems that can operate robustly in a variety of real-time environments. A real-time environment can be characterized by a *time-dependent* utility function. In almost all cases, the deliberation required to select optimal actions will degrade the system's overall utility. It is by now well-understood that a successful system must trade off decision quality for deliberation cost [2, 4, 15, 21, 29, 31, 32].

* Present address: Computer Science Department, Lederle Graduate Research Center, University of Massachusetts, Amherst, MA 01003, USA.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0004-3702(94)00074-3

The problem of deliberation cost has been widely discussed in artificial intelligence, economics and philosophy. In artificial intelligence in particular, researchers have proposed a number of meta-level architectures to control the cost of base-level reasoning [5, 6, 8, 12, 15, 28]. One promising approach is to use *anytime* [7] or *flexible* [14] algorithms, which allow the execution time to be specified, either as a parameter or by an interrupt, and exhibit a time/quality tradeoff defined by a *performance profile*. They provide a simple means by which a system can control its deliberation without significant overhead.

Soon after the introduction of anytime algorithms, it became apparent that their composition presents a vital, nontrivial problem [7]. This paper reports the first results on the composition problem showing that real-time systems can be modularly composed of anytime algorithms. Moreover, the meta-level scheduling problem is solved in polynomial time to yield optimal (near-optimal) performance for any tree (directed acyclic graph) structured program. These results extend the advantages of anytime algorithms to the design of complex real-time systems with many components.

In standard algorithms, the fixed quality of the output allows for composition to be implemented by a simple call-return mechanism. When algorithms have resource allocation as a degree of freedom, and can be interrupted at any time, the situation becomes more complex. Consider the following simple example: a real-time medical expert system containing a diagnosis component which passes its results to a treatment-planning component. The following issues arise:

(1) How can the individual components be designed as anytime algorithms?
(2) How can their performance be described as a function of time and the nature of the inputs?
(3) How does the output quality of the treatment component depend on the accuracy of the diagnosis it receives?
(4) What sort of programming language constructs are needed to specify how the system is built from its components?
(5) For any given amount of time, how should that time be allocated to each of the components?
(6) What if the condition of the patient suddenly requires intervention while the diagnosis component is still running and no treatment has been considered?
(7) How should the execution of the composite system be managed so as to optimize overall utility?

In other publications, particularly [34], we address these issues in some depth. Here, we focus on item (5), which we call the *compilation* problem. Given a system composed of anytime algorithms, compilation determines off-line the optimal allocation of time to the components for any given total allocation. The crucial meta-level knowledge for solving this problem is kept in the *anytime library* in the form of *conditional performance profiles*. These profiles characterize the performance of each elementary anytime algorithm as a function of run-time and input quality. In Section 2, we define the basic properties of anytime algorithms. An important distinction is made between *contract* algorithms, which require the determination of the total run-time when activated, and *interruptible* algorithms, whose total run-time is unknown in advance. The reduction theorem shows how to construct an interruptible algorithm once a contract algorithm is

compiled. In Section 3, we define the compilation problem and present a simple example of compilation. Then, in Section 4, we analyze in detail the compilation of functional composition. While the general compilation problem is shown to be NP-complete in the strong sense, local compilation techniques, whose complexity is linear in the size of the program, are shown to be both efficient and optimal for a large class of programs. In addition, a number of efficient approximation algorithms are given for the general case. Finally, Section 5 summarizes the benefits of compilation and outlines the direction for further work in this field.

2. Anytime algorithms

The term "anytime algorithm" was coined by Dean in the late 1980s in the context of his work on time-dependent planning. Anytime algorithms expand upon the traditional view of a computational procedure as they offer to fulfill an entire spectrum of input–output specifications, over the full range of run-times, rather than just a single specification. A standard algorithm is an implementation of a mapping from a set of inputs into a set of outputs. For each input that specifies a problem instance there is a particular element in the output set that is considered the *correct* solution to be generated by the algorithm. An anytime algorithm is an implementation of a mapping from a set of inputs and time allocation into a set of outputs. For each input there is a corresponding set of possible outputs, each of which is associated with a particular time allocation and some measure of its quality. The advantage of this generalization is that computation can be interrupted at any time and still produce results of a certain quality, hence the name "anytime algorithm".

2.1. Measuring the quality of results

In the context of anytime algorithms, a *quality measure* is typically a function from the output of the algorithm to the [0,1] interval. It may or may not be related to the utility function of the system that incorporates the algorithm; but it should measure some aspect of the algorithm's output that improves over time, at least on average. The following three metrics have proved useful in anytime algorithm construction:

(1) *Certainty*. This metric reflects the degree of certainty that a result is correct. The degree of certainty can be expressed using probabilities, fuzzy set membership, or any other method of expressing uncertainty. For example, consider an anytime diagnosis algorithm that is based on combining more and more evidence as computation time increases. The certainty that the diagnosis is correct increases as a function of run-time, but there remains a possibility that the correct result is entirely different from the result generated by the algorithm.

(2) *Accuracy*. This metric reflects the degree of accuracy in the value returned by the algorithm, typically through a bound on the difference from the exact solution. For example, if a Taylor series is being used to approximate a certain function, then the error bound (given by Lagrange's remainder formula) decreases with the iteration number. This error bound determines the quality of the results.

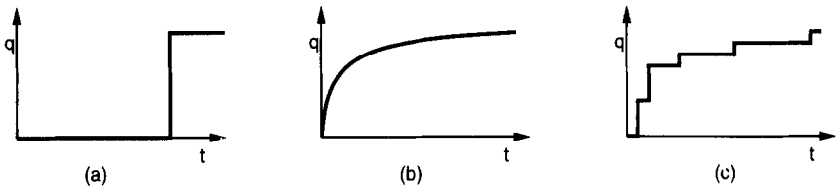


Fig. 1. Typical performance profiles: (a) Standard algorithm. (b) Idealized anytime algorithm. (c) Actual anytime algorithm.

(3) *Specificity*. This metric reflects the level of detail of the result. In this case, the anytime algorithm always produces *correct* results, but the level of *detail* is increased over time. For example, consider a hierarchical diagnosis algorithm that pinpoints a subassembly as the source of the fault. Over time, this can be refined all the way down to primitive components, but at any point the output is correct, even if not fully specific.

Notice that accuracy, a standard measure in numerical domains, can be mapped onto specificity, which is more commonly used in symbolic domains. An inaccurate numerical solution is very specific but incorrect, and could be mapped to an equally useful, correct statement that the solution lies within a certain interval. Anytime algorithms can also have multidimensional quality measures. For example, PAC algorithms for inductive learning are characterized by an uncertainty measure, δ, and a precision measure, ϵ.

2.2. Performance profiles

The *performance profile* of an algorithm characterizes the quality of its output as a function of computation time. All algorithms—whether standard or anytime—have a performance profile. Fig. 1 shows typical performance profiles for standard algorithms (a) and idealized anytime algorithms (b). The performance profile of the standard algorithm shows that no results are available until its termination at which point the exact result is returned. The idealized anytime algorithm provides output whose quality improves gradually over time. In practice, the improvement in quality of an anytime algorithm may look more like the profile shown in Fig. 1(c).

Strictly speaking, such profiles are defined only for a particular input, and only for deterministic algorithms. We will also need to describe the output quality for a population of inputs, and for a set of runs of a randomized algorithm. Further refinements are needed to describe how the performance depends on various aspects of the input such as quality and size. We also need ways to *acquire* and *represent* profiles. These issues are dealt with in the following subsections.

2.2.1. Categories of performance profiles

Given a deterministic anytime algorithm $\mathcal{A}$, let $q_{\mathcal{A}}(x,t)$ be the quality of results produced by $\mathcal{A}$ with input x and computation time t; let $q_{\mathcal{A}}(t)$ be the expected quality of results with computation time t; and let $p_{\mathcal{A},t}(q)$ be the probability (density function in the continuous case) that $\mathcal{A}$ with computation time t produces results of quality q.

The most informative type of performance profile used in this work is the performance distribution profile defined below:

Definition 2.1. The *performance distribution profile* (*PDP*) of an algorithm $\mathcal{A}$ is a function $D_{\mathcal{A}} : \mathbb{R}^+ \rightarrow Pr(\mathbb{R})$ that maps computation time to a probability distribution over the quality of the results.

An obvious simplification of the PDP is the expected performance profile (EPP), as used by Boddy and Dean [2] and by Horvitz [14]:

Definition 2.2. The *expected performance profile* (*EPP*) of an algorithm $\mathcal{A}$ is a function $E_{\mathcal{A}} : \mathbb{R}^+ \rightarrow \mathbb{R}$ that maps computation time to the expected quality of the results.

Note that $E_{\mathcal{A}}(t)$ can be calculated directly when the expected quality of the algorithm can be determined for each time allocation or it can be estimated by averaging the actual quality achieved over many problem instances (as shown in Eq. (1).

$$E_{\mathcal{A}}(t) = \sum_q p_{\mathcal{A},t}(q)q = \sum_x Pr(x)q_{\mathcal{A}}(x,t). \tag{1}$$

For any summary description of component algorithms, it is important to understand how the summary description of a composite system can be derived from the descriptions of its components. Suppose, for example, that for any particular input to a two-component system, the output quality is some function f of the qualities $q_1(x,t)$ and $q_2(x,t)$ achieved by the components. Unfortunately, it is generally the case that

$$f(E_x(q_1(x,t)), E_x(q_2(x,t))) \neq E_x(f(q_1(x,t), q_2(x,t))).$$

Hence the EPP of the composed system cannot be recovered easily from the EPPs of the components. EPPs are therefore most useful when the variance of the original PDPs is small, so that the error associated with composition of EPPs is also small. In the special case where the variance of the distribution is zero (or infinitesimal), the anytime algorithm is said to have a *fixed performance*. For such algorithms, an expected performance profile offers a complete, accurate description of performance.

Definition 2.3. The *performance interval profile* (*PIP*) of an algorithm $\mathcal{A}$ is a function $I_{\mathcal{A}} : \mathbb{R}^+ \rightarrow \mathbb{R} \times \mathbb{R}$ that maps computation time to the upper and lower bounds of the quality of the results.

Note that if $I_{\mathcal{A}}(t) = [L, U]$ then:

$$\forall x: \quad L \leqslant q_{\mathcal{A}}(x,t) \leqslant U. \tag{2}$$

Performance interval profiles offer a representation that is both compact and easy to manipulate. From the lower bounds on the qualities of the results of two algorithms, one can normally find a lower bound on the quality of their combined result. Hence, when a compact representation is preferred and the variance of the distribution is wide, performance interval profiles are useful.

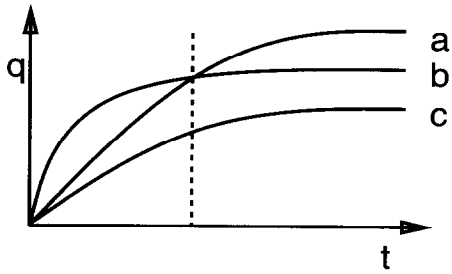


Fig. 2. Dominance relations among profiles: a and b dominate c, but neither of a and b dominates the other.

In order to define optimal compilation, we will also need a notion of *dominance* among profiles.

Definition 2.4. Let $\mathcal{A}$ and $\mathcal{B}$ be two anytime algorithms that solve the same problem, then $\mathcal{B}$ is said to *dominate* $\mathcal{A}$ ($\mathcal{B} \succ \mathcal{A}$) if for every input x and every time allocation t:

$$\forall x\ \forall t\colon \quad q_{\mathcal{B}}(x,t) \geqslant q_{\mathcal{A}}(x,t).$$

The relationship of dominance between anytime algorithms is a partial order. Given two anytime algorithms that solve a certain problem, it is possible that neither of them dominates the other (see Fig. 2).

2.2.2. Conditional performance profiles

It may happen that the performance of the algorithm depends significantly on the nature of the inputs, in which case the PDP will be too coarse for general use. If the input dependence can be attributed to a small set of features, one can use a *conditional performance profile* by partitioning the input domain into classes and storing a separate profile for each input class. The partitioning can be done using any attribute of the input that may influence performance, such as size or a complexity measure. Input classes of similar performance can also be derived automatically using Bayesian statistics by programs such as Autoclass [3].

In this paper, we consider conditioning on the input quality. A conditional performance profile therefore consists of a mapping from input quality and run-time to probability distribution of output quality:

Definition 2.5. The *conditional performance profile* (*CPP*) of an algorithm $\mathcal{A}$ is a function $C_{\mathcal{A}} : \mathbb{R} \times \mathbb{R}^{+} \rightarrow Pr(\mathbb{R})$ that maps input quality and computation time to a probability distribution over the quality of the results.

A CPP can also be seen as a family of PDPs, each for a different input quality and denoted by $C_{\mathcal{A},q}$. This leads to the graphical representation shown in Fig. 3. Each curve in the figure represents an expected performance profile for a particular input quality.

The information in the CPP is essential to the compilation process since the allocation of time to a certain module affects not only the quality of the result of that module but

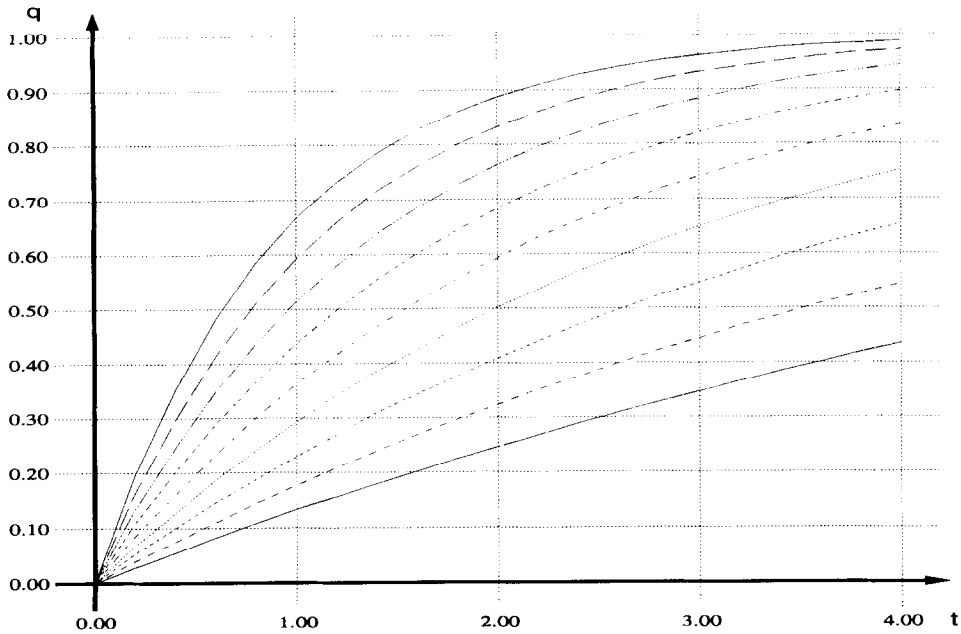


Fig. 3. Graphical representation of a conditional performance profile.

also the quality of the results of any module that uses that result. The CPP thus provides enough information to characterize the performance of any module in such a way that it can be combined optimally with other modules without "looking inside".

Definition 2.6 (*Input monotonicity*). A CPP for algorithm $\mathcal{A}$ exhibits input monotonicity if and only if

$$\forall p, q: \quad p > q \Rightarrow C_{\mathcal{A},p} \succ C_{\mathcal{A},q}.$$

That is, as the input quality improves, so should the performance profile. This is a very natural property, and also very useful as we show below.

2.2.3. Acquiring and representing performance profiles

Performance profiles can sometimes be calculated by algorithm analysis. For example, in many iterative algorithms, such as Newton's method, the error in the result is bounded by a function that depends on the number of iterations. In such cases, the performance profile can be calculated once the run-time of a single iteration is determined. In general, however, such structural analysis of the code is hard because the improvement in quality in each iteration and its run-time may be unpredictable. To overcome this difficulty, a general *simulation* method can be used. It is based on gathering statistics on the performance of the algorithm on randomly generated problem instances. Ideally, the statistics are gathered for the same population of instances as will appear when

the algorithm is deployed. This can be ensured by learning the profiles during actual operation.

Performance profiles can be represented either by a closed form or as a table of discrete entries. Since performance profiles are normally monotone functions of time, they can be approximated using a simple family of functions. Once the simulation data is gathered, the performance information can be derived by various curve fitting techniques. For example, Boddy and Dean [2] used the function: $Q(t) = 1 - e^{-\lambda t}$ to model the expected performance of an anytime planner. Performance distribution profiles can be approximated by applying a similar method to a family of distributions. For example, if the normal distribution is used, one can apply curve fitting techniques to approximate the mean and variance of the distribution as a function of time.

The advantage of using a closed-form representation of performance profiles is that optimization of time allocations can be performed for a general parameterized family of profiles, using straightforward calculus techniques. The results of such compilation can be used each time members of that family are compiled. Closed-form representation has two major disadvantages: (1) fitting a closed-form approximation to real data may involve a large error; and (2) it is hard to maintain closure under the compilation operation. The closure property requires that the result of compilation of two (or more) performance profiles that belong to a certain family be a member of the same family, or at least that it be approximable by a function in that family. The disadvantages of the closed-form representation led us to use a more flexible, discrete representation.

The discrete representation of performance profiles is based on a table that specifies the discrete probability distribution over quality for a range of time allocations. For this purpose, the complete range of qualities has to be divided into discrete qualities $q_1, \ldots, q_n$. The entry i, j in the table represents the discrete probability that with time allocation t_i the actual output quality q is in the range $[q_j - \delta, q_j + \delta]$. The size of the table is a system parameter that controls the accuracy of performance information. Linear interpolation is used to find the quality when the run-time does not match exactly one of the table entries.

2.3. Interruptible versus contract algorithms

We make an important distinction between two types of anytime algorithms called *interruptible* algorithms and *contract* algorithms. Interruptible algorithms produce results of the "advertised quality" even when interrupted unexpectedly. Contract algorithms, although capable of producing results whose quality varies with time allocation, must be given a particular time allocation in advance. If a contract algorithm is interrupted at any time shorter than its contract time, it may yield no useful results. Both interruptible and contract algorithms have been used in the past. Dean and Boddy's [7] definition of anytime algorithms refers to the interruptible case. Techniques such as depth-limited search and alpha-beta search, on the other hand, are more suited for contract algorithms. Although they can produce a suitable result for any given effort limit, they may return meaningless results if interrupted before completion.

In general, every interruptible algorithm is trivially a contract algorithm, but the converse is not true. Intuitively, one tends to think about anytime algorithms as interruptible,

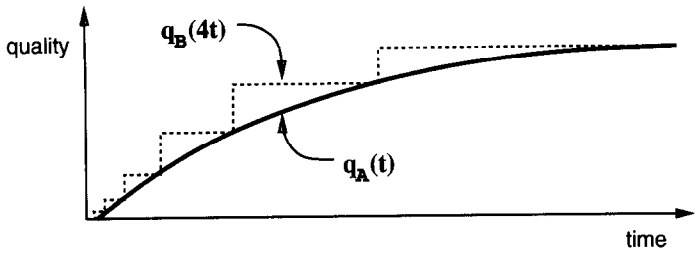


Fig. 4. Performance profiles of interruptible and contract algorithms.

yet the greater freedom of design makes it easier to construct contract algorithms than interruptible ones. In the case of functional composition, as illustrated by the real-time medical system mentioned above, it is possible to allocate a fixed contract time optimally between the two components. This results, however, in a contract algorithm since interrupting the system during diagnosis leaves one with no treatment recommendation at all. This is the case even if the individual components are themselves interruptible. Thus naïve composition destroys interruptibility. This problem is solved by the following reduction theorem [30]:

Theorem 2.7 (Reduction). *For any contract algorithm $\mathcal{A}$, an interruptible algorithm $\mathcal{B}$ can be constructed such that for any particular input $q_B(4t) \geqslant q_A(t)$.*

Proof. Construct $\mathcal{B}$ by running $\mathcal{A}$ repeatedly with exponentially increasing time limits. If interrupted, return the best result generated so far. Let the sequence of run-time segments be $\epsilon, 2\epsilon, \ldots, 2^i\epsilon, \ldots$, and assume that the time overhead of the code required to control this loop can be ignored. Note also that $\sum_{i=0}^{n-1} 2^i = 2^n - 1$. The worst case situation occurs when $\mathcal{B}$ is interrupted after almost $(2^n - 1)\epsilon$ time units, just before the last iteration terminates and the returned result is based on the previous iteration with a run-time of $2^{n-2}\epsilon$ time units. Since $(2^n - 1)/2^{n-2} < 4$, the factor of 4 results. If one replaces the multiplier of time intervals by α, one gets a time ratio of $(\alpha^n - 1)/(\alpha^{n-1} - \alpha^{n-2})$. The lower bound of this expression is 4, for $\alpha = 2$, hence 2 is the optimal multiplier under this strategy. □

Fig. 4 shows a typical performance profile for the contract algorithm $\mathcal{A}$, and the corresponding performance profile for the constructed interruptible algorithm $\mathcal{B}$, reduced along the time axis by a factor of 4. As an example, consider the application of this construction method to Korf's RTA*, a contract algorithm. As the time allocation is increased exponentially, the algorithm will increase its depth bound by a constant; the construction therefore generates an iterative deepening search automatically.

2.4. Programming techniques for anytime algorithms

The development of elementary anytime algorithms does not require a radical change in programming methodologies. Many existing programming and automated reasoning techniques produce useful anytime algorithms: search techniques such as iterative deep-

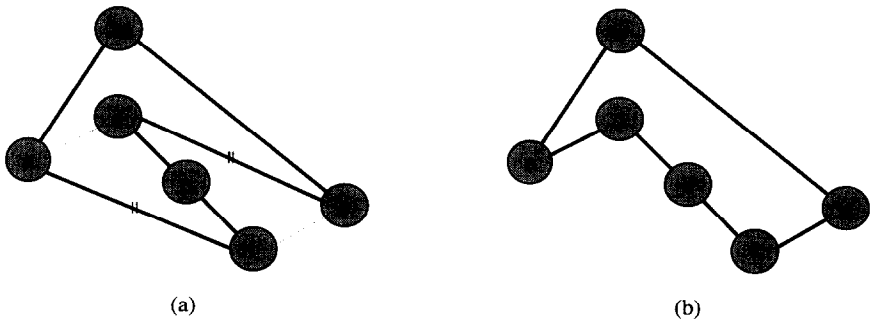


Fig. 5. The operation of randomized tour improvement.

ening; asymptotically correct inference algorithms such as approximate query answering [9,32], bounded cutset conditioning (see [14]), and variable precision logic [24]; various greedy algorithms (see [2]); iterative methods such as Newton's method; adaptive algorithms such as PAC learning algorithms or neural networks; randomized methods such as Monte Carlo algorithms or fingerprinting techniques [17]; and the use of optimal meta-level control of computation [28]. We conclude this section with an example of a particular anytime algorithm and its performance profile.

Example [The Traveling Salesman Problem] The *Traveling Salesman Problem* (TSP) involves a salesman that must visit n cities. If the problem is modeled as a complete graph with n vertices, the solution becomes a *tour*, or Hamiltonian cycle, visiting each city exactly once, starting and finishing at the same city. The cost function, $Cost(i,j)$, defines the cost of traveling directly from city i to city j. (The cost is not necessarily the Euclidean distance.) The problem is to find an optimal tour, that is, a tour with minimal total cost. The TSP is known to be NP-complete [10], hence it is hard to find an optimal tour when the problem includes a large number of cities. *Iterative improvement* algorithms can find a good approximation to an optimal solution, and naturally yield an interruptible anytime algorithm.

The anytime traveling salesman algorithm is a randomized algorithm that repeatedly tries to perform a tour improvement step [20,22]. In the general case of tour improvement procedures, r edges in a feasible tour are exchanged for r edges not in that solution as long as the result remains a tour and the cost of that tour is less than the cost of the previous tour. The simplest case is when $r = 2$. Fig. 5 demonstrates one step of tour improvement. An existing tour, shown in part (a), visits the vertices in the following order: a, b, c, d, e, f. The algorithm selects two random edges of the graph, (c,d) and (f,a) in this example, and checks whether the following condition holds:

$$Cost(c,f) + Cost(d,a) < Cost(c,d) + Cost(f,a). \tag{3}$$

If this condition holds, the existing tour is replaced by the new tour, shown in part (b), a, b, c, f, e, d. The improvement condition guarantees that the new path has a lower cost. The algorithm starts with a random tour that is generated by simply taking a random ordering of the cities. Then the algorithm tries to reduce the cost by a sequence

```
ANYTIME-TSP(V, iter)
  Tour ← INITIAL-TOUR(V)
  cost ← COST(Tour)
  REGISTER-RESULT(Tour)
  for i ← 1 to iter
    e1 ← RANDOM-EDGE(Tour)
    e2 ← RANDOM-EDGE(Tour)
    δ ← COST(Tour) − COST(SWITCH(Tour, e1, e2))
    if δ > 0 then
      Tour ← SWITCH(Tour, e1, e2)
      cost ← cost − δ
      REGISTER-RESULT(Tour)
  SIGNAL(TERMINATION)
  HALT
```

Fig. 6. The anytime traveling salesman algorithm.

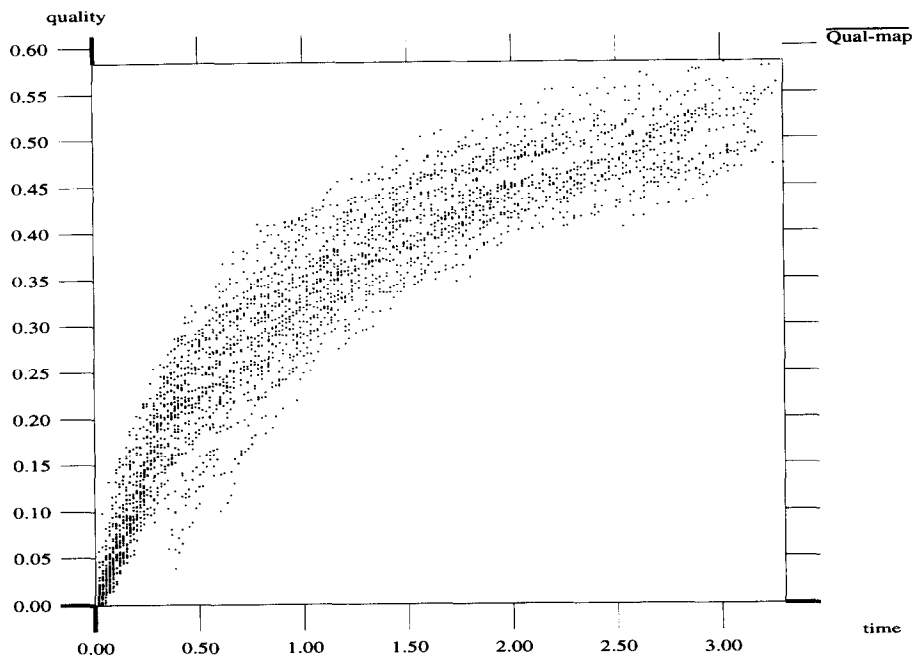


Fig. 7. The quality map of the TSP algorithm.

of random improvements. The result is an interruptible anytime algorithm, as shown in Fig. 6. Note that the algorithm has a generic design that includes an initial step to generate and register the first result followed by a loop containing an improvement step. The compiled code handles an interrupt by returning the most recently registered result.

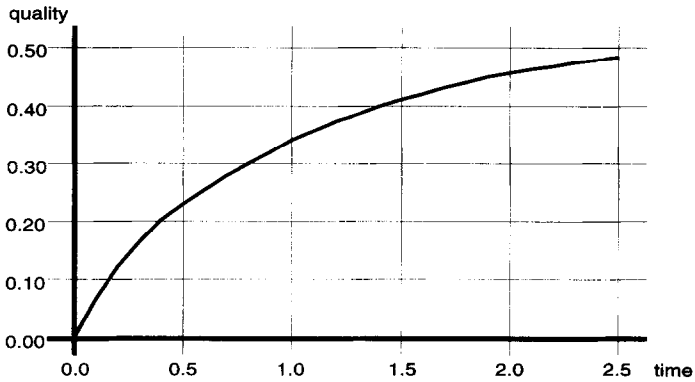


Fig. 8. The expected performance profile of the TSP algorithm.

The *iter* argument indicates the maximum number of iterations but execution can be interrupted by the monitor at an earlier point.

Fig. 7 shows the *quality map* of the algorithm, which summarizes the results of many activations with randomly generated input instances (including 50 cities). Each point (t, q) represents an instance for which quality q was achieved with run-time t. The quality of results in this experiment measures the percentage of tour length reduction with respect to the initial tour. These statistics form the basis for the construction of the performance profile of the algorithm. The resulting expected performance profile is shown in Fig. 8.

3. Compilation of anytime algorithms

We now turn from the examination of individual anytime algorithms to the problem of building large systems using anytime algorithms as components. The compilation process, illustrated in Fig. 9, plays a central role in the solution to this problem.

The input to the compiler is a compound anytime module, that is, a module composed of several elementary anytime algorithms. The primitive programming language constructs that are used to define compound modules can vary from a small set of simple constructs to a rich programming language [34]. The choice of language primitives determine the feasibility and complexity of the compilation problem. Compound modules do not include time allocation code and hence they are not readily executable. In addition to the compound module, the compiler's input includes the performance profiles of the elementary anytime algorithms. The result of the compilation process is an executable anytime module that consists of a compiled version of the original module, a pre-defined run-time monitor, and the performance profile of the system that may include some auxiliary time allocation information. The compiled module includes code to control the activation of the elementary components with an appropriate time allocation.

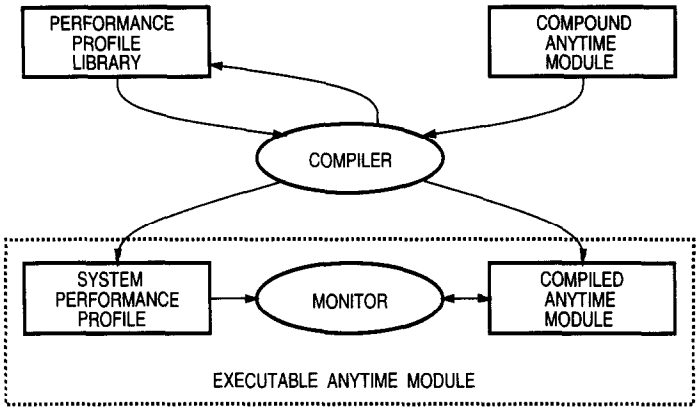


Fig. 9. Compilation and monitoring.

Optimal scheduling of the elementary components may also require run-time monitoring. The problem addressed by the monitor is similar to the *deliberation scheduling* problem introduced by Dean and Boddy in [7]. Previous solutions to the problem included only a small set of algorithms characterized by non-conditional performance profiles. In this work we have studied the composition and monitoring of an arbitrary number of different algorithms characterized by conditional performance profiles. We found that the complexity of the compilation process is largely determined by the choice of a run-time monitoring scheme. Active monitoring, that revises the allocation the the components while the system is active, is discussed in [34]. To simplify the discussion here, we assume that time allocation to the components is determined prior to the activation of the system.

3.1. Aspects of the compilation problem

The solution to the compilation problem depends on a number of factors that characterize the inputs and the outputs of the process. The main aspects of the problem are described below:

(1) *Program structure.* The structure of a compound anytime module is a primary factor that determines the complexity of its compilation. Some programming structures, such as sequencing, are easier to handle, while others, such as recursive function calls, are quite difficult to compile.

(2) *Type of performance profiles.* The type of performance profiles and their representation also influence the compilation process. Highly informative performance profiles, such as the performance distribution profile, are more difficult to compile and manipulate. The complexity of the compilation is increased due to the complexity of the representation and the requirement that the resulting performance profile provides the same level of information.

(3) *Type of anytime algorithms.* The type of algorithm used as input to the compiler and the desired type of the resulting algorithm have a direct effect on the

compilation process. Contract algorithms are normally easier to construct both as elementary and as compound algorithms. Interruptible algorithms are more complicated. One can, of course, construct first a contract algorithm and then use the result of Theorem 2.7 to make it interruptible. However, with some programming structures it is advantageous to generate an interruptible algorithm directly and avoid the constant slowdown of the reduction theorem.

(4) *Type of monitoring.* Anytime computation can be controlled using either passive or active monitoring. Passive monitoring means that meta-level scheduling decisions are made *before* the activation of the anytime algorithms. Elementary algorithms are activated as contract algorithms only and their run-time cannot be modified before the termination of the contract. Obviously, the assumption of passive monitoring limits the capability to optimize the performance profile of a system, but it also simplifies the compilation problem. With active monitoring, time allocation decisions may be made after the activation of the system, in response to the *actual* rather than *expected* performance of the components.

(5) *Quality of intermediate results.* With both interruptible and contract anytime algorithms, an active monitor can examine the quality of intermediate results in order to modify the allocation of the remaining time. However, this requires a capability to determine the *actual* quality of intermediate results. The quality of intermediate results may be a simple aspect that can be quickly calculated. For example, in the case of a bin packing program whose quality function is the proportion of the container space filled with packages, the quality of an intermediate result can be easily calculated. In other cases, such as a chess playing program, the quality of a recommended move is not apparent from the move itself. Hence, the capability to determine the quality of intermediate results is an important factor in compilation and monitoring.

3.2. Compilation examples

As a simple example of compilation, consider the composition of two anytime algorithms. Suppose that one algorithm takes the input and produces an intermediate result. This result is then used as input to another anytime algorithm which, in turn, produces the final result. Many systems can be implemented by a composition of a sequence of two or more algorithms. We will examine two particular systems. The first is a repair system whose elementary performance profiles are represented using a closed form. The second is a path planning system whose performance profiles are represented using the discrete tabular approach.

3.2.1. Composition of diagnosis and treatment planning

Consider an automated repair system that is composed of two anytime algorithms: diagnosis and treatment planning. The system can be represented by the following expression:

$$Output \leftarrow \text{TREATMENT}(\text{DIAGNOSIS}(Input)).$$

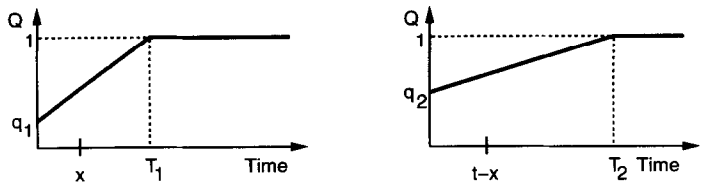


Fig. 10. Performance profiles of DIAGNOSIS and TREATMENT.

The input to DIAGNOSIS is a set of symptoms for which a diagnosis is computed. This diagnosis is used as input to TREATMENT that produces the final output—a treatment plan. Fig. 10 shows the linear performance profiles of the elementary anytime algorithms. They start with an arbitrary initial quality q_i (that may be zero) and reach the maximal quality of 1 at time T_i. Hence they can be represented by:

$$Q_1(t) = q_1 + \alpha_1 t \quad (0 \leqslant t \leqslant T_1),$$
$$Q_2(t) = q_2 + \alpha_2 t \quad (0 \leqslant t \leqslant T_2).$$

The quality of DIAGNOSIS, Q_1, reflects the probability that the diagnosis is correct. Similarly, the quality of TREATMENT, Q_2, reflects the probability that the treatment plan repairs the problem *given* that the diagnosis is correct. Assuming that the qualities of the two modules are independent, we can express the overall quality by the product of the qualities of the two modules. Our goal is to compile the best contract algorithm for the complete system. In other words, the compilation process has to create the following mappings:

$$\mathcal{T} : \mathbb{R}^+ \to \mathbb{R}^+ \times \mathbb{R}^+, \tag{4}$$

$$\mathcal{PP} : \mathbb{R}^+ \to [0, 1]. \tag{5}$$

The first mapping specifies for each total allocation the amount of time that should be allocated to each algorithm so as to maximize the output quality.[1] The second mapping is the performance profile of the composed algorithm based on optimal time allocation. For each total allocation, t, the compiler has to find the optimal allocation, x, to the first algorithm (which implies allocation $t - x$ to the second algorithm) such that the overall quality $Q(x)$ is maximal.

Theorem 3.1. *Given the performance profiles of the input modules, the optimal time allocation mapping is:*

$$\mathcal{T} : t \to \left(\frac{1}{2}\left(t - \frac{q_1}{\alpha_1} + \frac{q_2}{\alpha_2}\right), \frac{1}{2}\left(t + \frac{q_1}{\alpha_1} - \frac{q_2}{\alpha_2}\right)\right). \tag{6}$$

Proof. Since the overall output quality is:

$$Q(x) = -\alpha_1\alpha_2 x^2 + (\alpha_1\alpha_2 t - q_1\alpha_2 + q_2\alpha_1)x + q_1 q_2 + q_1\alpha_2 t. \tag{7}$$

[1] Only the appropriate allocation to the first component is really necessary because the allocation to the second is simply the remaining time.

the maximal quality is achieved when $\partial Q/\partial x = 0$, or when:

$$-2\alpha_1\alpha_2 x + \alpha_1\alpha_2 t - q_1\alpha_2 + q_2\alpha_1 = 0. \tag{8}$$

The solution of this equation yields the above allocation. □

A trivial correction is needed to cover boundary conditions (since allocation to DIAGNOSIS should be in $[0, T_1]$ and to TREATMENT in $[0, T_2]$): (1) if an algorithm gets more run-time than is necessary for its completion, then the extra time should be allocated to the other algorithm (or ignored when both algorithms terminate); and (2) if the time allocation to one algorithm is negative, then all the available time should go to the other algorithm.

3.2.2. Composition of sensing and path planning

Consider a robot navigation system that is composed of two anytime algorithms: visual sensing and path planning. The system can be represented by the following expression:

Output ← PATH-PLAN(*Start*, *Goal*, GET-DOMAIN-DESCRIPTION(*Input*)).

The input to GET-DOMAIN-DESCRIPTION is raw data from a visual sensor from which the module constructs an approximate map of the robot's local environment. This map is used as input to PATH-PLAN that produces the final output—a path from *Start* to *Goal*. The actual implementation of these anytime modules is described in [37]. Fig. 11 shows the performance profiles.

The domain is represented as a matrix of elementary positions each of which can be either free or occupied by an obstacle. The quality of GET-DOMAIN-DESCRIPTION reflects the probability that an elementary domain position would be wrongly identified, that is, identified as free space while actually blocked by an obstacle or vice versa. In Fig. 11, T_a is the minimal amount of time needed for the module to produce an initial domain description with quality Q_a. For a run-time t, $T_a \leqslant t \leqslant T_b$, the quality of GET-DOMAIN-DESCRIPTION improves from Q_a to the maximal quality Q_b.

Path planning is performed using a coarse-to-fine search algorithm (similar to that of Lozano-Pérez and Brooks [23]) that allows for unresolved path segments. In order to make it an anytime algorithm, we vary the abstraction level of the domain description. This allows the algorithm to find a feasible plan quickly, and then repeatedly refine it by replanning a segment of the plan in more detail. The quality of a plan is the ratio between the length of the shortest path and the path that the robot follows when it uses the abstract plan. To capture the dependency of the quality of planning on the quality of sensing, we used a conditional performance profile.

Performance profiles in this application were represented using the discrete tabular approach. Using this representation, the compilation of the two modules becomes a discrete optimization problem that we solved using a simple search algorithm. Fig. 12 shows the resulting performance profile. Also shown in that figure are the performance profiles of two other modules: MIN, that allocates to GET-DOMAIN-DESCRIPTION a minimal amount of time, T_a, and MAX, that allocates a maximal amount of time, T_b. The compiled performance profile is superior to both. It is closer to MIN with small allocations of time and is closer to MAX in the limit.

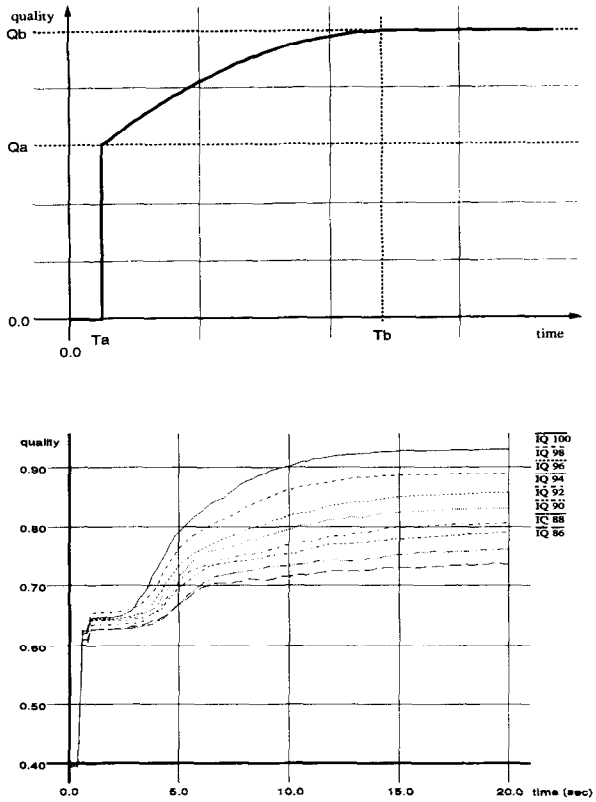


Fig. 11. (a) Expected performance profile of GET-DOMAIN-DESCRIPTION. (b) Conditional performance profile of PATH-PLAN, given input quality between 0.86 and 1.0.

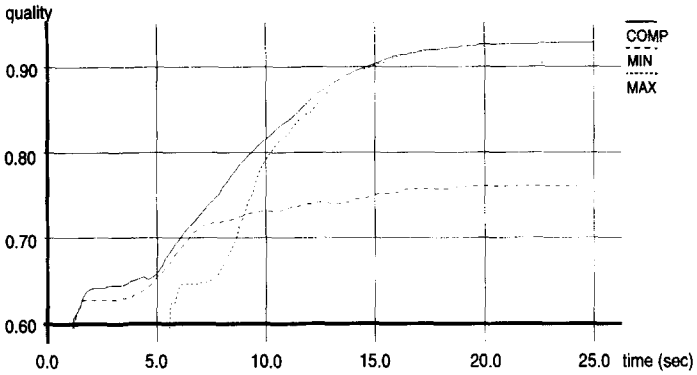


Fig. 12. The compiled performance profile for the composed system (COMP). The profiles labeled MIN and MAX show the result of minimal and maximal allocations to the vision component.

The above two examples demonstrate several general issues in compilation. When performance profiles are represented using a certain formula, as in the first example, the compilation problem involves solving a differential equation. The complexity of the equation, in terms of both size and number of variables, grows as a function of the number of elementary algorithms that are compiled. If a discrete tabular representation is used, then the compilation problem becomes a search problem in a discrete domain whose size grows exponentially with the number of modules. The problem of exponential growth in the complexity of compilation is addressed in the next section.

4. Compilation of functional expressions

We now turn to a more formal analysis of a general class of compilation problems, namely the family of programs created by functional composition of anytime algorithms. In functional composition each expression to be compiled is composed of an anytime function whose arguments may be either input variables or another expression created by functional composition. In the case of contract algorithms, the compilation task involves finding for each total allocation t, the best way to schedule the components so as to optimize the expected quality of the result of the complete expression.

Let $\mathcal{F}$ be a set of anytime functions. To simplify the discussion, assume that all function parameters are passed by value and that functions have no side-effects (as in pure functional programming). Let $\mathcal{I}$ be a set of input variables. The notion of a functional expression is defined as follows:

Definition 4.1. A *functional expression* over $\mathcal{F}$ with input $\mathcal{I}$ is:
(1) an input variable $i_j \in \mathcal{I}$, or
(2) an expression $f(g_1, \ldots, g_n)$ where $f \in \mathcal{F}$ and each g_j is a functional expression.

Each function $f \in \mathcal{F}$ has a fixed conditional performance profile associated with it that specifies the quality of its output as a function of time allocation and input quality.

Fig. 13 shows two possible graphical representations of the functional expression:

$$F(x) = E(D(B(A(x)), C(A(x)))).$$

The first representation is a tree constructed in the following way:
(1) If e is an input variable i_j, then it is represented by a leaf node i_j.
(2) If $e = f(g_1, \ldots, g_n)$, then it is represented by a tree whose root node is f and whose main subtrees are the trees representing $g_1, \ldots, g_n$.

The second representation is a directed acyclic graph (DAG) constructed in the following way:
(1) If e is an input variable i_j, then it is represented by a leaf node i_j.
(2) If $e = f(g_1, \ldots, g_n)$, then it is represented by a DAG that includes a node f and directed arcs from the (roots of the) DAGs representing $g_1, \ldots, g_n$ to f.

Notice that the DAG representation requires only one DAG to represent all the copies of a repeated subexpression, while the tree representation requires multiple copies of subtrees for repeated subexpressions. When a functional expression has no repeated

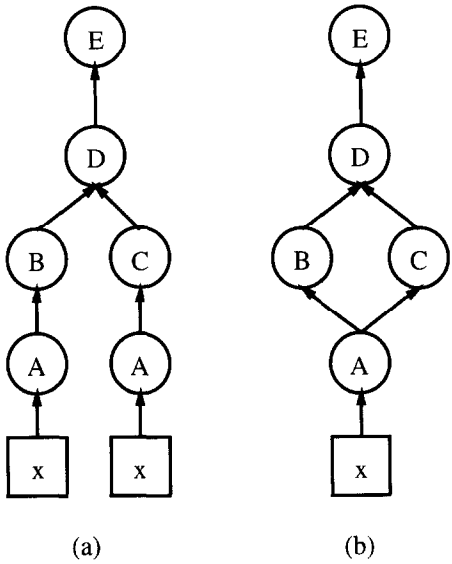


Fig. 13. Graph representation of functional expressions.

subexpressions, its tree and DAG representations are identical and its compilation is simplified.

4.1. The complexity of compilation

In this section we will analyze the complexity of compilation of functional expressions and show that the general problem is NP-complete in the strong sense. A relaxed version of the problem, that excludes repeated subexpressions, will be shown to be pseudo-polynomial.

The compilation problem is normally defined as an optimization problem, that is, a problem of finding a schedule for a set of components that yields maximal output quality. But in order to prove NP-completeness results, it is more convenient to refer to the decision problem variant of the compilation problem. This decision problem is stated as follows: given a functional expression e, the conditional performance profiles of its components, and a total allocation B, does there exist a schedule of the components that yields output quality greater than or equal to K? We refer to this decision problem as the problem of global compilation of functional expressions, or GCFE. The first complexity result asserts the following:

Theorem 4.2. *The GCFE problem is* NP-*complete in the strong sense.*

Proof. The GCFE problem is clearly NP since, given a particular allocation to the components, it is easy to determine in linear time the output quality of the expression. Hence, the verification problem is polynomial and the decision problem is NP. The rest of the proof is by transformation from the PARTIALLY ORDERED KNAPSACK

problem, an NP-complete problem in the strong sense [10] defined as follows:

- *Instance.* Finite set U, partial order $\prec$ on U, for each $u \in U$ a size $s(u) \in \mathbb{Z}^+$ and a value $v(u) \in \mathbb{Z}^+$, and positive integers B and K.
- *Question.* Is there a subset $U' \subseteq U$ such that if $u \in U'$ and $u' \prec u$, then $u' \in U'$, and such that $\sum_{u \in U'} s(u) \leqslant B$ and $\sum_{u \in U'} v(u) \geqslant K$?

An instance of the PARTIALLY ORDERED KNAPSACK problem can be directly transformed into a DAG representing a corresponding functional expression. To describe the construction of the DAG, we must first define the notion of a maximal element in a partially ordered set.

Definition 4.3. An element $u \in U$ is a *maximal element* of U if there is no other element $u' \in U$ such that $u \prec u'$.

The notion of a minimal element is defined in an analogous way. Every partially ordered set has at least one maximal element and at least one minimal element. Now, the construction of the DAG is defined as follows. For each $u \in U$ the DAG will contain a corresponding computational node. A direct arc goes from u_1 to u_2 if and only if u_1 is a maximal element of the set $\{u \mid u \prec u_2\}$ of all elements smaller than u_2. In addition, the DAG has a "root" node r with a directed arc from every other node $u \in U$ to r. The conditional performance profile of a node $u \in U$ is:

$$Q_u(q_1, \ldots, q_n, t) = \begin{cases} v(u), & \text{if } t \geqslant s(u) \text{ and } \forall i : q_i > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

where $q_1, \ldots, q_n$ are the qualities of the nodes that have a directed arc to u. If there is no such node, that is, if u is a minimal element of U, then its performance profile is:

$$Q_u(t) = \begin{cases} v(u), & \text{if } t \geqslant s(u), \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

The conditional performance profile of r is defined as follows:

$$Q_r(q_1, \ldots, q_k, t) = \sum_{i=1}^{k} q_i. \tag{11}$$

Finally, the overall output quality Q_{out} is defined as the quality of the root node, r.

It is easy to see that the construction of the DAG can be accomplished in polynomial time. All that is left to show is that the answer to the PARTIALLY ORDERED KNAPSACK problem is "yes" if and only if the answer to the corresponding GCFE problem is "yes".

If the answer to the GCFE problem is positive (with contract time B and minimal output quality K), then define U' as the set of nodes $u' \in U$ whose "output quality" in the DAG is positive. The sum of the output qualities of all the modules, except the root, must be at least K. Each module can only contribute its value to the output quality when its allocation is at least its size. In addition, the output quality of an internal node of the DAG is "enabled" only when all its inputs have positive quality, that is, all the

elements smaller than it are included. Therefore the condition that $u' \in U'$ when $u \in U'$ and $u' \prec u$ is satisfied. Finally, since the total allocation is B, $\sum_{u \in U'} s(u) \leqslant B$, and since the output quality is at least K, $\sum_{u \in U'} v(u) \geqslant K$, the answer to the PARTIALLY ORDERED KNAPSACK problem is also positive.

If the answer to the PARTIALLY ORDERED KNAPSACK problem is positive (with knapsack size B and minimal value K), then simply allocate to each computational node $u' \in U'$ an amount of time equal to its size. The definition of the PARTIALLY ORDERED KNAPSACK problem and the transformation to the DAG guarantee that the output quality of each u' would be equal to its value $s(u')$. Hence a minimal output quality of K is guaranteed and the answer to the GCFE problem is also positive.

Since the PARTIALLY ORDERED KNAPSACK problem is NP-complete in the strong sense, and since the above transformation is polynomial, the GCFE problem is NP-complete in the strong sense. □

We now turn to the analysis of a relaxed case of the compilation problem, referred to as tree-structured GCFE. In this case, no repeated subexpressions are allowed and as a result the DAG representation becomes a directed tree. We show that the tree-structured GCFE is NP-complete.

Theorem 4.4. *The tree-structured GCFE problem is* NP-*complete.*

Proof. As in the case of the GCFE problem, the verification problem is polynomial and the problem is therefore NP. The rest of the NP-completeness proof is by transformation from the KNAPSACK problem [10], defined as follows:

- *Instance.* Finite set U, for each $u \in U$ a size $s(u) \in \mathbb{Z}^+$ and a value $v(u) \in \mathbb{Z}^+$, and positive integers B and K.
- *Question.* Is there a subset $U' \subseteq U$ such that $\sum_{u \in U'} s(u) \leqslant B$ and $\sum_{u \in U'} v(u) \geqslant K$?

An instance of the KNAPSACK problem can be transformed into a tree-structured GCFE problem by constructing a binary tree whose leaves are the elements of U. Each element $u \in U$ corresponds to one leaf of the tree (one can add leaf nodes of zero size and value to make the number of leaves an exact power of 2). The performance profile of each leaf node is:

$$Q_u(t) = \begin{cases} v(u), & \text{if } t \geqslant s(u), \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Now, $O(|U|)$ internal nodes are added to construct a complete binary tree. The conditional performance profile of each internal node, w, is the sum of the qualities of its left and right branches:

$$Q_w(q_1, q_2, t) = q_1 + q_2. \tag{13}$$

Note that internal nodes of the tree do not consume any computation time. The output quality, Q_{out}, is the quality of the root node which is the sum of the values of all the the elements of U whose allocation exceeds their size.

It is easy to see that the construction of the tree can be accomplished in polynomial time. To complete the proof, we need to show that the answer to the KNAPSACK problem is "yes" if and only if the answer to the corresponding tree-structured GCFE problem is "yes". This is trivially true when one sets the contract time to B and the minimal output quality to K. The argument is similar to the previous proof. We conclude that the tree-structured GCFE problem is NP-complete. □

The KNAPSACK problem itself is pseudo-polynomial. In fact, the problem can be solved by a simple dynamic programming algorithm. This raises the question of whether the compilation problem of tree-structured expressions is also pseudo-polynomial. The next section identifies the conditions under which the answer to this question is positive.

4.2. Local compilation

Local compilation is the key mechanism in our model to cope with the exponential complexity of global compilation. The idea is to replace a single, complex optimization problem with a set of simple optimization problems whose number grows *linearly* with the size of the program being compiled. If these local optimization problems can be solved in polynomial time, then the total amount of work becomes polynomial.

Definition 4.5. *Local compilation* is the process of optimizing the quality of the output of each programming construct by considering only the performance profiles of its immediate subcomponents.

Local compilation solves the same type of problem as global compilation except for the fact that its scope is limited to one programming structure at a time. While global compilation derives directly the best time allocation to *all* the elementary components, local compilation computes the best time allocation to the immediate subcomponents, treating them as if they were elementary anytime algorithms. If a subcomponent is not elementary, then its performance profile is derived using local compilation as well.

A fundamental question regarding local compilation is the relationship between its result and the result of global compilation. Local compilation is said to be optimal with respect to a particular program structure if it always achieves a globally optimal expected performance. Our first goal in this section is to prove the optimality of local compilation of tree-structured functional expressions under the input monotonicity assumption. Without loss of generality, we will consider binary functions only and assume that the functional expression is a complete binary tree. The leaves of the tree are functions that take input variables as inputs and the internal nodes are functions that take functional expressions as inputs.

Let $f_{i,j}$ denote the jth function on the ith level of the tree. The root node is denoted accordingly by $f_{0,0}$. If the tree is of depth n, then the nodes corresponding to $f_{n,0}, \ldots, f_{n,2^n-1}$ are leaf nodes whose inputs are input variables. For any other node $f_{i,j}, 0 \leqslant i \leqslant n-1, 0 \leqslant j \leqslant 2^i - 1$, the inputs are: $f_{i+1,2j}$ and $f_{i+1,2j+1}$ as shown in Fig. 14.

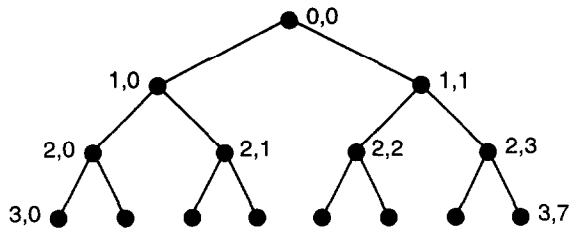


Fig. 14. Tree representation of a functional expression.

Corresponding to each node of the binary tree is a conditional performance profile $Q_{i,j}(q_1, q_2, t)$ which characterizes the output quality for that node as a function of its input qualities, q_1 and q_2, and time allocation t.

Given a functional expression e of depth n, and a particular input quality, the global compilation problem is to find the optimal time allocation to all the nodes of the tree that maximizes the quality of the output of the root node:

$$Q_e^{\mathrm{G}}(t) = arg \max_{t_{i,j}} Q_{0,0}(\cdot), \qquad \sum_{0 \leqslant i \leqslant n} \sum_{0 \leqslant j \leqslant 2^i - 1} t_{i,j} = t, \tag{14}$$

where $Q_{0,0}(\cdot)$ denotes the result of replacing (in the expression e) every function by its conditional performance profile and every input variable by its quality.

We define a local compilation scheme for e by induction on its structure. For a leaf node, the locally compiled performance profile is the conditional performance profile associated with that node:

$$Q_{n,j}^{\mathrm{L}}(t) = Q_{n,j}(q_{n,j,1}, q_{n,j,2}, t), \quad 0 \leqslant j \leqslant 2^n - 1, \tag{15}$$

where $q_{n,j,1}$ and $q_{n,j,2}$ are the qualities of the two inputs of the particular function. For each internal node, the locally compiled performance profile is defined using the performance profiles of its immediate inputs:

$$Q_{i,j}^{\mathrm{L}}(t) = arg \max_{t_1, t_2} \{Q_{i,j}(Q_{i+1,2j}^{\mathrm{L}}(t_1), Q_{i+1,2j+1}^{\mathrm{L}}(t_2), t - t_1 - t_2)\}. \tag{16}$$

Finally, the performance profile of e is denoted by the following expression:

$$Q_e^{\mathrm{L}}(t) = Q_{0,0}^{\mathrm{L}}(t). \tag{17}$$

Note that the external input quality was deliberately omitted in this notation since we focus on the result of local compilation for any *given* input quality. We are now ready to prove the following result:

Theorem 4.6 (Optimality of local compilation of functional expressions). *Let e be a functional expression of an arbitrary depth n whose conditional performance profiles satisfy the input monotonicity assumption, then for any input and total time allocation t:*

$$Q_e^{\mathrm{L}}(t) = Q_e^{\mathrm{G}}(t).$$

Proof. By induction on the depth of the tree. For trees of depth 1 the claim is trivially true because both compilation schemes solve the same optimization problem. Suppose that the claim is true for trees of depth $n-1$ or less. Let e be an expression of depth n, and let $t_{i,j}$ be the allocations to $f_{i,j}$ based on global compilation and resulting in a global optimum. Let t_l and t_r be respectively the total allocation to the left and right subtrees of the root node:

$$t_l = \sum_{i=1}^{n} \sum_{j=0}^{2^{i-1}-1} t_{i,j}, \tag{18}$$

$$t_r = \sum_{i=1}^{n} \sum_{j=2^{i-1}}^{2^i-1} t_{i,j}, \tag{19}$$

$$t = t_l + t_r + t_{0,0}. \tag{20}$$

Then:

$$Q_e^{\mathrm{G}}(t) =$$

By definition and input monotonicity

$$= Q_{0,0}(Q_{1,0}^{\mathrm{G}}(t_l), Q_{1,1}^{\mathrm{G}}(t_r), t_{0,0}) \tag{21}$$

By the induction hypothesis

$$= Q_{0,0}(Q_{1,0}^{\mathrm{L}}(t_l), Q_{1,1}^{\mathrm{L}}(t_r), t_{0,0}) \tag{22}$$

By definition of local compilation

$$\leqslant Q_{0,0}^{\mathrm{L}}(t) \tag{23}$$

By definition

$$= Q_e^{\mathrm{L}}(t) \tag{24}$$

By definition of global compilation

$$\leqslant Q_e^{\mathrm{G}}(t). \tag{25}$$

Hence $Q_e^{\mathrm{L}}(t) = Q_e^{\mathrm{G}}(t)$. □

Since local compilation yields optimal results, it is useful to determine the conditions under which it can reduce the complexity of the compilation problem. Obviously, if each function can take any number of arguments, we cannot guarantee any reduction in complexity. With unbounded number of inputs the depth of the corresponding tree may be one in which case local and global compilation solve the same problem. Hence, we will examine the complexity of local compilation under the *bounded degree assumption* that each node of the tree has a bounded degree. In other words, we assume that the number of inputs to each function is bounded. This assumption only reinforces the principle of modularity that has been long recognized in the development of complex systems.

Given a functional expression of size n and discrete performance profiles with maximal run-time t, we have the following result:

Theorem 4.7. *The tree-structured GCFE problem is polynomial in nt under the input monotonicity and bounded degree assumptions.*

Proof. Since local compilation guarantees optimality under input monotonicity and since local compilation needs to be repeated $O(n)$ times, we only need to show that local compilation of a single node is polynomial in t. This is trivially true under the bounded degree assumption. In particular, if the degree of each node is bounded by k, then the complexity of local compilation is $O(nt^k)$. Unless otherwise mentioned, we will assume in this section that $k = 2$ and that the complexity of local compilation is $O(nt^2)$. □

Note that there is no contradiction between this result and the NP-completeness of the KNAPSACK problem. Both the input monotonicity and the bounded degree assumptions are met by the reduction of Theorem 4.4. However, this does not imply that the KNAPSACK problem is polynomial in n. It does imply that the problem is polynomial in nt (where t represents the maximal element size), and this is already known. The dependency of the algorithm complexity on t is not a problem in our domain for several reasons. First, the range of possible run-times in a real-time system is normally bounded by some constant. Second, the fixed tabular representation of performance profiles allows us to limit the value of t by selecting the "appropriate" time unit for the application. When the performance profile size is bounded by a constant, local compilation can be performed in constant time at each node, and the complexity of the entire process becomes $O(n)$—linear in the program size.

In terms of space requirements, even though local compilation requires $O(n)$ separate performance profiles (one for each internal node of the tree), its total space requirement is only a constant factor more than the space requirement of global compilation. This is due the fact that a globally compiled performance profile must specify the allocation to *each* node of the tree while a locally compiled performance profile needs to specify only the allocation to the immediate successors of each node and to the node itself. To summarize, local compilation has the same space complexity as global compilation but it reduces the time complexity of the optimization problem from exponential to polynomial in nt.

4.3. Additional compositional operators

The family of functional expressions can be enriched with a large set of standard compositional operators. The optimality of local compilation remains valid as long as each operator, ϕ, satisfies two requirements: (1) the operator produces a result whose quality depends on the qualities of its inputs and on the amount of time allocated to the evaluation of the operator itself, t_ϕ; and (2) the conditional performance profile of the operator exhibits input monotonicity. Many useful operators satisfy these requirements. In many cases the evaluation time of such operators is a small constant time and their conditional performance profiles are represented as step functions.

For example, consider the operator *oneof*:

$$F(x) = oneof(M_1(x), \ldots, M_n(x)).$$

The output of *oneof* is the result of its single component, M_i, with the highest quality and its quality is the quality of that component. Suppose that each component, M_i, is an anytime algorithm whose performance profile is Q_i. The conditional performance profile of *oneof* is:

$$Q_{oneof}(q_1, \ldots, q_n, t) = \begin{cases} \max(q_1, \ldots, q_n), & \text{if } t > t_{oneof}, \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

This models a situation in which several alternative methods can be used to solve the same problem. For example, suppose that one needs to transport n identical packages using a certain container. The components of *oneof* might be several alternative bin packing algorithms where the quality of each algorithm is measured by the portion of the container's volume filled with packages. Obviously, the maximal volume that can be transported is proportional to the maximal quality among all the individual bin packing algorithms. Additional examples of such compositional operators appear in [34].

4.4. Repeated subexpressions

Local compilation does not produce good results when applied to functional expressions with repeated subexpressions. Using the tree representation, a repeated subexpression corresponds to a repeated subtree. The problem with local compilation is that it allocates computation time to *all* the nodes of the tree while time should be allocated only once to evaluate all the copies of a repeated subexpression. For example, consider the functional expression that appears in Fig. 13.

$$F(x) = E(D(B(A(x)), C(A(x)))).$$

The subexpression $A(x)$ appears twice and an efficient compiler should not allocate time to both copies. This means, however, that the allocation of time to $A(x)$ cannot be done locally, since it affects the output qualities of both C and D.

In this section, we present three time allocation methods that deal with general functional expressions:

- HILL-CLIMBING-ALLOCATION finds a solution to the global compilation problem directly, but does not guarantee global optimality.
- CONDITIONING-ALLOCATION tries all possible allocations to the repeated subexpressions, then applies local compilation to the resulting trees (note the analogy to conditioning methods in belief network evaluation [25]).
- TRADING-ALLOCATION begins with the allocation determined by local compilation, and then trades time among components so that only one copy of each repeated subexpression ends up with a nonzero allocation.

All three methods where developed using the discrete tabular representation of performance profiles. The complexity and optimality of the three methods are discussed below. For each algorithm we will compute the complexity of calculating *each* entry of

```
HILL-CLIMBING-ALLOCATION
for each Q_in ∈ [Q_L .. Q_U] do
  for each T ∈ [T_L .. T_U] do
    s ← INITIAL-RESOLUTION(T)
    t_i ← T/n   ∀i: 1 ≤ i ≤ n
    repeat
      while ∃i, j such that
        E(Q_out(Q_in, t_1, ..., t_i − s, ..., t_j + s, ..., t_n)) >
        E(Q_out(Q_in, t_1, ..., t_n))
        let i, j be the ones that maximize expected quality
        t_i ← t_i − s
        t_j ← t_j + s
      s ← s/2
    until s < ε
    T[Q_in, T] ← (t_1, ..., t_n)
```

Fig. 15. Time allocation using a hill-climbing search.

the table representing the performance profile (that is, the complexity of calculating the optimal allocation to the components for any particular input quality and total run-time).

4.4.1. Time allocation using a hill-climbing search

This time allocation algorithm uses the DAG representation of functional expressions. For each particular time allocation to the components of a DAG, the quality of the output can be computed using the conditional performance profiles of the components. This computation can be performed in linear time in the size of the graph. While the search space of all possible time allocations has exponential size, an efficient hill-climbing search procedure can be constructed by limiting the search space.

The time allocation algorithm, shown in Fig. 15, starts with an equal amount of time allocated to each component of the DAG. Then it considers trading s time units between two modules so as to increase the expected quality of the output. As long as it can improve the expected quality, it trades s time units between the two modules that have maximal effect on output quality. When no such improvement is possible with the current value of s, it divides s by 2 until s reaches a certain minimal value, ϵ. At that point, it reaches a local maximum and returns the best time allocation it found. As with any hill-climbing algorithm, it suffers from the problem of converging on a local maximum. An analysis of the algorithm shows that simple properties of the conditional performance profiles of the components, such as monotonicity, are not sufficient to guarantee global optimality.

Complexity

Let κ be the size of the functional expression (i.e. the number of nodes in the corresponding DAG), and let $\tau = T_{\max}/\epsilon$ be the maximal number of discrete time units to be allocated. The complexity of the algorithm is then $O(\kappa^3 \log \tau)$. This is due to

```
CONDITIONING-ALLOCATION
for each Q_in ∈ [Q_L .. Q_U] do
  for each T ∈ [T_L .. T_U] do
    Q_max ← 0
    r_max ← 0
    for r ← 0 to T step ε
      t ← T − r
      ADJUST-PP(r)
      APPLY-LOCAL-COMPILATION(e, t)
      Q ← Q_out(Q_in, (r | t_1, ..., t_{n−m}))
      if Q > Q_max then do
        Q_max ← Q
        A_opt ← (r | t_1, ..., t_{n−m})
    T[Q_in, T] ← A_opt
```

Fig. 16. Time allocation with pre-determined time to repeated subexpressions.

the fact that for each search resolution s, the algorithm needs to find the optimal pair of modules for trading time. This is done in $O(\kappa^2)$ by considering every possible pair. This step repeats only a constant number of times. Finding the expected quality of the output is performed in $O(\kappa)$ and the number of time resolution steps is $O(\log \tau)$.

4.4.2. Pre-determined allocation to repeated subexpressions

The second method, CONDITIONING-ALLOCATION, is based on fixing the allocation to each repeated subexpression before computing the allocation to the other components. The allocation to the other components is determined based on standard local compilation. Time allocation is made only once to all the copies of each repeated subexpression. Once that allocation is decided, the complete expression is treated as a tree rather than a DAG and the efficient local compilation scheme is used.

Let e be a functional expression of size n. Assume that e has only one repeated subexpression e' that appears $m > 1$ times in e. The copies of e' are denoted by $e'_1, \ldots, e'_m$. Let $(r \mid t_1, \ldots, t_{n-m}))$ represent allocation of r time units to $e'_1, \ldots, e'_m$ and $t_1, \ldots, t_{n-m}$ to the remaining $n-m$ modules. Fig. 16 shows the time allocation algorithm. Its central idea is to *reserve* a certain amount of time r, out of the total allocation t, for evaluating a single copy of the repeated subexpression e'. All the other copies "enjoy for free" the result of this evaluation. The fact that r time units are reserved for e' is communicated to the local compilation process by adjusting the performance profile of e'. The new performance profile is a step function that returns quality $Q_{e'}(r)$ at zero time and provides no further improvement of quality. Since no improvement of quality is possible, an optimal schedule would not allocate time to any of the copies and hence standard local compilation is guaranteed to allocate the remaining time optimally to the *other* components. The algorithm performs a search to find the best pre-determined reserved time r for which the output quality is maximal.

If the conditional performance profiles of all the components of e satisfy the input monotonicity assumption, then any optimal schedule has the following property:

Lemma 4.8. *Any optimal schedule for the evaluation of e allocates time to a single copy of e'.*

Proof. Suppose that there is an optimal schedule in which more than one copy of e' is evaluated. Let $r_1, \ldots, r_m$ be the allocations to the m copies, and let $r = \sum r_i$. By the monotonicity of the performance profile of e', the quality achieved by allocating r time units to a single copy is greater than any of the qualities achieved with allocations $r_1, \ldots, r_m$. Hence, by substituting the result of that single copy for all the copies without changing the allocation to the other components, and by the monotonicity of the conditional performance profiles, it is apparent that the output quality would increase. This contradicts the optimality of the original schedule. Therefore, time must be allocated to a single copy only. □

CONDITIONING-ALLOCATION can be viewed as a two phase optimization process. Its first phase determines the optimal r and its second phase finds the optimal allocation to the other components. Having established the fact that any optimal schedule must activate e' only once, we can conclude the global optimality of this method:

Theorem 4.9 (Optimality of CONDITIONING-ALLOCATION). *Let e be a functional expression with a single repeated subexpression e', then* CONDITIONING-ALLOCATION *returns a globally optimal schedule for evaluating e.*

Proof. An immediate result of Lemma 4.8 and the optimality of local compilation. □

Complexity

Again, let κ be the size of the functional expression and let $\tau = T_{\max}/\epsilon$ be the maximal number of time units to be allocated. The complexity of the algorithm is then $O(\kappa\tau^3)$. This is due to the fact that the complexity of the search for the optimal value of r is $O(\tau)$ and the most complicated step inside the loop is local compilation with complexity $O(\kappa\tau^2)$.

To extend this method to work with p different repeated subexpressions, the algorithm must consider any possible pre-determined allocation to (single copies of) each repeated subexpression. The complexity of this step is $O(\tau^p)$ when $p \ll \tau$. The overall complexity becomes $O(\kappa\tau^{(p+2)})$.

4.4.3. Learning the allocation to repeated subexpressions

The third method, TRADING-ALLOCATION, is based on learning the allocation to repeated subexpressions through standard local compilation. To be able to apply local compilation, the algorithm first ignores the repetition of subexpressions and uses standard local compilation. Then it applies a series of performance profile adjustments followed by local compilation. The process converges on a single allocation to each repeated subexpression.

```
TRADING-ALLOCATION
for each Q_in ∈ [Q_L .. Q_U] do
  for each T ∈ [T_L .. T_U] do
    r ← 0
    repeat
      t ← T − r
      SHIFT-PP(r)
      APPLY-LOCAL-COMPILATION(e, t)
      Let r_1, ..., r_m be the allocations to e_1, ..., e_m
      r ← r + max{r_i}
    until Σ r_i = 0
    T[Q_in, T] ← (r | t_1, ..., t_n)
```

Fig. 17. Learning the allocation to repeated subexpressions.

Again, let e be a functional expression. As with CONDITIONING-ALLOCATION, we consider first the case where e has only one repeated subexpression e' with copies $e'_1, \ldots, e'_m$. Fig. 17 shows the time allocation algorithm. It learns the allocation r to a single copy of e'. Starting with $r = 0$, the algorithm repeatedly increases r until local compilation allocates no additional time to the copies of e'. In each iteration, the current value of r is used to determine how much time to reserve for evaluating e'. The fact that r time units are reserved for e' is communicated to the local compilation process by adjusting the performance profile of e'. The time origin of the performance profile is shifted r units to the right. Standard local compilation is then applied and the optimal allocation to *all* the components is computed. Suppose that, based on the adjusted performance profile, the allocations to the m copies of e' are $r_1, \ldots, r_m$. Then, the maximal allocation among those is used to increase the value of r. This process is repeated until no additional time is allocated to any of the copies beyond the reserved time allocation r. This time allocation algorithm does not guarantee global optimality [34].

Complexity

Using the same notation as above, the complexity of the algorithm is $O(\kappa\tau^3)$. This is due to the fact that the complexity of the search for r is $O(\tau)$ (since r may be incremented by 1 unit of time in each iteration). The most complicated step inside the loop is local compilation with complexity $O(\kappa\tau^2)$. Note that in practice the convergence of the search for r is much faster than $O(\tau)$.

The extension to multiple repeated expressions is straightforward. The algorithm needs to maintain a sequence of reserved allocations for each repeated subexpression. The rest of the algorithm remains the same. The advantage of TRADING-ALLOCATION is that its complexity grows only linearly with the number of repeated subexpressions, p. This is due to the fact that a single loop is used to update *all* the reserved allocations to repeated subexpressions and the worst case complexity of that loop is only $O(p\tau)$. Hence the overall complexity in the general case is $O(\kappa p\tau^3)$.

Summary

We have examined three time allocation algorithms designed to cope with the difficulty of compiling general functional expressions. HILL-CLIMBING-ALLOCATION has a complexity $O(\kappa^3 \log \tau)$ and finds only local optimum. CONDITIONING-ALLOCATION has complexity $O(\kappa\tau^{(p+2)})$ and TRADING-ALLOCATION $O(\kappa p\tau^3)$. When $\kappa \ll \tau$ the first algorithm is the most efficient one. The second method guarantees optimality, but its complexity grows exponentially with the number of repeated subexpressions. To address this problem, the last method can be used. Its complexity grows only linearly with the number of repeated subexpressions but it does not guarantee global optimality. By using local compilation to determine the allocation to the rest of the components, TRADING-ALLOCATION is more likely to converge on the global optimum than HILL-CLIMBING-ALLOCATION.

5. Conclusion

This paper examines the possibility of extending the advantages of anytime algorithms to the construction of complex real-time systems. The first results on this vital problem show that a modular composition of anytime algorithms can be implemented efficiently. In particular, we show that:

(1) The performance profile of a composite system can be derived automatically and efficiently by off-line compilation techniques. The compilation process optimizes the overall quality of the system as a *contract* algorithm.

(2) The resulting system can be made interruptible with only a small, constant penalty.

(3) Our approach separates two central aspects of system development, namely the construction of the performance components and the optimization of performance. In real-time system construction this separation isolates each module from the time constraints that it must satisfy. As a result, our compilation mechanism simplifies the design of real-time systems and allows for modularity and abstraction to be applied.

(4) The resulting real-time system is machine-independent in the sense that it can adapt its internal time allocation to the available computational resources.

The main contribution of the paper includes: (1) formalizing the compilation problem and solving it for the case of functional composition; (2) making the interruptible/contract distinction that facilitates a two-step solution to the compilation problem; and (3) formalizing the notion of conditional performance profiles that allow us to solve a large part of the problem off-line.

Further work in this area is currently aimed at: (1) developing larger applications to further evaluate the components of the model; (2) extending the scope of compilation by studying the compilation of additional programming structures; (3) extending the scope of anytime algorithms to include anytime sensing and anytime action; and (4) building a programming environment to support anytime algorithm development. Our ultimate goal is to construct robust real-time systems in which perception, deliberation and action are governed by a collection of anytime algorithms.

Acknowledgements

The authors would like to thank the members of Shlomo Zilberstein's dissertation committee—Alice Agogino, Tom Dean and Susan Graham. Tom Dean in particular inspired an initial interest in anytime algorithms and has continued to provide important insights and comments. Thanks also to the members of the RUGS seminar—Francesca Barrientos, Othar Hansson, Tim Huang, Andrew Mayer, Ron Musick, Gary Ogasawara and Ron Parr. This research was partially supported by the National Science Foundation under grants IRI-8903146 and IRI-9058427 and by the Malcolm R. Stacey Fellowship to Shlomo Zilberstein.

References

[1] M. Boddy, Anytime problem solving using dynamic programming, in: *Proceedings AAAI-91*, Anaheim, CA (1991) 738-743.

[2] M. Boddy and T.L. Dean, Solving time-dependent planning problems, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 979-984.

[3] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor and D. Freeman, Autoclass: a Bayesian classification system, in: *Proceedings Fifth International Conference on Machine Learning*, Ann arbor, MI (1988).

[4] B. D'Ambrosio, Resource bounded agents in an uncertain world, in: *Working Notes of the IJCAI-89 Workshop on Real-Time Artificial Intelligence Problems*, Detroit, MI (1989).

[5] R. Davis, Meta-rules: reasoning about control, *Artif. Intell.* **15** (1980) 179-222.

[6] T.L. Dean, Intractability and time-dependent planning, in: M.P. Georgeff and A.L. Lansky, eds., *Proceedings of the 1986 Workshop on Reasoning about Actions and Plans*, Los Altos, CA (Morgan Kaufmann, 1987).

[7] T.L. Dean and M. Boddy, An analysis of time-dependent planning, in: *Proceedings AAAI-88*, St. Paul, MN (1988) 49-54.

[8] J. Doyle, Rationality and its roles in reasoning, in: *Proceedings AAAI-89*, Boston, MA (1990) 1093-1100.

[9] C. Elkan, Incremental, approximate planning: abductive default reasoning, in: *Working Notes of the AAAI Spring Symposium on Planning in Uncertain Environments*, Palo Alto, CA (1990).

[10] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, CA, 1979).

[11] A. Garvey and V. Lesser, Design-to-time real-time scheduling, in: *IEEE Trans. Syst. Man Cybern.* **23** (6) (1993).

[12] M.R. Genesereth, An overview of metalevel architectures, in: *Proceedings AAAI-83*, Washington, DC (1983) 119-123.

[13] J.A. Hendler, Real-time planning, in: *Working Notes of the AAAI Spring Symposium on Planning and Search*, Stanford, CA (1989).

[14] E.J. Horvitz, Reasoning about beliefs and actions under computational resource constraints, in: *Proceedings 1987 Workshop on Uncertainty in Artificial Intelligence*, Seattle, WA (1987).

[15] E.J. Horvitz and J.S. Breese, Ideal partition of resources for metareasoning, Technical Report KSL-90-26, Stanford Knowledge Systems Laboratory, Stanford, CA (1990).

[16] E.J. Horvitz, H.J. Suermondt and G.F. Cooper, Bounded conditioning: flexible inference for decision under scarce resources, in: *Proceedings 1989 Workshop on Uncertainty in Artificial Intelligence*, Windsor, Ont. (1989) 182-193.

[17] R.M. Karp, An introduction to randomized algorithms, Technical Report TR-90-024, International Computer Science Institute, Berkeley, CA (1990).

[18] R.E. Korf, Depth-first iterative-deepening: An optimal admissible tree search, *Artif. Intell.* **27** (1985) 97-109.

[19] R.E. Korf, Real-time heuristic search, *Artif. Intell.* **42** (3) (1990) 189-212.

[20] E.L. Lawler et al., eds., *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization* (Wiley, New York, 1987).

[21] V. Lesser, J. Pavlin and E. Durfee, Approximate processing in real-time problem-solving, *AI Mag.* **9** (1) (1988) 49–61.

[22] S. Lin and B.W. Kernighan, An effective heuristic algorithm for the Traveling Salesman Problem, *Oper. Res.* **21** (1973) 498–516.

[23] T. Lozano-Pérez and R.A. Brooks, An approach to automatic robot programming, in: M.S. Pickett and J.W. Boyse, eds., *Solid Modeling by Computers* (Plenum, New York, 1984) 293–327.

[24] R.S. Michalski and P.H. Winston, Variable precision logic, *Artif. Intell.* **29** (2) (1986) 121–146.

[25] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, Los Altos, CA, 1988).

[26] A. Pos, Time-constrained model-based diagnosis, Master Thesis, Department of Computer Science, University of Twente, Enschede, Netherlands (1993).

[27] S.J. Russell, D. Subramanian and R. Parr, Provably bounded optimal agents, in: *Proceedings IJCAI-93*, Chambery, France (1993) 338–344.

[28] S.J. Russell and E.H. Wefald, Principles of metareasoning, in: R.J. Brachman et al., eds., *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning* (Morgan Kaufmann, San Mateo, CA, 1989).

[29] S.J. Russell and E.H. Wefald, *Do the Right Thing: Studies in limited rationality*, (MIT Press, Cambridge, MA, 1991).

[30] S.J. Russell and S. Zilberstein, Composing real-time systems, in: *Proceedings IJCAI-91*, Sydney, Australia (1991) 212–217.

[31] H.A. Simon, *Models of Bounded Rationality, Vol. 2* (MIT Press, Cambridge, MA, 1982).

[32] S.V. Vrbsky and J.W.S. Liu, Producing monotonically improving approximate answers to database queries, in: *Proceedings IEEE Workshop on Imprecise and Approximate Computation*, Phoenix, AZ (1992) 72–76.

[33] L.A. Zadeh, Fuzzy logic and approximate reasoning, *Synthese* **30** (1975) 407–428.

[34] S. Zilberstein, Operational rationality through compilation of anytime algorithms, Ph.D. Dissertation, Computer Science Division, University of California, Berkeley, CA (1993).

[35] S. Zilberstein and S.J. Russell, Efficient resource-bounded reasoning in AT-RALPH, in: *Proceedings First International Conference on AI Planning Systems*, College Park, MD (1992) 260–266.

[36] S. Zilberstein and S.J. Russell, Constructing utility-driven real-time systems using anytime algorithms, in: *Proceedings IEEE Workshop on Imprecise and Approximate Computation*, Phoenix, AZ (1992) 6–10.

[37] S. Zilberstein and S.J. Russell, Anytime sensing, planning and action: A practical model for robot control, in: *Proceedings IJCAI-93*, Chambery, France (1993) 1402–1407.

ELSEVIER

Artificial Intelligence 82 (1996) 215–235

Artificial Intelligence

Knowledge-level analysis of belief base operations

Sven Ove Hansson*

Uppsala University, Department of Philosophy, Villavägen 5, S-752 36, Uppsala, Sweden

Received November 1993; revised November 1994

Abstract

Recent developments in philosophical studies of belief revision have led to the construction of models that are much closer to actual computing systems than the earlier models. An operation of contraction (revision) on a logically closed belief set can be obtained by assigning to it a finite belief base and an operation of contraction (revision) on that base. Exact characterizations of such base-generated operations have been obtained. In this way, computational tractability and other realistic properties of operations on finite bases can be combined with a precise analysis that is independent of the symbolic representation.

1. Introduction

The dominating formal model of belief revision and database updating is the AGM model, so called after the three authors, Carlos Alchourrón, Peter Gärdenfors and David Makinson, of the seminal paper in which this model was first presented [1]. In the AGM model, belief states are represented by logically closed sets of sentences, and operations of change are performed on these sets.

A major alternative framework is that of belief bases. These are sets of sentences that—at least as a rough approximation—contain those sentences that represent the explicit information from which all other information is derived. Operations of change are performed on the belief base, and non-explicit information is changed only as the result of changes of the base. Belief bases have the advantages of computational tractability and of a greater expressive power that allows for realistic logical properties that are more difficult to obtain with belief set models.

One of the most important advantages of the AGM model is its representation theorems that exactly characterize the properties of its operations. Until recently,

* E-mail: filsoh@strix.udac.uu.se.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0004-3702(95)00005-4

no corresponding characterization was available of the properties, on the knowledge level, of belief base operations. In this paper, such characterizations are reported, and their significance for computer science are discussed.

2. The AGM model

In the AGM model, an individual's belief state (or the database state of an artificial system) is represented by a set $\boldsymbol{K}$ of sentences, the *belief set*. The belief set is closed under logical consequence. In other words, you are supposed to believe in all the logical consequences of what you believe.

To express logical closure in the formal language, a *consequence operator* Cn is used. For any set A of sentences, $\mathrm{Cn}(A)$ is the set of sentences that follow logically from A. A set A is closed under logical consequence if and only if $A = \mathrm{Cn}(A)$.

Cn satisfies the standard conditions for a consequence operator, namely inclusion ($A \subseteq \mathrm{Cn}(A)$), monotony (if $A \subseteq B$, then $\mathrm{Cn}(A) \subseteq \mathrm{Cn}(B)$), and iteration ($\mathrm{Cn}(A) = \mathrm{Cn}(\mathrm{Cn}(A))$). Furthermore, it satisfies the properties of supraclassicality (if α follows by classical truth-functional logic from A, then $\alpha \in \mathrm{Cn}(A)$), deduction (if $\beta \in \mathrm{Cn}(A \cup \{\alpha\})$, then $(\alpha \rightarrow \beta) \in \mathrm{Cn}(A)$), and compactness (if $\alpha \in \mathrm{Cn}(A)$, then $\alpha \in \mathrm{Cn}(A')$ for some finite set $A' \subseteq A$). $A \vdash \alpha$ will be used as an alternative notation for $\alpha \in \mathrm{Cn}(A)$.

Logical closure is not satisfied by actual doxastic agents. We do not believe in all the consequences of our beliefs, for the simple reason that we are unable to make all the inferences and draw all the conclusions that this would require. Isaac Levi has clarified this by pointing out that a belief set consists of the sentences that someone is *committed to believe in*, not those that she actually believes in [19,20]. According to Levi, you are doxastically committed to believe in all the logical consequences of your beliefs, but typically your performance does not fulfil this commitment. Belief sets (Levi's "corpora") represent your commitments.

Changes in belief are represented by operations on the belief set. There are three types of change. By an *expansion* is meant that a new proposition, typically consistent with a given belief set, is set-theoretically added to that belief set, without anything else being retracted. In *contraction*, a proposition which was earlier in a belief set is rejected from it. By *revision* is meant that a proposition is added under the condition that the revised belief set be consistent and closed under logical consequence [1, p. 510].

Operations on belief sets should preserve logical closure. This is the reason why expansion is defined as $\mathrm{Cn}(\boldsymbol{K} \cup \{\alpha\})$ rather than the simpler $\boldsymbol{K} \cup \{\alpha\}$.

Definition 1 (Isaac Levi [19]). Let $\boldsymbol{K}$ be a belief set and α a sentence. $\boldsymbol{K} + \alpha$ ("$\boldsymbol{K}$ expanded by α"), *the* (*closing*) *expansion* of $\boldsymbol{K}$ by α, is defined as follows:

$$\boldsymbol{K} + \alpha = \mathrm{Cn}(\boldsymbol{K} \cup \{\alpha\}) .$$

The standard AGM operator of contraction is *partial meet contraction*. It is based on a selection among the maximal subsets of the belief set that do not imply the sentence to be discarded. The outcome of partial meet contraction by a sentence α is equal to the intersection of the set of selected maximal subsets of the original set that do not imply α.

Definition 2 [2]. Let A be a set of sentences and α a sentence. The set $A \perp \alpha$ ("A less alpha") is the set such that $B \in A \perp \alpha$ if and only if:
(1) $B \subseteq A$,
(2) $\alpha \notin \mathrm{Cn}(B)$,
(3) there is no set B' such that $B \subset B' \subseteq A$ and $\alpha \notin \mathrm{Cn}(B')$.

Definition 3 [1]. Let A be a set of sentences. A *selection function* for A is a function γ such that for all sentences α:
(1) if $A \perp \alpha$ is non-empty, then $\gamma(A \perp \alpha)$ is a non-empty subset of $A \perp \alpha$, and
(2) if $A \perp \alpha$ is empty, then $\gamma(A \perp \alpha) = \{A\}$.

Definition 4 [1]. Let A be a set of sentences and γ a selection function for A. The *partial meet contraction* on A that is generated by γ is the operation $\sim_\gamma$ such that for all sentences α:

$$A \sim_\gamma \alpha = \bigcap \gamma(A \perp \alpha).$$

An operation $\div$ on A is a partial meet contraction if and only if there is a selection function γ such that for all sentences α: $A \div \alpha = A \sim_\gamma \alpha$.

A variety of special cases of partial meet contraction are referred to in the AGM tradition. In particular, cases in which the selection function is based on a relation have turned out to have interesting formal properties.

Definition 5. A selection function γ for a set A, and its associated operator $\sim_\gamma$ of partial meet contraction, are:
(1) *maxichoice* if and only if for all α, $\gamma(A \perp \alpha)$ has exactly one element [3];
(2) *full meet* if and only if for all α, $A \perp \alpha \subseteq \gamma(A \perp \alpha)$ [3];
(3) *relational* if and only if there is a relation $\sqsubseteq$ such that for all sentences α, if $A \perp \alpha$ is non-empty, then [1]

$$\gamma(A \perp \alpha) = \{B \in A \perp \alpha \mid C \sqsubseteq B \text{ for all } C \in A \perp \alpha\};$$

(4) *transitively relational* if and only if γ is based in that manner on a relation $\sqsubseteq$ that is transitive [1];
(5) *transitively maximizingly relational* (TMR) if and only if γ is based in that manner on a relation $\sqsubseteq$ that is transitive and also satisfies the maximizing property:

If $A \subset B$, then $A \sqsubset B$

(where $\sqsubset$ is the strict part of $\sqsubseteq$) [13].

It has been shown that an operator of partial meet contraction on a (logically closed) belief set is transitively maximizingly relational (TMR) if and only if it is transitively relational [14].

Revision is the consistency-preserving incorporation of a new belief. In the AGM model, the revision of a belief set $\boldsymbol{K}$ by a sentence α takes place in two steps:

(1) contract $\boldsymbol{K}$ by $\neg\alpha$,

(2) expand the resulting belief set by α.

In the first step, enough of $\boldsymbol{K}$ is removed to ensure that it no longer contradicts α (which a belief set does if and only if it contains $\neg\alpha$). After that, α can be added through expansion, which is done in the second step.

Letting $*$ stand for revision, this procedure can be summarized in the form of an equation:

$$\boldsymbol{K} * \alpha = (\boldsymbol{K} \div \neg\alpha) + \alpha \ .$$

This is the *Levi identity* [3, 6]. In the AGM model, operators of revision are derived from operators of partial meet contraction via the Levi identity:

Definition 6 [1]. Let $\boldsymbol{K}$ be a belief set and γ a selection function for $\boldsymbol{K}$. The operator $\mp_\gamma$ such that

$$\boldsymbol{K} \mp_\gamma \alpha = (\boldsymbol{K} \sim_\gamma \neg\alpha) + \alpha$$

for all sentences α is the operator of *partial meet revision* on $\boldsymbol{K}$ that is based on γ.

$\mp_\gamma$ stands in the same relation to $*$ as $\sim_\gamma$ to $\div$. We use $*$ ($\div$) to denote operators of revision (contraction) in general, and $\mp_\gamma$ ($\sim_\gamma$) to denote the operator that is generated by a particular selection function γ.

Most logicians and philosophers who have studied belief revision have focused on operators of contraction, whereas computer scientists have been primarily interested in operators of revision. The Levi identity provides a firm connection between the two approaches.

The contraction and revision operators of AGM owe much of their attractiveness to a set of representation theorems. The following postulates are used to characterize contraction:

(G $\div$ 1) $\boldsymbol{K} \div \alpha$ is a theory if $\boldsymbol{K}$ is a theory (closure).

(G $\div$ 2) $\boldsymbol{K} \div \alpha \subseteq \boldsymbol{K}$ (inclusion).

(G $\div$ 3) If $\alpha \notin \mathrm{Cn}(\boldsymbol{K})$, then $\boldsymbol{K} \div \alpha = \boldsymbol{K}$ (vacuity).

(G $\div$ 4) If $\alpha \notin \mathrm{Cn}(\emptyset)$, then $\alpha \notin \mathrm{Cn}(\boldsymbol{K} \div \alpha)$ (success).

(G $\div$ 5) If $\alpha \leftrightarrow \beta = \mathrm{Cn}(\emptyset)$, then $\boldsymbol{K} \div \alpha = \boldsymbol{K} \div \beta$ (extensionality).

(G $\div$ 6) $\boldsymbol{K} \subseteq \mathrm{Cn}((\boldsymbol{K} \div \alpha) \cup \{\alpha\})$ whenever $\boldsymbol{K}$ is a theory (recovery).

(G $\div$ 7) $(\boldsymbol{K} \div \alpha) \cap (\boldsymbol{K} \div \beta) \subseteq \boldsymbol{K} \div (\alpha \& \beta)$ (conjunctive overlap).

(G $\div$ 8) If $\alpha \notin \boldsymbol{K} \div (\alpha \& \beta)$, then $\boldsymbol{K} \div (\alpha \& \beta) \subseteq \boldsymbol{K} \div \alpha$ (conjunctive inclusion).

An operator $\div$ on a (logically closed) belief set $\boldsymbol{K}$ is a partial meet contraction if and only if it satisfies (G $\div$ 1)–(G $\div$ 6), the "basic Gärdenfors postulates". It is

transitively relational if and only if it also satisfies (G ÷ 7)–(G ÷ 8), the "supplementary Gärdenfors postulates" [1].

The following postulates are used to characterize partial meet revision:

(G∗1) $\boldsymbol{K} * \alpha$ is a belief set (closure).

(G∗2) $\alpha \in \boldsymbol{K} * \alpha$ (success).

(G∗3) $\boldsymbol{K} * \alpha \subseteq \mathrm{Cn}(\boldsymbol{K} \cup \{\alpha\})$ (inclusion).

(G∗4) If $\neg\alpha \notin \boldsymbol{K}$, then $\mathrm{Cn}(\boldsymbol{K} \cup \{\alpha\}) \subseteq \boldsymbol{K} * \alpha$ (vacuity).

(G∗5) $\boldsymbol{K} * \alpha$ is inconsistent if and only if α is inconsistent (consistency).

(G∗6) If $(\alpha \leftrightarrow \beta) \in \mathrm{Cn}(\emptyset)$, then $\boldsymbol{K} * \alpha = \boldsymbol{K} * \beta$ (extensionality).

(G∗7) $\boldsymbol{K} * (\alpha \& \beta) \subseteq \mathrm{Cn}((\boldsymbol{K} * \alpha) \cup \{\beta\})$.

(G∗8) If $\neg\beta \notin \boldsymbol{K} * \alpha$, then $\mathrm{Cn}((\boldsymbol{K} * \alpha) \cup \{\beta\}) \subseteq \boldsymbol{K} * (\alpha \& \beta)$.

An operator $*$ on a (logically closed) belief set $\boldsymbol{K}$ is a partial meet revision if and only if it satisfies (G∗1)–(G∗6), the "basic Gärdenfors postulates". It is transitively relational if and only if it in addition satisfies (G∗7)–(G∗8), the "supplementary Gärdenfors postulates".

3. Weaknesses of the AGM model

The identification of belief states with belief sets has non-negligible disadvantages, in particular from a computational point of view. A belief set is a very large entity. For any two sentences α and β, if α is included in the belief set, then so are both $\alpha \vee \beta$ and $\alpha \vee \neg\beta$, since they are logical consequences of α. Therefore, if the language is sufficiently rich, then the belief set contains myriads of sentences that the believer has never thought of. In particular, if the language is infinite, then the belief set will contain an infinite number of sentences.

It seems unnatural for changes to be performed on such large entities as belief sets, that contain all kinds of irrelevant and never-thought-of sentences. This alone is a sufficient reason to search for alternatives to the AGM model. Another reason for this is that one of the characteristic properties of partial meet contraction, namely recovery, is "open to query from the point of view of acceptability under its intended reading" [21]. Several authors have argued against recovery as a general principle of belief contraction [5, 11, 14, 18, 20, 24]. The following two examples indicate that recovery does not always hold:

Example [11]. I have read in a book about Cleopatra that she had both a son and a daughter. My set of beliefs therefore contains both ϕ and ψ, where ϕ denotes that Cleopatra had a son and ψ that she had a daughter. I then learn from a knowledgeable friend that the book is in fact a historical novel. After that I contract $\phi \vee \psi$ from my set of beliefs, i.e., I do not any longer believe that Cleopatra had a child. Soon after that, however, I learn from a reliable source that Cleopatra had a child. It seems perfectly reasonable for me to then add $\phi \vee \psi$ to my set of beliefs without also reintroducing either ϕ or ψ.

Example. I previously entertained the two beliefs "George is a criminal" (α) and

"George is a mass murderer" (β). When I received information that induced me to give up the first of these beliefs (α), the second (β) had to go as well (since α would otherwise follow from β).

I then received new information that made me accept the belief "George is a shoplifter" (δ). The resulting new belief set is the expansion of $\boldsymbol{K} \div \alpha$ by δ, $(\boldsymbol{K} \div \alpha) + \delta$. Since α follows from δ, $(\boldsymbol{K} \div \alpha) + \alpha$ is a subset of $(\boldsymbol{K} \div \alpha) + \delta$. By recovery, $(\boldsymbol{K} \div \alpha) + \alpha$ includes β, from which follows that $(\boldsymbol{K} \div \alpha) + \delta$ includes β.

Thus, since I previously believed George to be a mass murderer, I cannot any longer believe him to be a shoplifter without believing him to be a mass murderer.

4. Belief bases

As an alternative to belief sets, belief states can be represented by sets of sentences that are *not* closed under logical consequence. Such sets are called *belief bases*.

Definition 7. Any set A of sentences is a *belief base*.

Let $\boldsymbol{K}$ be a belief set. Then a set A of sentences is a *belief base* for $\boldsymbol{K}$ if and only if $\boldsymbol{K} = \mathrm{Cn}(A)$.

The criterion for a sentence α to be believed is that it is a consequence of the belief base, $\alpha \in \mathrm{Cn}(A)$. The elements of the belief base are the *basic beliefs*, and the elements of its logical closure that are not elements of the belief base itself are the (merely) *derived beliefs*.

Belief bases are not required by definition to be finite, but in all realistic applications they will be so.

Changes are performed on the belief base. Although we (are committed to) believe the logical consequences of our basic beliefs, these consequences are subject only to exactly those changes that follow from changes of the basic beliefs. The underlying intuition is that the merely derived beliefs are not worth retaining for their own sake. If one of them loses the support that it had in basic beliefs, then it will automatically be discarded.

Example. Let α denote that I have money in my bank account, and β that I own a large fortune in stocks and shares. For good reasons, I believe that α is true and β is wrong. As a consequence of my belief in α, I also believe that $\alpha \vee \beta$ is true. One day, may bank informs me that my account is empty. I therefore lose my belief in α. If I am a sensible person, I will then also lose my belief in $\alpha \vee \beta$.

- *Belief base approach:* $\alpha \vee \beta$ is automatically lost when it is no longer implied by the belief base.
- *Belief set approach:* The retraction of $\alpha \vee \beta$ has to be ensured by the use of a selection mechanism that determines (among other things) whether to retain $\alpha \vee \beta$ or $\alpha \vee \neg\beta$.

Thus, when elements of a belief base are removed, this leads to disbelief in all sentences that depended on the removed elements. This is the process of "disbelief propagation" that has been discussed by computer scientists [22].

One and the same belief set can be represented by different belief bases. In this sense, belief bases have more expressive power than belief sets. As an example, the two belief bases $\{\alpha, \beta\}$ and $\{\alpha, \alpha \leftrightarrow \beta\}$ have the same logical closure, since $\mathrm{Cn}(\{\alpha, \beta\}) = \mathrm{Cn}(\{\alpha, \alpha \leftrightarrow \beta\})$. Nevertheless, these belief bases are not identical. They are *statically equivalent*, in the sense of representing the same beliefs. On the other hand, the following example shows that they are not *dynamically equivalent* in the sense of behaving in the same way under operations of change. They can therefore be taken to represent different ways of holding the same beliefs.

Example. Let α denote that the Liberal Party will support the proposal to subsidize the steel industry, and let β denote that Ms. Smith, who is a liberal MP, will vote in favour of that proposal.

Abe has the basic beliefs α and β, whereas Bob has the basic beliefs α and $\alpha \leftrightarrow \beta$. Thus, their beliefs (on the belief set level) with respect to α and β are the same.

Both Abe and Bob receive and accept the information that α is wrong, and they both revise their belief states to include the new belief that $\neg\alpha$. After that, Abe has the basic beliefs $\neg\alpha$ and β, whereas Bob has the basic beliefs $\neg\alpha$ and $\alpha \leftrightarrow \beta$. Now, their belief sets are no longer the same. Abe believes that β whereas Bob believes that $\neg\beta$.

(In the AGM approach, cases like these are taken care of by assuming that although Abe's and Bob's belief states are represented by the same belief set, this belief set is associated with different selection mechanisms in the two cases [7]. Abe has a selection mechanism that gives priority to β over $\alpha \leftrightarrow \beta$, whereas Bob's selection mechanism has the opposite priorities.)

There is only one logically closed inconsistent set, namely the whole language. Hence, there is only one inconsistent belief set. In other words, if $\boldsymbol{K}$ and $\boldsymbol{K}'$ are inconsistent belief sets, then $\boldsymbol{K} = \boldsymbol{K}'$. The corresponding property does not hold for belief bases. The inconsistent belief bases $\{p, \neg p, q\}$ and $\{p, \neg p, \neg q\}$ are statically equivalent, but they are not dynamically equivalent since, by any reasonable operator of contraction, $\{p, \neg p, q\} \div p = \{\neg p, q\}$ and $\{p, \neg p, \neg q\} \div p = \{\neg p, \neg q\}$.

This is a valuable property of belief bases as compared to belief sets. A database that contains inconsistent information does not have to be beyond repair. It should be possible to remove the inconsistency, while keeping the information that is not affected by contradiction. This can be done if the inconsistent belief state is represented by a belief base. On the other hand, once we have arrived at an inconsistent belief *set*, all distinctions are lost, and they cannot be so easily regained by operations performed directly on the belief set.

5. Operations on belief bases

Expansion on belief bases must be different from expansion on belief sets in order not to introduce logical closure.

Definition 8. Let A be a belief base and α a sentence. $A + \alpha$, the (*non-closing*) *expansion* of A by α, is defined as follows:

$$A + \alpha = A \cup \{\alpha\} .$$

In order to limit the number of symbols, the same symbol ("+") will be used for both closing and non-closing expansions.

Partial meet contraction, as outlined in Section 2, is equally applicable to belief bases. An important difference is that recovery ($A \subseteq \mathrm{Cn}((A \div \alpha) \cup \{\alpha\})$) does not hold for partial meet contraction on belief bases. Another interesting difference is that some transitively relational partial meet contractions on bases are not transitively maximizingly relational (TMR). For belief sets, the two categories of contraction coincide [13].

Partial meet contraction is based on a selection among maximal subsets of A that do not imply α. Another possible approach is to select the sentences to be discarded. They should be chosen among those elements of A that contribute to make it imply α. An operator that does exactly this, *kernel contraction*, was introduced recently [15].

Definition 9 [15]. Let A be a set of sentences and α a sentence. Then $A \perp\!\!\!\perp \alpha$ is the set such that $X \in A \perp\!\!\!\perp \alpha$ if and only if:

(1) $X \subseteq A$,
(2) $X \vdash \alpha$, and
(3) if $Y \subset X$, then $Y \nvdash \alpha$.

$A \perp\!\!\!\perp \alpha$ is a *kernel set*, and its elements are the *α-kernels* of A.

The function that selects sentences to be removed is called an *incision function* since it makes an incision into every α-kernel.

Definition 10 [15]. Let A be a set of sentences. An *incision function* for A is a function σ such that for all sentences α:

(1) $\sigma(A \perp\!\!\!\perp \alpha) \subseteq \bigcup(A \perp\!\!\!\perp \alpha)$,
(2) if $\emptyset \neq X \in A \perp\!\!\!\perp \alpha$, then $X \cap \sigma(A \perp\!\!\!\perp \alpha) \neq \emptyset$.

The outcome of a kernel contraction consists of all elements of the original set not selected for removal by the incision function:

Definition 11 [15]. Let A be a set of sentences and σ an incision function for A. The *kernel contraction* on A that is generated by σ is the operation $\approx_\sigma$ such that for all sentences α:

$$A \approx_\sigma \alpha = A \backslash \sigma(A \perp\!\!\!\perp \alpha) .$$

An operation $\div$ on A is a kernel contraction if and only if there is some incision function σ for A such for all sentences α: $A \div \alpha = A \approx_\sigma \alpha$.

All partial meet contractions are kernel contractions. The converse relationship does not hold, or in other words: *some kernel contractions are not partial meet contractions* [15]. Thus, as applied to a belief base, kernel contraction is a more general type of operation than partial meet contraction. It turns out, however, that when we apply the two types of operation directly to belief sets, then the distinction disappears (unless we accept contraction-outcomes that are not logically closed, which is clearly not reasonable in contraction of belief sets.)

Observation 12 [15]. Let $\boldsymbol{K}$ be a belief set, and let $\div$ be an operation on $\boldsymbol{K}$ that satisfies *closure*, i.e., such that for all α, $\boldsymbol{K} \div \alpha = \mathrm{Cn}(\boldsymbol{K} \div \alpha)$. Then the following two conditions are equivalent:
(1) $\div$ is a kernel contraction on $\boldsymbol{K}$,
(2) $\div$ is a partial meet contraction on $\boldsymbol{K}$.

Operators of *revision* on belief bases can be obtained in essentially the same way as for belief sets, i.e., through the Levi identity. The only difference is that here, of course, non-closing rather than closing expansion has to be applied.

Definition 13. Let γ be a selection function for the belief base A. Then the operator $\mp_\gamma$ of *internal partial meet revision* on A is defined as follows:

$$A \mp_\gamma \alpha = (A \sim_\gamma \neg\alpha) \cup \{\alpha\} .$$

The Levi identity is based on the following recipe for revision:
(1) Contract by $\neg\alpha$.
(2) Expand by α.
Alternatively, the two suboperations may take place in reverse order [16]:
(1) Expand by α.
(2) Contract by $\neg\alpha$.
In equational terms, this corresponds to the following identity:

$$A * \alpha = (A + \alpha) \div \neg\alpha \quad (\textit{reversed Levi identity } [16]) .$$

The composite operation $*$ obtained in this way is called *external revision*. The names "internal" and "external" indicate that in internal revision, the suboperation of contraction takes place inside the original belief base, whereas in external revision it takes place outside of the original set.

In typical cases, external revision has an *intermediate inconsistent state* in which both α and $\neg\alpha$ are believed, whereas internal revision has an *intermediate non-committed state* in which neither α nor $\neg\alpha$ is believed. The two types of operations have different logical properties [16].

(In external revision, when one and the same belief base (A) is revised by two

different sentences α and β, the suboperation of contraction will be applied to different belief bases, namely $A \cup \{\alpha\}$ and $A \cup \{\beta\}$, respectively. Therefore, external revision requires a global contraction operator that can be applied to any belief base. The global generalization of partial meet contraction is straightforward, but will not be performed here [12].)

It does not make much sense to apply external revision to belief sets. Since $\boldsymbol{K} + \alpha = \boldsymbol{K}' + \alpha$ for all belief sets $\boldsymbol{K}$ and $\boldsymbol{K}'$ that contain $\neg\alpha$, the information inherent in the previous belief set is completely lost in the first step, expansion, of belief-contravening external revision, and it cannot be regained in the second step.

6. Knowledge-level analysis

In an influential 1982 paper, Allen Newell postulated the existence in artificial intelligence of a *knowledge level* [23]. In the traditional hierarchy of system levels, beginning with the device level and the circuit level, the knowledge level is positioned immediately above the symbol level (program level). The knowledge level is specified entirely in terms of the contents of the knowledge (beliefs). There is no distinction on this level between information that is explicitly available and information that is implied by available information [4]. It should be possible to predict and understand what an agent does on the knowledge level, without referring to the symbol level, in much the same way as the symbol level should allow for prediction and understanding without reference to the lower levels of the system.

Belief sets are knowledge-level entities. More precisely put, changes on belief sets represent on the knowledge level "what an ideal reasoner would or should do when forced to reorganize his beliefs", thus indicating what people and computers should do "if they were not bounded by limited logical reasoning capabilities" [9].

We are thus faced with a dilemma. On the one hand, we wish to be able to describe and analyze epistemic behaviour on the knowledge level. This would lead us to use belief sets as models of belief states. On the other hand, we need models that do not force us to accept the recovery postulate, that have a finite representation and are suitable for actual computing. This leads us to use finite belief bases, rather than belief sets, to represent belief states.

This dilemma can be solved. It is possible to exactly characterize the operations on a belief set that can be generated by assigning to it a (finite) belief base and operators of contraction and revision that are applied to that belief base. In other words, we can perform *knowledge-level analysis of belief base operations*.

Let $\boldsymbol{B}$ be a belief base, and let $\boldsymbol{K}$ be the corresponding belief set, i.e., let $\boldsymbol{K} = \mathrm{Cn}(\boldsymbol{B})$. Furthermore, let $-$ be an operator of contraction on $\boldsymbol{B}$. Let $\div$ be an operator on $\boldsymbol{K}$ such that

$$\text{for all } \alpha, \quad \boldsymbol{K} \div \alpha = \mathrm{Cn}(\boldsymbol{B} - \alpha)\,.$$

Then $\div$ is an operator of contraction on $\boldsymbol{K}$. It will be called the *closure* of the operator $-$ on $\boldsymbol{B}$. More generally:

Definition 14. Let $\circ$ be an operator on a set $\boldsymbol{B}$. The *closure* of $\circ$ is the operator $\circ'$ on $\mathrm{Cn}(\boldsymbol{B})$ such that for all sentences α:

$$(\mathrm{Cn}(\boldsymbol{B}))\circ'\alpha = \mathrm{Cn}(\boldsymbol{B}\circ\alpha)\,.$$

In this way, every operator of belief base contraction generates an operator of belief set contraction.

For simplicity, closures of various types of operations will be referred to as "base-generated" versions of these operations. (The operations that are applied directly to the belief set will be referred to as the "direct" versions.) Thus, the closure of an operator of partial meet contraction will be called a "base-generated partial meet contraction".

In addition to the postulates referred to in Section 2, the following contraction postulates have turned out to be useful for the characterization of base-generated contraction operators:

(1) There is a finite set A such that for every sentence α, $\boldsymbol{K}\div\alpha = \mathrm{Cn}(A')$ for some $A'\subseteq A$ (*finitude*).
(2) If it holds for all δ that $\boldsymbol{K}\div\delta\vdash\alpha$ if and only if $\boldsymbol{K}\div\delta\vdash\beta$, then $\boldsymbol{K}\div\alpha = \boldsymbol{K}\div\beta$ (*symmetry*).
(3) If $\boldsymbol{K}\div\beta\not\subseteq\boldsymbol{K}\div\alpha$, then there is some δ such that $\boldsymbol{K}\div\delta\nvdash\alpha$ and $\boldsymbol{K}\div\delta\cup\boldsymbol{K}\div\beta\vdash\alpha$ (*weak conservativity*).
(4) If $\boldsymbol{K}\div\beta\not\subseteq\boldsymbol{K}\div\alpha$, then there is some δ such that $\boldsymbol{K}\div\alpha\subseteq\boldsymbol{K}\div\delta\nvdash\alpha$ and $\boldsymbol{K}\div\delta\cup\boldsymbol{K}\div\beta\vdash\alpha$ (*conservativity*).
(5) If $\vdash\alpha\rightarrow\beta$ and the set of β-removals that are also maximally preservative α-removals is non-empty, then it coincides with the set of maximally preservative β-removals (*regularity*). (An α-removal is a contraction-outcome $\boldsymbol{K}\div\beta$ such that $\boldsymbol{K}\div\beta\nvdash\alpha$. A maximally preservative α-removal is a contraction-outcome $\boldsymbol{K}\div\beta$ such that $\boldsymbol{K}\div\alpha\subseteq\boldsymbol{K}\div\beta\nvdash\alpha$ and that for all δ, if $\boldsymbol{K}\div\beta\subset\boldsymbol{K}\div\delta$ then $\boldsymbol{K}\div\delta\vdash\alpha$.)
(6) If $\alpha\in\boldsymbol{K}\div(\alpha\&\beta)$, then $\alpha\in\boldsymbol{K}\div(\alpha\&\beta\&\delta)$ (*conjunctive trisection*).

The following axiomatic characterizations have been obtained for base-generated contractions.

Theorem 15. *An operator $\div$ on a consistent belief set $\boldsymbol{K}$ is generated*

(1) *by an operator of kernel contraction on a finite base for $\boldsymbol{K}$ if and only if $\div$ satisfies closure, inclusion, vacuity, success, extensionality, finitude, symmetry, and weak conservativity* [15];
(2) *by an operator of partial meet contraction on a finite base for $\boldsymbol{K}$ if and only if $\div$ satisfies closure, inclusion, vacuity, success, extensionality, finitude, symmetry, and conservativity* [17];
(3) *by an operator of TMR partial meet contraction on a finite base for $\boldsymbol{K}$ if and only if $\div$ satisfies closure, inclusion, vacuity, success, extensionality, finitude, symmetry, conservativity, and regularity* [17];
(4) *by an operator of TMR partial meet contraction on a finite, disjunctively closed base for $\boldsymbol{K}$ if and only if $\div$ satisfies closure, inclusion, vacuity,*

success, extensionality, finitude, symmetry, conservativity, conjunctive overlap, conjunctive inclusion, and conjunctive trisection [14].

A belief base $\boldsymbol{B}$ is disjunctively closed if and only if it holds for all sentences α and β that if $\alpha \in \boldsymbol{B}$ and $\beta \in \boldsymbol{B}$, then $\alpha \vee \beta \in \boldsymbol{B}$.

7. Comments on the representation results

The representation results of Theorem 15 refer to several new postulates, and also to disjunctive closure of belief bases. In this section, the intuitive plausibility of these new formal properties will be discussed.

7.1. Finitude

We have finite minds, and other belief-carrying systems such as computers and animals are finite as well. A realistic representation of a belief state should reflect the finiteness of actual belief systems. This was pointed out in a 1988 paper by two members of the AGM trio, Peter Gärdenfors and David Makinson. They wrote:

> In all applications, the knowledge sets [belief sets] will be *finite* in the sense that the consequence relation $\vdash$ partitions the elements of $\boldsymbol{K}$ into a finite number of equivalence classes. [8]

The property referred to here can also be expressed as follows:

> *Strong finitude:*
> If A is an infinite set such that $\mathrm{Cn}(\{\alpha\}) \neq \mathrm{Cn}(\{\beta\})$ for all $\alpha, \beta \in A$, then $\mathrm{Cn}(A) \not\subseteq \boldsymbol{K}$.

The following example brings out the force of strong finitude:

Example. For every positive integer n, let π_n denote that the Roman Catholic Church has a present at least one and at most n popes. I believe in every sentence in the infinite sequence $\pi_1, \pi_2, \pi_3, \ldots$.

The "infiniteness" of this example should be admissible even for a finite mind. Surely a finite mind, such as that of a human being, should be capable of holding all elements of this infinite set of beliefs. Then, however, its belief set will violate strong finitude, since no two sentences in the sequence $\pi_1, \pi_2, \pi_3, \ldots$ are logically equivalent.

More generally, strong finitude precludes the use of a language with infinitely many non-equivalent sentences. The finiteness of actual belief systems seems to be weaker than what is expressed by this postulate.

There are at least two weaker but much more plausible finiteness properties. The first of these follows from the condition that belief sets have finite representations:

Finite representability:
For every sentence α, there is some finite set A such that $\boldsymbol{K} \div \alpha = \mathrm{Cn}(A)$.

Finite representability need not be violated in our example. π_1 implies all of π_2, $\pi_3, \ldots$. Therefore, for the infinite sequence $\pi_1, \pi_2, \pi_3, \ldots$ to be included in $\boldsymbol{K}$ (or $\boldsymbol{K} \div \alpha$) it is sufficient that π_1 is included in the finite representation (base) of $\boldsymbol{K}$ (respectively $\boldsymbol{K} \div \alpha$).

The second of the two finiteness properties states that although there may be infinitely many sentences by which the belief set can be contracted, there are only a finite number of belief sets that can be obtained through contraction.

Finite number of contractions:
$\{\boldsymbol{K}' \mid \boldsymbol{K}' = \boldsymbol{K} \div \alpha$ for some $\alpha\}$ is finite.

This postulate need not either be violated in our example. With reasonable background beliefs about the Roman Catholic Church, π_2, π_3, etc. are believed only as a consequence of belief in π_1. They all stand or fall with π_1, so that if one of them is lost, then the rest of them will be lost as well. Thus, e.g., $\boldsymbol{K} \div \pi_8 = \boldsymbol{K} \div \pi_9$.

These two finiteness properties can be combined into the following:

Finitude [17]:
There is a finite set A such that for every sentence α, $\boldsymbol{K} \div \alpha = \mathrm{Cn}(A')$ for some $A' \subseteq A$.

The following observation follows directly from the definitions:

Observation 16. A contraction operator $\div$ on $\boldsymbol{K}$ satisfies *finitude* if and only if it satisfies both *finite representability* and *finite number of contractions*.

Finitude is acceptable as a general requirement on rational belief change.

7.2. Symmetry

In actual belief states, some beliefs are so strongly connected that they stand or fall together:

Example. I believe that either Paris or Oslo is the capital of France (α). I also believe that either Paris or Stockholm is the capital of France (β). Both these beliefs are entirely based on my belief that Paris is the capital of France. Therefore, a contraction by some sentence δ removes α if and only if it removes β (namely if and only if it removes the common justification of these two beliefs). There is no contraction by which I can retract α without retracting β or vice versa.

This pattern can be generalized. If two beliefs stand or fall together, then this must be because their justifications are the same, so that you cannot remove the

justifications of one without removing those of the other. Contraction should be a process in which the belief state is changed so that the belief to be contracted is no longer justified. Therefore it can reasonably be expected that two beliefs that stand or fall together have the same contraction-outcome. This assumption is expressed in the symmetry postulate:

Symmetry [17]:
If it holds for all δ that $\boldsymbol{K} \div \delta \vdash \alpha$ if and only if $\boldsymbol{K} \div \delta \vdash \beta$, then $\boldsymbol{K} \div \alpha = \boldsymbol{K} \div \beta$.

7.3. Conservativity and weak conservativity

As we saw in Section 3, the recovery postulate seems to be too strong as a general requirement on belief contraction. To replace it we need some weaker postulate that prevents unmotivated deletions from the original belief set. The following is a first approximation:

(1) If $\varepsilon \in \boldsymbol{K}$ and $\varepsilon \notin \boldsymbol{K} \div \alpha$, then ε contributes to the fact that $\boldsymbol{K}$ implies α.

In order to translate (1) into formal language, we need to interpret what it means to "contribute to the fact that $\boldsymbol{K}$ implies α". The following seems to be a reasonable interpretation:

(2) If $\varepsilon \in \boldsymbol{K}$ and $\varepsilon \notin \boldsymbol{K} \div \alpha$, then there is a set A such that $A \subseteq \boldsymbol{K}$, $\alpha \notin \mathrm{Cn}(A)$ and $\alpha \in \mathrm{Cn}(A \cup \{\varepsilon\})$.

This postulate has been called "core-retainment". Unfortunately, it fails for our purposes, since any operation on a belief set that satisfies core-retainment also satisfies recovery [11]. The following example helps us to see why the application of core-retainment to belief sets goes wrong.

Example. Let α denote that John's wife is faithful and β that the Earth rotates around the Sun. When John's wife tells him that she is having a love affair, he contracts his belief set by α. Presumably, his belief in β will remain after this contraction.

All that core-retainment requires for $\beta \notin \boldsymbol{K} \div \alpha$ is that there is some $A \subseteq \boldsymbol{K}$ such that $\alpha \notin \mathrm{Cn}(A)$ and $\alpha \in \mathrm{Cn}(A \cup \{\beta\})$. This is clearly satisfied in our example; just let $A = \{\beta \rightarrow \alpha\}$.

However, $\{\beta \rightarrow \alpha\}$ is a highly artificial subset of $\boldsymbol{K}$ (and the same applies to its closure $\mathrm{Cn}(\{\beta \rightarrow \alpha\})$. It does not seem reasonable for John to give up both α and β but retain belief in $\beta \rightarrow \alpha$, that he only believed in from the beginning as a consequence of his belief in α. Although $\mathrm{Cn}(\{\beta \rightarrow \alpha\})$ is a subset of the original belief set, it is not itself a reasonable belief set. In order to exclude constructions of this type we should require of A in (2) that it be a reasonable belief set (from the perspective of $\boldsymbol{K}$). This leads us to the following condition:

(3) If $\varepsilon \in \boldsymbol{K}$ and $\varepsilon \notin \boldsymbol{K} \div \alpha$, then there is a reasonable belief set $\boldsymbol{K}'$ such that $\boldsymbol{K}' \subseteq \boldsymbol{K}$, $\alpha \notin \mathrm{Cn}(\boldsymbol{K}')$, and $\alpha \in \mathrm{Cn}(\boldsymbol{K}' \cup \{\varepsilon\})$.

The next step is to determine which of the subsets of $\boldsymbol{K}$ are reasonable belief sets. It does not seem possible to do this in terms of the truth-functional relationships of the elements of $\boldsymbol{K}$. The only other means that we have at our disposal is the operator of contraction. The following definition employs that operator in what seems to be the most intuitively reasonable way:

(A) A subset $\boldsymbol{K}'$ of $\boldsymbol{K}$ is a reasonable belief set from the viewpoint of $\boldsymbol{K}$ if and only if there is some finite series of contractions, starting with $\boldsymbol{K}$, that results in $\boldsymbol{K}'$, i.e.

$$\boldsymbol{K}' = \boldsymbol{K} \div \alpha_1 \div \alpha_2 \div \cdots \div \alpha_n .$$

Since iterated contractions are often difficult to handle in a formal context, it is worth investigating whether our purposes can be served by the following approximation of (A):

(B) A subset $\boldsymbol{K}'$ of $\boldsymbol{K}$ is a reasonable belief set from the viewpoint of $\boldsymbol{K}$ if and only if there is some sentence α such that $\boldsymbol{K}' = \boldsymbol{K} \div \alpha$.

If (B) is applied to (3), then we obtain the following:

(4) If $\varepsilon \in \boldsymbol{K}$ and $\varepsilon \notin \boldsymbol{K} \div \alpha$, then there is a sentence δ such that $\boldsymbol{K} \div \delta \subseteq \boldsymbol{K}$ and $\alpha \notin \mathrm{Cn}(\boldsymbol{K} \div \delta)$ and $\alpha \in \mathrm{Cn}(\boldsymbol{K} \div \delta \cup \{\varepsilon\})$.

Provided that the inclusion postulate is satisfied, (4) is equivalent to the following:

(5) If $\varepsilon \in \boldsymbol{K}$ and $\varepsilon \notin \boldsymbol{K} \div \alpha$, then there is a sentence δ such that $\boldsymbol{K} \div \delta \nvdash \alpha$ and $\boldsymbol{K} \div \delta \cup \{\varepsilon\} \vdash \alpha$.

However, we are not quite finished. In (1), and consequently in (2)–(5), it was assumed that *all* elements ε of $\boldsymbol{K}$ are worth retaining, in the sense that if $\varepsilon \notin \boldsymbol{K} \div \alpha$, then ε contributes to the fact that $\boldsymbol{K}$ implies α. This is not a realistic feature. We should restrict (5) to hold only for those sentences ε in $\boldsymbol{K}$ that represent self-sustained beliefs:

(6) If ε is a self-sustained belief in $\boldsymbol{K}$ and $\varepsilon \notin \boldsymbol{K} \div \alpha$, then there is a sentence δ such that $\boldsymbol{K} \div \delta \nvdash \alpha$ and $\boldsymbol{K} \div \delta \cup \{\varepsilon\} \vdash \alpha$.

Our next task is to explicate in formal terms what it means for a belief to be self-sustained, or capable of standing on its own. For a belief to be self-sustained in $\boldsymbol{K}$, it should be in principle possible to "peel off" all other beliefs in $\boldsymbol{K}$, and retain only this one belief. In other words:

(C) ε is a self-sustained belief in $\boldsymbol{K}$ if and only if there are sentences $\beta_1, \beta_2, \ldots, \beta_n$ such that $\mathrm{Cn}(\{\varepsilon\}) = \boldsymbol{K} \div \beta_1 \div \beta_2 \div \cdots \div \beta_n$.

(C) has the same disadvantages as (A). By an analogous argument, it can be simplified as follows:

(D) ε is a self-sustained belief in $\boldsymbol{K}$ if and only if there is a sentence β such that $\mathrm{Cn}(\{\varepsilon\}) = \boldsymbol{K} \div \beta$.

By applying (D) to (6) we obtain:

(7) If there are ε and β such that $\mathrm{Cn}(\{\varepsilon\}) = \boldsymbol{K} \div \beta$, and $\varepsilon \in \boldsymbol{K}$ and $\varepsilon \not\in \boldsymbol{K} \div \alpha$, then there is a sentence δ such that $\boldsymbol{K} \div \delta \not\vdash \alpha$ and $\boldsymbol{K} \div \delta \cup \{\varepsilon\} \vdash \alpha$.

Under the assumption that closure holds, $\varepsilon \not\in \boldsymbol{K} \div \alpha$ in (7) can be replaced by $\boldsymbol{K} \div \beta \not\subseteq \boldsymbol{K} \div \alpha$. Since $\boldsymbol{K}$ is logically closed, $\varepsilon \in \boldsymbol{K}$ can be replaced by $\boldsymbol{K} \div \beta \subseteq \boldsymbol{K}$. Under the assumption that inclusion holds, this is a redundant condition. We can therefore transform (7) to the following:

(8) If there are ε and β such that $\mathrm{Cn}(\{\varepsilon\}) = \boldsymbol{K} \div \beta$, and $\boldsymbol{K} \div \beta \not\subseteq \boldsymbol{K} \div \alpha$, then there is a sentence δ such that $\boldsymbol{K} \div \delta \not\vdash \alpha$ and $\boldsymbol{K} \div \delta \cup \boldsymbol{K} \div \beta \vdash \alpha$.

If finite representability holds, then (8) is equivalent to the following:

(9) If $\boldsymbol{K} \div \beta \not\subseteq \boldsymbol{K} \div \alpha$, then there is some δ such that $\boldsymbol{K} \div \delta \not\vdash \alpha$ and $\boldsymbol{K} \div \delta \cup \boldsymbol{K} \div \beta \vdash \alpha$ (*weak conservativity*).

Admittedly, the argument that led up to the postulate of weak conservativity was somewhat roundabout, as a result of compromises aiming at making the underlying intuitions accessible to formal treatment. Since this formulation has turned out to work, in the sense of corresponding to plausible operations of contraction (see Section 6), it can nevertheless be accepted, at least as a first approximation.

One more aspect needs attention: In (1), we required of an excluded sentence ε that it "contributes to the fact that $\boldsymbol{K}$ implies α". A stronger but not implausible requirement is that ε "contributes to the fact that $\boldsymbol{K}$ but not $\boldsymbol{K} \div \alpha$ implies α". The starting point for our derivation will then be:

(1′) If $\varepsilon \in \boldsymbol{K}$ and $\varepsilon \not\in \boldsymbol{K} \div \alpha$, then ε contributes to the fact that $\boldsymbol{K}$ but not $\boldsymbol{K} \div \alpha$ implies α.

The same chain of arguments can be applied to (1′) as to (1), leading us to the following postulate:

(9′) If $\boldsymbol{K} \div \beta \not\subseteq \boldsymbol{K} \div \alpha$, then there is some δ such that $\boldsymbol{K} \div \alpha \subseteq \boldsymbol{K} \div \delta \not\vdash \alpha$ and $\boldsymbol{K} \div \delta \cup \boldsymbol{K} \div \beta \vdash \alpha$ (*conservativity*).

7.4. Conjunctive trisection

Roughly speaking, there are three ways to ensure that a set does not imply $\alpha \& \beta \& \delta$, namely to see to it that it does not imply α, that it does not imply β, or that it does not imply δ. The first two of these are also the two ways to make it not imply $\alpha \& \beta$. If $\alpha \in \boldsymbol{K} \div (\alpha \& \beta)$, then it is better (more information-economical) to exclude β than to exclude α. Thus, there is at least one way to remove $\alpha \& \beta \& \delta$ that is better than to exclude α. We can therefore reasonably expect that $\alpha \in \boldsymbol{K} \div (\alpha \& \beta \& \delta)$. Hence, conjunctive trisection should hold.

7.5. Regularity

A contraction-outcome $K \div \beta$ is an α-removal if and only if $\alpha \notin K \div \beta$. An α-removal $K \div \beta$ will be called a *preservative α-removal* if and only if $K \div \alpha \subseteq K \div \beta$, and a *strictly preservative α-removal* if and only if $K \div \alpha \subset K \div \beta$. A strictly preservative α-removal is an operation that removes α, and does this in a more economical way than what is done by the contraction by α.

Often, a belief can be removed more economically if more specified information can be used. As an example, I believe that Albert Schweitzer was a German missionary (α). Let α_1 denote that he was a German and α_2 that he was a missionary, so that $\alpha \leftrightarrow \alpha_1 \& \alpha_2$. If I have to contract my belief set by α, then the contracted belief set will contain neither α_1 nor α_2. Admittedly it would be logically sufficient to withdraw one of them. However, they are both equally entrenched, so that I do not know which to choose in preference over the other. Therefore, both will have to go. On the other hand, if I have to contract my belief set by α_1, then I have no reason to let go of α_2. To contract by α_1 is, given the structure of my belief state, a more specified way to remove α. Thus we may expect that $K \div \alpha \subset K \div \alpha_1$, so that $K \div \alpha_1$ is a strictly preservative α-removal.

Let δ denote that Albert Schweitzer was a Swede, and let us consider the contraction of K by $\alpha_1 \vee \delta$, "Albert Schweitzer was a German or a Swede". Since I believe in $\alpha_1 \vee \delta$ only as a consequence of my belief in α_1, I can only retract $\alpha_1 \vee \delta$ by retracting α_1. Therefore, $K \div (\alpha_1 \vee \delta)$ is not a proper superset of $K \div \alpha_1$, i.e., it is not a more conservative α-withdrawal than $K \div \alpha_1$. Indeed, the way my beliefs are structured, α_1 cannot be further subdivided in the way that α was subdivided into α_1 and α_2. There is no part of α_1 that stands on its own and can be retracted from K without the rest of α_1 being lost as well. In this sense, no α-removal can be more conservative than $K \div \alpha_1$.

More generally, $K \div \beta$ is a *maximally preservative α-removal* if and only if it is a preservative α-removal and there is no α-removal $K \div \delta$ such that $K \div \beta \subset K \div \delta$. Intuitively, to perform a maximally preservative α-removal is to make the belief set not imply α, making use of information that is sufficiently specified to allow one to remove a part of α so small that no smaller part of it can be removed alone.

Maximally preservative removals are information-maximal contraction-outcomes. In an orderly and coherent belief state, one would expect their relative epistemic value (entrenchment) to be independent of what sentence we are contracting by.

As an illustration, let us extend the Albert Schweitzer example. Let α_1 denote that Schweitzer was a German, α_2 that he was a missionary and α_3 that he was a physician. Let us assume that $K \div \alpha_1$ is a maximally preservative $(\alpha_1 \& \alpha_2 \& \alpha_3)$-removal, i.e. a maximally economical way to remove $\alpha_1 \& \alpha_2 \& \alpha_3$ from the belief set. Since $K \div \alpha_1$ is also an $(\alpha_1 \& \alpha_2)$-removal, and since the $(\alpha_1 \& \alpha_2)$-removals are a subset of the $(\alpha_1 \& \alpha_2 \& \alpha_3)$-removals, it should also be maximally preservative among these. Furthermore, if $K \div \alpha_2$ is equally economical as $K \div \alpha_1$ in the context of removing $\alpha_1 \& \alpha_2$ (i.e., if it is also a maximally preservative $(\alpha_1 \& \alpha_2)$-

removal), then it should also be equally economical as $\boldsymbol{K} \div \alpha_1$ in the context of removing $\alpha_1 \& \alpha_2 \& \alpha_3$ (i.e., it should also be a maximally preservative $(\alpha_1 \& \alpha_2 \& \alpha_3)$-removal). In general:

> If $\vdash \alpha \rightarrow \beta$ and the set of β-removals that are also maximally preservative α-removals is non-empty, then it coincides with the set of maximally preservative β-removals (*regularity*).

It may be convenient to divide *regularity* into two parts:

> If $\vdash \alpha \rightarrow \beta$ and some maximally preservative α-removal is also a β-removal, then all maximally preservative β-removals are maximally preservative α-removals (*regularity* 1).

> If $\vdash \alpha \rightarrow \beta$ and $\boldsymbol{K} \div \delta$ is both a β-removal and a maximally preservative α-removal, then it is a maximally preservative β-removal (*regularity* 2).

Regularity 1 is closely related to Amartya Sen's β property for rational choice behaviour, and *regularity* 2 to his α property [25]. Clearly, *regularity* holds if and only if both *regularity* 1 and *regularity* 2 hold.

7.6. Disjunctive closure

The ultimate criterion for a sentence to be an element of the belief base is that it is "self-sustained", i.e., worth retaining for its own sake (even if it is not implied by some other sentence that is worth retaining). Belief bases have often been taken to consist of the beliefs that have an independent justification. This is only a very rough approximation. Independently justified beliefs are worth retaining for their own sake, but they are not necessarily the only beliefs that are worth retaining for their own sake. This will be clear from the following example:

Example [10]. I originally believed, for good and independent reasons, both that Andy is the son of the mayor (α) and that Bob is the son of the mayor (β). Then I hear the mayor say in a public speech: "I certainly have nothing against our youth studying abroad. My only son did it for three years".

Upon hearing this, I contract my belief state by $\alpha \& \beta$. As a result of this I lose both my belief in α and my belief in β. However, I retain my belief that $\alpha \vee \beta$, i.e., that either Andy or Bob is the son of the mayor.

In this case, we may assume that $\alpha \vee \beta$ had no independent justification. It was believed only as a consequence of my beliefs in α and β. It seems reasonable, however, in this and many other cases, to retain belief in the disjunction of two independently justified beliefs, when they can no longer coexist and one cannot choose between them.

If we wish to make this into a general pattern of belief change, then we should postulate that if both α and β are worth retaining for their own sake, then so is $\alpha \vee \beta$. This is another way of saying that if both α and β satisfy the requirements

for being elements of the base, then so does $\alpha \vee \beta$. Such a belief base will be *closed under disjunction*.

Disjunctively closed belief bases do not share the immediate appeal of bases *simpliciter*. This should be seen as an interesting special case rather than as a mandatory property of plausible belief bases.

8. Base-generated revision

Operators of revision on a belief set can be base-generated in the same sense as operators of contraction. Base-generated internal partial meet revision and direct (internal) partial meet revision turn out to coincide in the following sense:

Observation 17 [14]. The following two statements about an operation $*$ on a belief set $\boldsymbol{K}$ are equivalent:

(1) There is some selection function γ for $\boldsymbol{K}$ such that for all α, $\boldsymbol{K} * \alpha = \boldsymbol{K} \mp_\gamma \alpha$.

(2) There is some base $\boldsymbol{B}$ for $\boldsymbol{K}$ and some selection function γ for $\boldsymbol{B}$ such that for all α, $\boldsymbol{K} * \alpha = \mathrm{Cn}(\boldsymbol{B} \mp_\gamma \alpha)$.

The concurrence of base-generated and direct revision comes to an end if further requirements are imposed on the revision operators. In particular, this will happen if the operators are required to be transitively maximizingly relational (TMR). The following two statements about an operation $*$ on a logically closed set $\boldsymbol{K}$ are *not* equivalent [14]:

(3) There is some TMR selection function γ for $\boldsymbol{K}$ such that for all α, $\boldsymbol{K} * \alpha = \boldsymbol{K} \mp_\gamma \alpha$.

(4) There is some base $\boldsymbol{B}$ for $\boldsymbol{K}$ and some TMR selection function γ for $\boldsymbol{B}$ such that for all α, $\boldsymbol{K} * \alpha = \mathrm{Cn}(\boldsymbol{B} \mp_\gamma \alpha)$.

However, if the belief base is also required to be disjunctively closed, then an equivalence similar to that of Observation 17 is obtained.

Observation 18 [14]. The following two statements about an operation $*$ on a belief set $\boldsymbol{K}$ are equivalent:

(3) There is some TMR selection function γ for $\boldsymbol{K}$ such that for all α, $\boldsymbol{K} * \alpha = \boldsymbol{K} \mp_\gamma \alpha$.

(5) There is some disjunctively closed base $\boldsymbol{B}$ for $\boldsymbol{K}$ and some TMR selection function γ for $\boldsymbol{B}$ such that for all α, $\boldsymbol{K} * \alpha = \mathrm{Cn}(\boldsymbol{B} \mp_\gamma \alpha)$.

9. Conclusion

By shifting the operations of contraction and revision from the belief set itself to some finite base for the belief set, we have obtained more plausible models of belief change; in particular the controversial recovery property was avoided in this way. Furthermore, since these models are finite, they are more realistic from a

computational point of view. These advantages have been obtained while still having exact characterizations on the knowledge level of the type that is one of the hallmarks of the AGM model.

Idealized models of updating, such as those introduced here, provide background values against which actual updating systems can be tested. Comparisons between theoretical and actual updating systems should largely be performed on the knowledge level, rather than on the symbol level or lower levels of the system. In order to make such knowledge-level comparisons possible, we need (1) knowledge-level characterizations of idealized models of updating, and (2) knowledge-level characterizations of actual updating systems. The results reported in this paper are a contribution to the first of these two preconditions for a convergence of theoretical and practical studies of database updating.

References

[1] C.E. Alchourrón, P. Gärdenfors and D. Makinson, On the logic of theory change: partial meet contraction and revision functions, *J. Symbolic Logic* **50** (1985) 510–530.

[2] C.E. Alchourrón and D. Makinson, Hierarchies of regulation and their logic, in: Hilpinen, ed., *New Studies in Deontic Logic* (Reidel, Dordrecht, Netherlands, 1981) 125–148.

[3] C.E. Alchourrón and D. Makinson, On the logic of theory change: contraction functions and their associated revision functions, *Theoria* **48** (1982) 14–37.

[4] R.J. Brachman and H.J. Levesque, What makes a knowledge base knowledgeable? A view of databases from the knowledge level, in: Kerschberg, ed., *Expert Database Systems, Proceedings from the First International Workshop* (Benjamin/Cummings, 1986) 69–78.

[5] A. Fuhrmann, Theory contraction through base contraction, *J. Philos. Logic* **20** (1991) 175–203.

[6] P. Gärdenfors, An epistemic approach to conditionals, *Am. Philos. Q.* **18** (1981) 203–211.

[7] P. Gärdenfors, The dynamics of belief systems: foundations vs. coherence theories, *Revue Int. Philos.* **44** (1990) 24–46.

[8] P. Gärdenfors and D. Makinson, Revisions of knowledge systems using epistemic entrenchment, in: *Proceedings Second Conference on Theoretical Aspects of Reasoning about Knowledge* (1988) 83–95.

[9] P. Gärdenfors and H. Rott, Belief revision, in: *Handbook of Logic in AI and Logic Programming* (to appear).

[10] S.O. Hansson, New operators for theory change, *Theoria* **55** (1989) 114–132.

[11] S.O. Hansson, Belief contraction without recovery, *Studia Logica* **50** (1991) 251–260.

[12] S.O. Hansson, A dyadic representation of belief, in: P. Gärdenfors, ed., *Belief Revision* (Cambridge University Press, Cambridge, 1992) 89–121.

[13] S.O. Hansson, Similarity semantics and minimal changes of belief, *Erkenntnis* **37** (1992) 401–429.

[14] S.O. Hansson, Changes on disjunctively closed bases, *J. Logic Language Inform.* **2** (1993) 255–284.

[15] S.O. Hansson, Kernel contraction, *J. Symbolic Logic* **59** (1994) 845–859.

[16] S.O. Hansson, Reversing the Levi identity, *J. Philos. Logic* **22** (1993) 637–669.

[17] S.O. Hansson, Theory contraction and base contraction unified, *J. Symbolic Logic* **58** (1993) 602–625.

[18] S. Lindström and W. Rabinowicz, Epistemic entrenchment with incomparabilities and relational belief revision, in: A. Fuhrmann, ed., *The Logic of Theory Change* (Springer-Verlag, Berlin, 1991) 93–126.

[19] I. Levi, Subjunctives, dispositions and chances, *Synthese* **34** (1977) 423–455.

[20] I. Levi, *The Fixation of Belief and Its Undoing* (Cambridge University Press, Cambridge, 1991).

[21] D. Makinson, On the status of the postulate of recovery in the logic of theory change, *J. Philos. Logic* **16** (1987) 383–394.
[22] J.P. Martins and S.C. Shapiro, A model for belief revision, *Artif. Intell.* **35** (1988) 25–79.
[23] A. Newell, The knowledge level, *Artif. Intell.* **18** (1982) 87–127.
[24] R. Niederée, Multiple contraction. A further case against Gärdenfors's principle of recovery, in: A. Fuhrmann, ed., *The Logic of Theory Change* (Springer-Verlag, Berlin, 1991) 322–334.
[25] A. Sen, *Collective Choice and Social Welfare* (Holden-Day, San Francisco, CA, 1970).

ELSEVIER

Artificial Intelligence 82 (1996) 237–257

Artificial Intelligence

Searching game trees under a partial order ★

Pallab Dasgupta, P.P. Chakrabarti *, S.C. DeSarkar

Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India 721302

Received January 1994; revised November 1994

Abstract

The problem of partial order game tree search arises from game playing situations where multiple, conflicting and non-commensurate criteria dictate the merit of a position of the game. In partial order game trees, the outcomes evaluated at the tip nodes are vectors, where each dimension of the vector represents a distinct criterion of merit. This leads to an interesting variant of the game tree searching problem where corresponding to every game playing strategy of a player, several outcomes are possible depending on the individual priorities of the opponent. In this paper, we identify the necessary and sufficient conditions for a *set* of outcomes to be inferior to another set of outcomes for *every* strategy. Using an algebra called *Dominance Algebra* on sets of outcomes, we describe a bottom-up approach to find the non-inferior sets of outcomes at the root node. We also identify shallow and deep pruning conditions for partial order game trees and present a partial order search algorithm on lines similar to the α-β pruning algorithm for conventional game trees.

1. Introduction

Current game tree searching methods assume that a given position of the game can be evaluated as a single numerical value which indicates the merit of that position. In normal two-player games, a MIN-MAX value [4] is defined, which indicates the best alternative available to a player. Depth-first algorithms like α-β *pruning* [1] and best-first algorithms like *SSS** [6] are known to efficiently determine this MIN-MAX value. These studies have also been extended to multiplayer games [2].

In this paper, we study a variant of the game tree search problem, where the costs evaluated at the tip nodes of the game tree are vectors. The objective of such a framework

★ Originally submitted as Research Note.

* Corresponding author. This author acknowledges the support of the Indian National Science Academy for partial support of this work.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0004-3702(94)00085-9

is to model game playing situations where the merit of a position of the game is dictated by multiple non-commensurate criteria. In such game playing situations, a position of the game can be better than another in some criteria of merit and worse in some of the other criteria. The individual priorities of each player decides which position would be most desirable for it.

It is well known that the presence of an adversary can lead to game playing situations in optimization problems. Such an adversary may appear as an actual competitor (as in competitive markets) or as random factors that emerge during problem solving and affect the final outcome. For example, the problem of searching for a maximum capacity path in a flow tree has been modeled as a game tree search problem [4] when the edge capacities are random. Papadimitriou and Yannakakis [3] have described an extension of the Traveling Salesperson Problem (TSP) called Canadian TSP that can be modeled as a game playing problem.

Optimization problems involving multiple, conflicting and non-commensurate objectives have motivated a new search model introduced by Stewart and White [5]. In this model, the cost structure is vector-valued, where each dimension of the cost vector represents a distinct criterion to be optimized. A partial order called *dominance* is used to eliminate clearly inferior solutions and obtain the set of non-inferior solutions in the search space. In this paper, we extend the same framework for modeling game playing situations involving multiple non-commensurate criteria.

Multiobjective game playing situations may be modeled by game trees where the tip nodes have vector-valued costs. Each dimension of the cost vector represents a distinct criterion of merit. The partial order game tree search problem is to find the non-inferior options of a player by using the following partial order.

Definition 1.1 (*Dominance*). Let $y^1 \equiv \langle y_1^1, y_2^1, \ldots, y_K^1 \rangle$ and $y^2 \equiv \langle y_1^2, y_2^2, \ldots, y_K^2 \rangle$ be two K-dimensional vectors. Then y^1 dominates y^2 iff:

$$y_i^1 \geqslant y_i^2 \qquad \forall i \quad 1 \leqslant i \leqslant K \quad \text{and} \quad y^1 \neq y^2.$$

A vector $y \in Y$ is said to be "non-dominated" in Y if there does not exist another vector $y' \in Y$ such that y' dominates y.

The definition of dominance is similar to that used by Stewart and White [5] to define the partial order for searching multiobjective OR-graphs.

In conventional game trees, there exists a total order on the values (outcomes) evaluated at the tip nodes. Given a set of outcomes at a MAX-node, player-1 chooses the maximum one. Likewise, the minimum outcome is chosen by player-2 at MIN-nodes. In partial order game trees, we can eliminate the clearly inferior outcomes using the *dominance* relation, but the choice of the desired outcome from the remaining non-dominated set of outcomes will depend on the individual preferences of the player. We call the preferences of a player combined with the resulting selection procedure the *strategy* of the player.

It is easy to see that if the individual strategies of both players are known, then by applying the strategy of player-1 at MAX-nodes and the strategy of its opponent (player-2) at the MIN-nodes, we can solve the problem using conventional game tree

search strategies. If the individual strategies of the players are not known, then for each strategy of player-1, several outcomes are possible depending on the strategy adopted by the opponent. Corresponding to each move from a node, a set of outcomes is possible for every strategy of the player. The objective of the partial order game tree search problem is to find the *non-inferior* sets of outcomes through each move from the root node of the game tree. Player-1 can then apply its own strategy on these sets of outcomes and select the move that returns the best set of outcomes based on its strategy. We shall show that even if the strategy of either player is known, it cannot be used in the interior nodes of the game tree, and hence it is necessary to find every non-inferior set of outcomes at the root node.

In this paper, we analyze the partial order game tree search problem and identify the necessary and sufficient conditions for a set of outcomes to be inferior to another set of outcomes for *every* strategy. For convenience of representation, we use an algebra called *Dominance Algebra* on the sets of outcomes to describe the non-inferior sets of outcomes and a bottom-up approach to determine the non-inferior sets of outcomes at the nodes of the game tree. Finally, we identify shallow and deep pruning conditions for partial order game trees and on the basis of those pruning conditions, we present a partial order search algorithm on lines similar to the α-β pruning algorithm for conventional game trees.

The paper is organized as follows. In Section 2 we describe the partial order game tree search problem. The use of *Dominance Algebra* is described in Section 3. In Section 4 we describe a brute force method to determine the non-inferior sets of outcomes for a player. The shallow and deep pruning conditions, as well as the search algorithm which uses the pruning conditions are described in Section 5.

2. The partial order game tree search problem

In the conventional game tree search problem, the values at the tip nodes of the game tree are members of a totally ordered set. Given the MIN-MAX values [4] of the children of a node, it is possible to determine the MIN-MAX value of the parent simply by using the total order to decide which value is the best. Using a bottom-up approach it is possible to determine the MIN-MAX value corresponding to each move at the root node of the game tree.

In the partial order game tree search problem, we only have a partial order on the the vector-valued outcomes. In the cases where the partial order can determine the better outcome, the choice is obvious. For example, in Fig. 1(a) it is obvious that player-1 at the MAX-node P will select the move with outcome $(11,5)$ since $(11,5)$ dominates both $(9,4)$ and $(7,3)$. Likewise, in Fig. 1(b) player-2 will select the move with outcome $(7,3)$ from node Q. In such situations, it is possible to decide the best move without any knowledge of the individual preferences of the players. On the other hand, consider the situation in Fig. 2(a). At node Q, player-2 has a choice between the outcomes $(11,5)$ and $(5,7)$. Since none dominates the other, the decision will be based on the preferences of player-2.

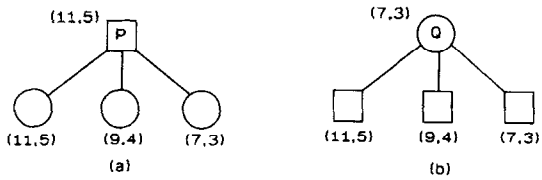


Fig. 1. Cases illustrating simple dominance.

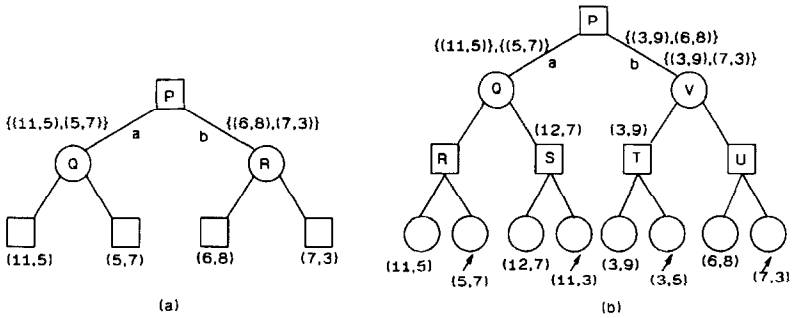


Fig. 2. Examples showing multiple sets of outcomes.

At node Q of Fig. 2(a), if the preferences of player-2 are not known, then it is not possible for player-1 to decide which outcome will be selected. Therefore, corresponding to move a at node P, we have a set of possible outcomes $S_a = \{(11,5),(5,7)\}$. In a similar manner, corresponding to move b at node P, we have a set of outcomes $S_b = \{(6,8),(7,3)\}$. Thus, at node P, player-1 has to choose between the *sets of outcomes* S_a and S_b, and accordingly take either move a or move b. This choice will depend on the preferences of player-1. The question which arises at this point is: *how does a player use its preferences to choose between such sets of outcomes?*

Thus the problem of the player is to choose between two sets of outcomes (such as S_a and S_b) with the knowledge that the preferences of the opponent (which are not known to this player) will decide the final outcome from the selected set. Let $\vec{x}$ denote the worst outcome in a given set based on the preferences of the player. Then the best that the given set can *guarantee* for the player is $\vec{x}$. This is similar to the familiar notion of MIN in conventional game tree search. Therefore, to compare two sets of outcomes based on individual preferences, we compare the worst outcome from each set based on those preferences. In case of a tie, we compare the other outcomes. The complete procedure is as follows:

Compare(S_1, S_2, ϕ)
To compare sets of outcomes S_1 and S_2 on the basis of preferences ϕ
1. If only S_1 is empty, then declare S_2 as better.
   Likewise, if only S_2 is empty, then declare S_1 as better.
   If both S_1 and S_2 are empty then
   select S_1 or S_2 randomly and declare it to be better.

2. Let $\vec{x}_1$ be the worst outcome in S_1 and
   $\vec{x}_2$ be the worst outcome in S_2 based on ϕ.
3. If $\vec{x}_1$ and $\vec{x}_2$ are of equal preference then
   3.1 Drop all outcomes from S_1 and S_2 that are of equal preference to $\vec{x}_1$
   3.2 Goto [Step 1]
4. If $\vec{x}_1$ is better than $\vec{x}_2$ based on ϕ, then declare S_1 as better else declare S_2 as better.

As an example, if the preferences of player-1 are such that $(5,7)$ is preferred over $(11,5)$ and $(11,5)$ is preferred over $(13,3)$, then the set $\{(13,3),(5,7)\}$ is preferred over the set $\{(13,3),(11,5)\}$. Also the set $\{(13,3),(5,7)\}$ is preferred over the set $\{(13,3)\}$, since depending on the preferences of the opponent there is a possibility of reaching the better outcome $(5,7)$ from the former set.

It should also be noted that two outcomes may have equal preference. Thus, the preferences of a player do not exactly induce a total order on the set of vector-valued outcomes; rather, they induce a many-to-one mapping from the set of outcomes to a totally ordered set which preserves the partial order imposed by the dominance relation.

It is easy to see that the way in which a player will behave can be described by its individual preferences and a procedure for comparing sets of outcomes. Throughout this paper we assume that the players use the procedure *Compare* to compare the sets of outcomes. We define the strategy of a player as follows.

Definition 2.1 (*Strategy*). The strategy of a player is a selection mechanism based on the procedure *Compare* and the individual preferences of the player which is consistent with the partial order imposed by the dominance relation. Thus if $\vec{x}$ is an outcome, which dominates an outcome $\vec{y}$, then for all strategies of player-1, $\vec{x}$ is better than $\vec{y}$ and for all strategies of player-2, $\vec{y}$ is better than $\vec{x}$.

Initially we analyze the situation that arises when the strategy of neither player is known. Later we shall show that even if the strategy of either of the players is known, the search problem remains the same.

In Fig. 2(a), corresponding to each move at node P we had only one set of outcomes. The following example illustrates that corresponding to a move, it is possible to have multiple sets of outcomes.

Example 2.2. Consider the game tree in Fig. 2(b). At node T player-1 will obviously select the outcome $\{(3,9)\}$. At node U player-1 can choose either $(6,8)$ or $(7,3)$ depending on its strategy. This strategy is not known to player-2; therefore, at node V, it has to choose between the sets $\{(6,8),(7,3)\}$ and $\{(3,9)\}$. Player-2 may choose either set depending on its own strategy.

Now suppose the strategy of player-1 is such that it prefers $(6,8)$ over $(7,3)$. If the game reaches node U, then it will select $(6,8)$. Therefore, corresponding to this strategy of player-1, move b (at node P) presents the set of outcomes $\{(3,9),(6,8)\}$ such that the opponent's strategy decides whether $(3,9)$ or $(6,8)$ will be reached. On the other hand, if player-1 prefers $(7,3)$ over $(6,8)$, then corresponding to that strategy, move

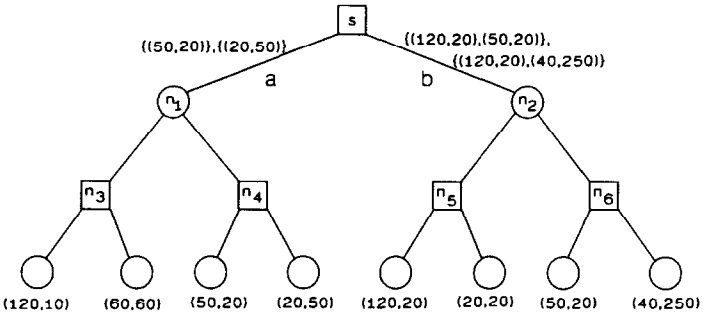


Fig. 3. Game tree illustrating that a strategy should not be used at interior nodes.

b presents the set of outcomes $\{(3,9),(7,3)\}$. Thus, if move b is selected at node P, then the outcome will either belong to the set $\{(3,9),(6,8)\}$ or the set $\{(3,9),(7,3)\}$ depending on the strategy adopted by player-1.

In a similar situation at node Q, player-2 has to choose between the sets $\{(12,7)\}$ and $\{(11,5),(5,7)\}$. In this case, since both $(11,5)$ and $(5,7)$ are dominated by $(12,7)$, player-2 will always select the move to node R. Therefore, corresponding to move a, we have two singleton sets, namely $\{(11,5)\}$ and $\{(5,7)\}$.

At node P of Fig. 2(b), we can provide player-1 with four sets of outcomes to choose from, namely $\{(11,5)\}$, $\{(5,7)\}$, $\{(3,9),(6,8)\}$ and $\{(3,9),(7,3)\}$. Are all these sets candidates for selection? Let us compare the sets $\{(11,5)\}$ and $\{(3,9),(7,3)\}$. Since $(11,5)$ dominates $(7,3)$, it is easy to see (through the procedure *Compare*) that there can be no strategy for player-1 that prefers the set $\{(3,9),(7,3)\}$. This set is therefore an inferior set of outcomes.

Example 2.2 shows that even if the individual strategies of the players are not known, certain sets of outcomes can be clearly discarded. The objective of partial order game tree search is to discard such sets of outcomes and provide the player at the root node with the non-inferior sets of outcomes.

Before giving the formal definition of the problem, it is necessary to make one further observation. Suppose that each player knows its own strategy but not that of its opponent. In the game tree representation, this means that only the strategy of player-1 is known. Can we use the strategy at the interior MAX-nodes of the game tree to prune away several sets of outcomes? The following example shows that the answer is negative in general.

Example 2.3. Consider the game tree in Fig. 3. Let the strategy of player-1 be such that given any two outcomes, the outcome which is greater in the first dimension is preferred. If two outcomes are equal in the first dimension, then the outcome with the larger second dimension is preferred. Let us first analyze the problem using the strategy of player-1 at the interior nodes. At node n_3 player-1 will choose the outcome $(120,10)$, and at node n_4 it will select $(50,20)$ based on its strategy. At node n_1, it is not known whether player-2 will select $(120,10)$ or $(50,20)$, and so corresponding to move a

at node P, we have the set of outcomes $\{(120,10),(50,20)\}$. Likewise, player-1 will select $(120,20)$ at node n_5 and $(50,20)$ at node n_6. At node n_2, player-2 will definitely select $(50,20)$ since it is dominated by $(120,20)$. Thus corresponding to move b we have the set $\{(50,20)\}$. If we compare the sets corresponding to move a and b on the basis of the strategy of player-1, we find that the set $\{(120,10),(50,20)\}$ is preferred over the set $\{(50,20)\}$, and so, move a appears to be better.

In this analysis, we have overlooked one vital point, that is, the opponent does not know the strategy of player-1. Therefore, from the point of view of player-2, player-1 can select either of the outcomes at nodes n_3 and n_4. Therefore, at node n_1, player-2 has to choose between the sets $\{(120,10),(60,60)\}$ and $\{(50,20),(20,50)\}$. Since both $(50,20)$ and $(20,50)$ are better than $(60,60)$ for every strategy of player-2, therefore player-2 will always select the set $\{(50,20),(20,50)\}$ and take the move to node n_4. Thus, corresponding to move a, we have two sets of outcomes, namely $\{(50,20)\}$ and $\{(20,50)\}$. For the given strategy of player-1, the set $\{(50,20)\}$ is preferred.

Using a similar reasoning, at node n_2, player-2 will have to select between the sets $\{(120,20)\}$ and $\{(50,20),(40,250)\}$. Player-2 can select either set depending on its strategy, therefore corresponding to move b, we have two sets of outcomes, namely $\{(120,20),(50,20)\}$ and $\{(120,20),(40,250)\}$. For the given strategy of player-1, the set $\{(120,20),(50,20)\}$ is preferred. By comparing this set with the set $\{(50,20)\}$ (which was preferred through move a) we find that actually move b is better for player-1.

The above example shows that unless the strategies of both players are known, we cannot apply the individual strategy of either player at an internal node while determining the sets of outcomes. Therefore, we can eliminate only those sets of outcomes that are inferior with respect to every strategy. In other words, we have to find the non-inferior *packets* of outcomes, where a *packet* of outcomes is defined as follows.

Definition 2.4 (*Packet of outcomes*). At a node n a set of outcomes P corresponding to a move a will be called a *packet* for a player iff it has the following two properties:

(1) There exists a strategy of the player, such that after taking move a, irrespective of the strategy adopted by the opponent, the final outcome will be better than or equal to an outcome in that set in every dimension.

(2) For every outcome $\vec{x}$ in P, there exists an opponent strategy such that if move a is taken, then the final outcome is $\vec{x}$.

From the previous discussion it follows that there may be several packets corresponding to a single move. The set of packets at a node is the union of the set of packets corresponding to every move at that node.

The second property of a *packet* ensures that redundant outcomes (that is, outcomes which will never be reached) are not included in a *packet*. For example in Fig. 3, the set of outcomes $\{(120,10),(50,20)\}$ corresponding to move a at node s satisfies the first property, but is not a packet for player-1 since the opponent at node n_1 will always select the move to n_4 (see Example 2.3), and therefore there is no opponent strategy to reach the outcome $(120,10)$.

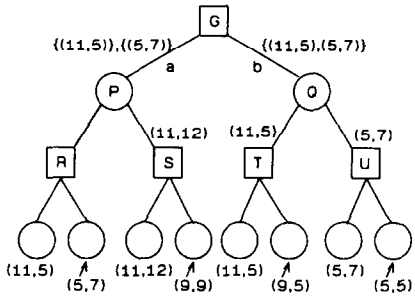


Fig. 4. Dominance by freedom of choice.

If we can find the entire set of packets at the root node of the game tree, then what we have is the sets of outcomes corresponding to every strategy of the player at the root node. Out of the entire set of packets, some packets may be inferior to other packets for every strategy of the player. What are the necessary and sufficient conditions for a packet to be inferior among the set of packets at a node? We identify two conditions as follows.

Clear Dominance. A packet P is inferior to a packet P' for a player if:

(1) each outcome in P' is either equal to some outcome in P or is strictly better than some outcome in P, and

(2) there exists at least one outcome in P' which is strictly better than some outcome in P.

In such cases we say that P' dominates P by *clear dominance.*

Dominance by Freedom of Choice. A packet P is inferior to a set of packets $P_1, \ldots, P_J$ if each packet P_i contains fewer outcomes than P, and the union of them yields P. In such cases we say that $P_1, \ldots, P_J$ dominates P by *freedom of choice.*

Before proving that these are the necessary and sufficient conditions for a packet to be inferior, we illustrate the idea of *dominance by freedom of choice* through an example. Note that the idea of *clear dominance* has already been illustrated in Example 2.2 and Example 2.3.

Example 2.5. Consider the game tree in Fig. 4. At node P, player-2 will always select the set $\{(11,5),(5,7)\}$ from the two sets $\{(11,5),(5,7)\}$ and $\{(11,12)\}$. Thus corresponding to move a at node G, we have two sets of outcomes, namely $\{(11,5)\}$ and $\{(5,7)\}$. At the nodes T and U, player-1 will always select $(11,5)$ and $(5,7)$ respectively. Thus, at node Q, player-2 can either select $(11,5)$ or $(5,7)$ depending on its strategy. Therefore, corresponding to move b we have the set of outcomes $\{(11,5),(5,7)\}$.

If the strategy of player-1 is such that $(11,5)$ is preferred over $(5,7)$, then the set $\{(11,5)\}$ will be preferred over $\{(11,5),(5,7)\}$ based on the procedure *Compare.* On

the other hand, if the strategy of player-1 is such that $(5,7)$ is preferred over $(11,5)$, then $\{(5,7)\}$ will be preferred over $\{(11,5),(5,7)\}$. Thus, whatever be the strategy of player-1, the set $\{(11,5),(5,7)\}$ will never be selected. The sets $\{(11,5)\}$ and $\{(5,7)\}$ together provide more freedom of choice than the union $\{(11,5),(5,7)\}$ and therefore dominate the set $\{(11,5),(5,7)\}$.

Theorem 2.6. *The conditions of* clear dominance *and* dominance by freedom of choice *are sufficient conditions for a packet to be inferior among a set of packets.*

Proof. Suppose a packet P' dominates a packet P by *clear dominance*. Consider a strategy ST of the player. Let $\vec{x}$ be the worst outcome in P' based on ST. From the definition of clear dominance, there exists some outcome $\vec{y}$ in P which is either equal to $\vec{x}$ or worse for every strategy. If $\vec{y}$ is worse than $\vec{x}$ then P is obviously inferior. If $\vec{x}$ is equal to $\vec{y}$, we drop them from P and P', and use the same reasoning on the next worst outcome. Since there exists at least one outcome in P' which is strictly better than an outcome in P, it follows that P is inferior to P' for every strategy ST.

Suppose the set of packets $P_1,\ldots,P_J$ dominates a packet P by *freedom of choice*. Consider a strategy ST of the player. Let $\vec{x}$ be the best outcome in P on the basis of ST. From the definition of dominance by freedom of choice, there exists a packet P_i which is a subset of P and contains $\vec{x}$. Let us compare P_i and P on the basis of ST. If the worst outcome in P does not belong to P_i, then P is inferior. Otherwise, we drop that outcome from both sets and consider the next worst outcome and so on. Since P_i contains the best outcome in P and contains fewer outcomes than P, we find that P is inferior to P_i for the strategy ST. It follows that P is inferior to at least one of the of packets $P_1,\ldots,P_J$ for every strategy. □

Theorem 2.7. *For a packet to be inferior among a set of packets S, either of the conditions* clear dominance *or* dominance by freedom of choice *is necessary.*

Proof. Let us consider a packet P which is neither dominated by *clear dominance*, nor by *freedom of choice*. Let P_u denote the union of all packets in S which are subsets of P. Let P' denote the set of outcomes in P that are not in P_u. Since P is not dominated by freedom of choice, P' is non-empty. We now consider a strategy as follows:

(1) Every outcome in P_u has equal priority. Every outcome in P' has equal priority. Outcomes from P' are preferred over outcomes from P_u.
(2) Every outcome that does not dominate an outcome in P, or is not equal to an outcome in P, has a lower priority than every outcome in P.

Since P' is non-empty and its outcomes are preferred over those in P_u, it follows that P is preferred over all packets which are subsets of P. P is not dominated by clear dominance; therefore every packet that is not a subset of P must contain an outcome which neither dominates nor is equal to an outcome in P. Therefore, P is preferred over such packets as well. It follows that P is the most preferred packet based on the above strategy, and therefore not an inferior packet. □

The game tree search problem studied in this paper may now be defined as follows.

The Partial Order Game Tree Search Problem.

- *Given*: A game tree where the values at the tip nodes are K-dimensional vectors.
- *To find*: The set of non-inferior packets at the root node of the game tree.

To address the above problem, we will first analyze the problem of finding the minimal set of packets at a node n when the set of packets for the player at node n is given. For this purpose we shall use an algebra called *Dominance Algebra*. Subsequently, we shall address the problem of determining the set of packets at a node using partial order game tree search.

3. Dominance Algebra

Given the set of packets at a node, we have to identify the set of non-inferior packets (that is, those that may be selected by the player at that node). For convenience of representation we use an algebra to describe the type of operations that take place at the MAX- and MIN-nodes.

At a given node n, we have a set of packets $P_1, \ldots, P_m$ for the player who makes the move at that node. If n is a MAX-node, then we denote the options of player-1 at node n by a *MAX-expression* F_n as follows:

$$F_n = P_1 +_{\max} P_2 +_{\max} \cdots +_{\max} P_m.$$

The operator $+_{\max}$ is a commutative operator. Following the definition of packet dominance, we define another property of the $+_{\max}$ operator through the following *MAX-absorption law*.

MAX-absorption Law. A packet P_i at a MAX-node n can be absorbed under the following two situations:

- *Clear Dominance*. There exists a packet P_j at node n such that each outcome in P_j dominates (or is equal to) some outcome in P_i, and at least one outcome in P_j dominates an outcome in P_i. If this condition holds then:

  $$P_j +_{\max} P_i = P_j.$$

  In other words, P_i is absorbed in the MAX-expression F_n at node n.
- *Dominance by Freedom of Choice*. There exists a set of packets $P'_1, \ldots, P'_j$ at node n such that each of them contains fewer outcomes than P_i, and the union of them yields P_i. If this condition holds then:

  $$(P'_1 +_{\max} \cdots +_{\max} P'_j) +_{\max} P_i = P'_1 +_{\max} \cdots +_{\max} P'_j.$$

  In other words, P_i is absorbed in the MAX-expression F_n at node n.

Thus the $+_{\max}$ operator is actually the dominance operator over packets of outcomes. Using the MAX-absorption law, we can obtain a minimal MAX-expression at node n. A MAX-expression is actually a *set* of packets over which we can apply the MAX-absorption law to eliminate dominated packets. Therefore, throughout this paper we may

use statements such as "a packet belongs to a MAX-expression" or "a packet is in a MAX-expression".

We now prove that given the set of packets at node n, application of the MAX-absorption law leads to a unique minimal MAX-expression F_n for that node. The following lemma shows that the sequence in which dominated packets are absorbed does not affect the final MAX-expression.

Lemma 3.1. *If F is a MAX-expression and P_1 and P_2 are packets such that $F +_{max} P_1 +_{max} P_2 = F$ (through some sequence of application of MAX-absorption law) then $F +_{max} P_1 = F$ and $F +_{max} P_2 = F$.*

Proof. It is easy to see that, if $F +_{\max} P_1 \neq F$ and $F +_{\max} P_2 \neq F$, then $F +_{\max} P_1 +_{\max} P_2$ cannot be equal to F. Without loss of generality, let $F +_{\max} P_1 = F$. (There is no loss of generality because $+_{\max}$ is commutative.) Now, if P_1 is not instrumental in the absorption of P_2, then the result follows trivially. Let us consider the cases where P_1 is used in the absorption of P_2. If P_1 is used in the absorption of P_2, then each outcome in P_1 must either dominate or be equal to some outcome in P_2.

The MAX-expression F can absorb P_1 through *clear dominance* or through *freedom of choice*. We analyze both situations:

- *Clear Dominance.* If F absorbs P_1 through clear dominance, then there exists a packet P_i in the MAX-expression F, such that P_i clearly dominates P_1. Then each outcome in P_i either dominates or is equal to some outcome in P_1, and at least one outcome in P_i dominates an outcome in P_1. Since each outcome in P_1 in turn either dominates or is equal to some outcome in P_2, we find that P_i absorbs P_2 through clear dominance.
- *Dominance by Freedom of Choice.* If F absorbs P_1 through freedom of choice then we can have two cases:
  (1) If P_1 absorbs P_2 through clear dominance, then there exists an outcome $\vec{x}$ in P_1 that dominates an outcome in P_2. Each outcome in P_1 must belong to some packet P_i in the MAX-expression F (from the definition of absorption by freedom of choice). Let P_i be the packet containing $\vec{x}$. Since every outcome in P_i is equal to some outcome in P_1, we find that P_i absorbs P_2 through clear dominance.
  (2) Since F absorbs P_1 through freedom of choice, there exist packets $P'_1, \ldots, P'_J$ in the MAX-expression F, such that the set of packets $S_1 = \{P'_1, \ldots, P'_J\}$ dominates P_1 by freedom of choice. Now if P_1 is used to absorb P_2 by freedom of choice then there exists a set S_2 of packets in the MAX-expression F, such that $S_2 \cup \{P_1\}$ absorbs P_2. It is easy to see then that $S_2 \cup S_1$ can absorb P_2 by freedom of choice. Therefore F absorbs P_2 by freedom of choice. □

The lemma effectively states that if P_1 is instrumental in the absorption of P_2, and P_1 is absorbed by F before the absorption of P_2 then F will also absorb P_2.

Theorem 3.2. *Given the set of packets for player-1 at a MAX-node n, application of the MAX-absorption law leads to a unique minimal MAX-expression for node n.*

Proof. If a packet is instrumental in the absorption of another packet then Lemma 3.1 shows that even if the former packet is absorbed earlier, the latter packet will still be absorbed. Thus the absorption of the dominated packets is independent of the order in which the packets are absorbed. The result follows. □

Given the set of packets for player-1 at a MAX-node we can individually test each packet to see whether it is absorbed by the other packets. Theorem 3.2 shows that this will lead to an unique minimal set of packets at the node. It may be easily shown that a similar analysis may be applied to the packets for player-2 (the opponent) at a MIN-node. If n is a MIN-node, then we denote the options of player-2 at node n by a *MIN-expression* F_n as follows:

$$F_n = P_1 +_{\min} P_2 +_{\min} \cdots +_{\min} P_m,$$

where $P_1, \ldots, P_m$ are packets for player-2 at node n. The operator $+_{\min}$ is a commutative operator similar to $+_{\max}$ except that it obeys the following *MIN-absorption law*.

MIN-absorption Law. A packet P_i at a MIN-node n can be absorbed under the following two situations:

- *Clear Dominance.* There exists a packet P_j at node n such that each outcome in P_j is dominated by (or is equal to) some outcome in P_i, and at least one outcome in P_j is dominated by an outcome in P_i. If this condition holds then:

$$P_j +_{\min} P_i = P_j.$$

- *Dominance by Freedom of Choice.* There exists a set of packets $P'_1, \ldots, P'_J$ at node n such that each of them contains fewer outcomes than P_i, and the union of them yields P_i. If this condition holds then:

$$(P'_1 +_{\min} \cdots +_{\min} P'_J) +_{\min} P_i = P'_1 +_{\min} \cdots +_{\min} P'_J.$$

Theorem 3.3. *Given the set of packets for player-2 at a MIN-node n, application of the MIN-absorption law leads to a unique minimal MIN-expression for node n.*

Proof. On lines similar to the proof of Theorem 3.2. □

Theorem 3.2 and Theorem 3.3 effectively prove that the $+_{\max}$ and $+_{\min}$ operators are associative. Dominance Algebra consists of sets (packets) of K-dimensional vectors and the two commutative and associative operators $+_{\max}$ and $+_{\min}$. Given the MAX-expression at a MAX-node (or a MIN-expression at a MIN-node), we can use the MAX-absorption law (MIN-absorption law) to obtain the minimal MAX-expression (MIN-expression) at that node.

4. Finding the packets

The question that has been answered so far is how to determine the minimal set of packets at a node when the entire set of packets at that node is given. Let us now

address the problem of identifying the set of packets at a node. In particular, we address the following problems:

(1) How to find the minimal set of packets for player-1 at a given MAX-node n when the minimal set of packets for player-2 at the child MIN-nodes is given.
(2) How to find the minimal set of packets for player-2 at a given MIN-node n when the minimal set of packets for player-1 at the child MAX-nodes is given.

For the first problem, we define a function called *MIN-to-MAX* that converts a given MIN-expression to a MAX-expression. The function is based on the following result.

Lemma 4.1. *Let $\{P_1, \ldots, P_m\}$ be the minimal set of packets for player-2 at a MIN-node n. If we construct a set S of outcomes by selecting one outcome from each P_i, $1 \leqslant i \leqslant m$, then S is a packet for the parent MAX-node of n.*

Proof. We show that there exists a strategy for player-1 such that the final outcome either dominates or is equal to some outcome in S. If player-2 does not make a mistake, then it will select one of the packets $P_1, \ldots, P_m$. Without loss of generality, let us assume that player-2 selects the packet P_i. Then from the definition of a packet there exists a strategy for player-1 to reach any desired outcome from P_i. The result follows because, by the construction of S, one outcome in P_i belongs to S.

If player-2 makes a mistake, then it will select a packet P' that can be absorbed by $P_1 +_{\min} \cdots +_{\min} P_m$. Then, *every* outcome in those packets that are instrumental in the absorption of P' will be either dominated by or equal to some outcome in P'. Thus, if player-2 selects P', then there exists a strategy for player-1 to reach an outcome from P' which is either better than some outcome in S or equal to it. It follows that S is a packet for player-1. □

MIN-to-MAX(F: MIN-expression)

(1) Let $F = P_1 +_{\min} P_2 +_{\min} \cdots +_{\min} P_m$.
(2) Construct all possible sets of outcomes by selecting one outcome from each packet P_i, $1 \leqslant i \leqslant m$. Let these sets be $S_1, S_2, \ldots, S_J$.
(3) Using the MAX-absorption law, minimize the MAX-expression:

$$F' = S_1 +_{\max} S_2 +_{\max} \cdots +_{\max} S_J.$$

(4) Return the MAX-expression F'.

Lemma 4.2. *If n_i is the ith child of the MAX-node n and F_{n_i} is the MIN-expression corresponding to the MIN-node n_i, then the set of packets for player-1 at node n through the child n_i can be represented by the MAX-expression returned by MIN-to-MAX(F_{n_i}).*

Proof. From Lemma 4.1 it follows that the set of packets constructed by the function *MIN-to-MAX* are packets for player-1 at node n. Once the game reaches node n_i, there exists a strategy for player-2 to ensure that the game reaches either some outcome from the selected packet at node n_i, or an outcome that is dominated by some outcome from the selected packet. Therefore, through node n_i player-1 can (at best) reach only those outcomes that are present in the packets at node n_i. The result follows. □

Theorem 4.3. *If F_{n_i} denotes the MIN-expression of the ith child of the MAX-node n, then the MAX-expression of the node n is:*

$$F_n = \textit{MIN-to-MAX}(F_{n_1}) +_{max} \textit{MIN-to-MAX}(F_{n_2}) +_{max} \cdots +_{max} \textit{MIN-to-MAX}(F_{n_m}),$$

where m denotes the number of children of node n.

Proof. Follows from Lemma 4.2. □

The above analysis shows that using the *MIN-to-MAX* function, we can construct the set of packets (for player-1) at a MAX-node when the set of packets (for player-2) is given for each child MIN-node. The set of packets at a MIN-node can be constructed in a very similar fashion using the following *MAX-to-MIN* function.

MAX-to-MIN(F: MAX-expression)
(1) Let $F = P_1 +_{\max} P_2 +_{\max} \cdots +_{\max} P_m$.
(2) Construct all possible sets of outcomes by selecting one outcome from each packet P_i, $1 \leqslant i \leqslant m$. Let these sets be $S_1, S_2, \ldots, S_J$.
(3) Using the MIN-absorption law, minimize the MIN-expression:

$$F' = S_1 +_{\min} S_2 +_{\min} \cdots +_{\min} S_J.$$

(4) Return the MIN-expression F'.

Theorem 4.4. *If F_{n_i} denotes the MAX-expression of the ith child of the MIN-node n, then the MIN-expression of the node n is:*

$$F_n = \textit{MAX-to-MIN}(F_{n_1}) +_{min} \textit{MAX-to-MIN}(F_{n_2}) +_{min} \cdots +_{min} \textit{MAX-to-MIN}(F_{n_m}),$$

where m denotes the number of children of node n.

Proof. On lines similar to the proof of Theorem 4.3. □

At the leaf nodes of the game tree, the values themselves are the packets. Using the *MAX-to-MIN* and *MIN-to-MAX* functions and the absorption laws, we can now compute (in a bottom-up manner) the minimal set of packets at each node of the game tree (and ultimately those at the root node). Let us consider an example.

Example 4.5. Consider the game tree in Fig. 5. Let F_i denote the MAX-expression at the MAX-node i and the MIN-expression at the MIN-node i. It is easy to see that $F_4 = \{(5,4)\} +_{\min} \{(3,10)\}$ and $F_5 = \{(11,5)\} +_{\min} \{(5,9)\}$. Therefore, *MIN-to-MAX*(F_4) is $\{(5,4),(3,10)\}$ and *MIN-to-MAX*(F_5) is $\{(11,5),(5,9)\}$. The MAX-expression F_3 can be computed as:

$$F_3 = \{(5,4),(3,10)\} +_{\max} \{(11,5),(5,9)\} = \{(11,5),(5,9)\}.$$

Likewise, it is easy to see that $F_7 = \{(12,7)\} +_{\min} \{(7,12)\}$ and $F_{15} = \{(13,6)\} +_{\min} \{(6,13)\}$. Therefore, *MIN-to-MAX*(F_7) is $\{(12,7),(7,12)\}$ and *MIN-to-MAX*(F_{15}) is $\{(13,6),(6,13)\}$. The MAX-expression F_6 can be computed as:

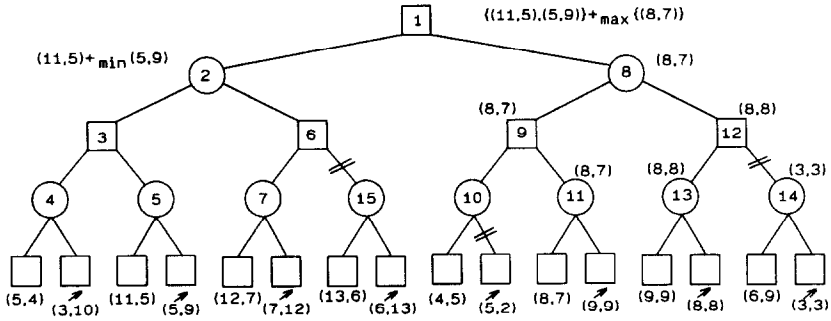


Fig. 5. Game tree showing partial order pruning.

$$F_6 = \{(12,7),(7,12)\} +_{\max} \{(13,6),(6,13)\}.$$

MAX-to-MIN(F_3) can be computed as $\{(11,5)\} +_{\min} \{(5,9)\}$. Also, *MAX-to-MIN*($F_6$) can be computed as:

$$\{(12,7),(13,6)\} +_{\min} \{(12,7),(6,13)\} +_{\min} \{(7,12),(13,6)\}$$
$$+_{\min}\{(7,12),(6,13)\}.$$

The MIN-expression F_2 can now be computed as:

$$F_2 = \{(12,7),(13,6)\} +_{\min} \{(12,7),(6,13)\} +_{\min} \{(7,12),(13,6)\}$$
$$+_{\min}\{(7,12),(6,13)\} +_{\min} \{(11,5)\} +_{\min} \{(5,9)\}$$
$$= \{(11,5)\} +_{\min} \{(5,9)\}.$$

Looking at the other side of the game tree, the MIN-expression at node 10 is $F_{10} = \{(4,5)\} +_{\min} \{(5,2)\}$ and that at node 11 (using the MIN-absorption law) is $F_{11} = \{(8,7)\}$. Using, the *MIN-to-MAX* function on F_{10} and F_{11} we obtain the MAX-expression at node 9 as follows:

$$F_9 = \{(4,5),(5,2)\} +_{\max} \{(8,7)\} = \{(8,7)\}.$$

Similarly, using the MIN-absorption law we have $F_{13} = \{(8,8)\}$ and $F_{14} = \{(3,3)\}$. Therefore the MAX-expression at node 12 is:

$$F_{12} = \{(8,8)\} +_{\max} \{(3,3)\} = \{(8,8)\}.$$

The MIN-expression at node 8 can now be computed as:

$$F_8 = \{(8,7)\} +_{\min} \{(8,8)\} = \{(8,7)\}.$$

Now, *MIN-to-MAX*(F_2) is $\{(11,5),(5,9)\}$ and *MIN-to-MAX*(F_8) is $\{(8,7)\}$. Therefore, the MAX-expression at node 1 is:

$$F_1 = \{(11,5),(5,9)\} +_{\max} \{(8,7)\}.$$

Thus at the root node of the game tree of Fig. 5, player-1 has two packets to choose from, namely $\{(11,5),(5,9)\}$ and $\{(8,7)\}$.

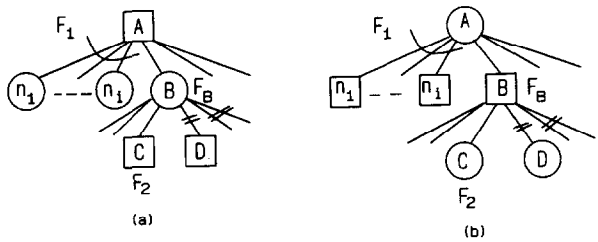


Fig. 6. Shallow pruning.

5. Partial order α-β pruning

In the previous section, we have seen how the set of non-dominated packets can be identified. However, the bottom-up approach is a brute force method that may require too much time and space. If we can find conditions under which certain branches of the game tree can be pruned, then we can apply techniques similar to α-β pruning to achieve our objective more efficiently.

We define two boolean functions $max_prune(F_1, F_2)$ and $min_prune(F_1, F_2)$ as follows:

Definition 5.1 (*Dominance test for pruning*).

- $max_prune(F_1, F_2)$: If every packet in the MAX-expression F_2 is absorbed by the MAX-expression F_1 using clear dominance (of the MAX-absorption law), then $max_prune(F_1, F_2)$ is true, else it is false.
- $min_prune(F_1, F_2)$: If every packet in the MIN-expression F_2 is absorbed by the MIN-expression F_1 using clear dominance (of the MIN-absorption law), then $min_prune(F_1, F_2)$ is true, else it is false.

Using the above functions, we define pruning conditions in partial order game trees which are somewhat analogous to α-β pruning in conventional game trees.

5.1. Shallow α-β pruning

The following lemmas help in identifying shallow pruning conditions.

Lemma 5.2. *Consider the game tree in Fig.* 6(a). F_1 *denotes the MAX-expression obtained at node A by collecting the packets (for player-*1) *backed up by the children* $n_1, \ldots, n_i$ *only.* F_2 *denotes the MAX-expression for the entire set of packets (for player-*1) *backed up at node C.* F_B *denotes the MIN-expression for the entire set of packets backed up (for player-*2) *at node B. If* $max_prune(F_1, F_2)$ *is true, then the following equality holds.*

$$F_1 +_{max} MIN\text{-}to\text{-}MAX(F_B) = F_1.$$

Proof. Let us assume the contrary. Then there exists a packet P in *MIN-to-MAX*(F_B) which cannot be absorbed by F_1. This means that if player-1 takes the move to B, there exists a strategy of player-1 to reach an outcome in P or some outcome which dominates an outcome in P. Thus, even if player-2 takes the move to C, then there exists a subset P' of P, such that player-1 is able to reach some outcome in P' (or some outcome that dominates an outcome in P'). Therefore P' is either a packet at node C, or dominated by some packet at node C. In either case, P' can be absorbed by some packet P_i in F_1 using clear dominance. Since P' is a subset of P, it follows that P can also be absorbed by P_i using clear dominance. This contradicts our assumption that the packet P cannot be absorbed by F_1. The result follows. □

Lemma 5.2 shows that shallow pruning can be effected on the other successors of node B when *max_prune*(F_1, F_2) is true, where F_1 and F_2 are as described in the statement of the lemma. It may be noted that in Fig. 6(a), if F_1 absorbs a packet of F_2 using freedom of choice (in the MAX-absorption law), then the pruning is not possible. For example, let F_1 be $\{(11,5)\} +_{\max} \{(5,7)\}$ and F_2 be $\{(11,5),(5,7)\}$. Let the MAX-expression F_D at node D be $\{(2,9)\}$. Let there be no other children of node B (except nodes C and D). Then node B can back up the packet $\{(11,5),(5,7),(2,9)\}$ at node A, which cannot be absorbed by F_1. If node D would have been pruned, then this packet would not be backed up. This is the reason for our using only the clear dominance criteria (of the absorption laws) for defining the pruning conditions.

Reasoning in a similar way, we can define pruning conditions where the function *min_prune* is applicable.

Lemma 5.3. *Consider the game tree in Fig.* 6(b). F_1 *denotes the MIN-expression obtained at node A by collecting the packets (for player-*2*) backed up by the children* $n_1, \ldots, n_i$ *only.* F_2 *denotes the MIN-expression for the entire set of packets (for player-*2*) backed up at node C.* F_B *denotes the MAX-expression for the entire set of packets backed up (for player-*1*) at node B. If min_prune*(F_1, F_2) *is true, then the following equality holds.*

$$F_1 +_{min} \textit{MAX-to-MIN}(F_B) = F_1.$$

Proof. On lines similar to the proof of Lemma 5.2. □

5.2. Deep α-β pruning

Deep pruning conditions can also be identified for partial order game trees as follows.

Lemma 5.4. *Consider a path in a game tree as shown in Fig.* 7(a). F_i *denotes the MAX-expression formed by collecting the set of packets (for player-*1*) backed up by those children of MAX-node* n_i *which are to the left of the child in the given path.* F_C *denotes the set of packets (for player-*1*) at node C. Let F denote the following MAX-expression:*

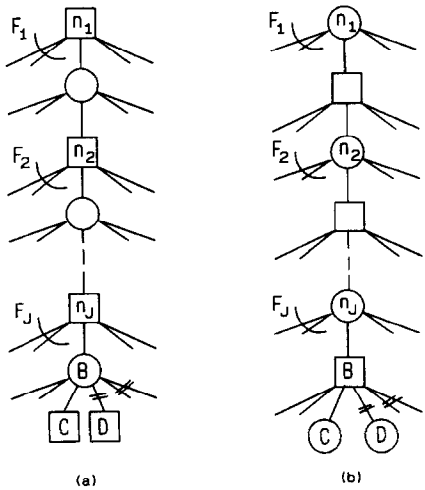


Fig. 7. Deep pruning.

$$F = F_1 +_{max} F_2 +_{max} \cdots +_{max} F_J.$$

If max_prune(F, F_C) is true, then at node n_J the move to node B will never be selected by player-1.

Proof. Let us assume the contrary, that is, player-1 selects the move to node B when the game reaches node n_J. Since player-2 can then select the move to node C, it is easy to see that each packet (for player-1) backed up by node B to n_J must contain a subset which is either a packet at node C or dominated by a packet at node C.

In the partial order game tree search problem, player-1 will have a set of packets of outcomes to choose from at the root node n_1. By definition, if the player-1 selects a packet, then there exists a strategy for it to ensure that the final outcome is either some outcome from that set or dominates some outcome from that set. Therefore, if the strategy of player-1 is such that it selects the packet P from the set of packets at the root node, then the game will go through those MAX-nodes n_i that has a non-dominated packet which is either some subset of P, or dominates some subset of P (and therefore, dominates P).

Therefore, if the game reaches node B, then player-1 must have selected a packet P at the root node such that there exists a packet P' (for player-1) backed up by node B to node n_J which is either a subset of P or dominates P. It follows that in each node n_i in the path to n_J, there exists a non-dominated packet P_i such that either P' is a subset of P_i, or P' dominates P_i.

Since P' is a packet backed up by node B, it contains a subset which is either a packet P_c at node C or dominated by a packet P_c at node C. Since *min_prune*(F, F_C) is true, there exists a packet P_i' at some node n_i which dominates P_c using clear dominance. P_i' will then also dominate P' and therefore, will dominate P_i. Thus P_i is a dominated packet at node n_i which brings us to a contradiction. The result follows. □

Using similar reasoning, we can prove the following lemma.

Lemma 5.5. *Consider a path in a game tree as shown in Fig. 7(b). F_i denotes the MIN-expression formed by collecting the set of packets (for player-2) backed up by those children of MIN-node n_i which are to the left of the child in the given path. F_C denotes the set of packets (for player-2) at node C. Let F denote the following MAX-expression:*

$$F = F_1 +_{min} F_2 +_{min} \cdots +_{min} F_J.$$

If min_prune(F, F_C) is true, then at node n_J, the move to node B will never be selected by player-1.

Proof. On lines similar to the proof of Lemma 5.4. □

Lemma 5.4 and Lemma 5.5 allows us to define *α-expressions* for MIN-nodes and *β-expressions* for MAX-nodes (using the idea of the α and β bounds in conventional game tree search).

Definition 5.6 (*α-expression*). The α-expression at a MIN-node B is the MAX-expression formed by collecting the packets (for player-1) currently backed up at all MAX ancestors of B.

Lemma 5.4 shows that the exploration of a MIN-node B can be terminated as soon as the MAX-expression backed up by any of its children is absorbed by the α-expression at B (using clear dominance).

Definition 5.7 (*β-expression*). The β-expression at a MAX-node B is the MIN-expression formed by collecting the packets (for player-2) currently backed up at all MIN-ancestors of B.

Lemma 5.5 shows that the exploration of a MAX-node B can be terminated as soon as the MIN-expression backed up by any of its children is absorbed by the β-expression at B (using clear dominance).

Using these results, we may now develop an algorithm on lines similar to the α-β pruning algorithm of conventional game tree search. Given heuristic vector estimates $e(n)$ at each terminal node n, the following procedure uses a technique similar to α-β pruning to determine the set of non-dominated packets for player-1. The call $F(s, -\vec{\infty}, +\vec{\infty})$ returns the MAX-expression at node s. $+\vec{\infty}$ is a packet containing a single K-dimensional outcome which has all dimensions equal to $+\infty$. Likewise $-\vec{\infty}$ is a packet containing a single K-dimensional outcome which has all dimensions equal to $-\infty$.

Procedure $F(n, \alpha, \beta)$
1. IF n is a terminal node THEN Return $e(n)$
2. Generate successors $m_1, m_2, \ldots, m_b$ of n

2.1. Set $i \leftarrow 1$
2.2. IF n is a MAX-node THEN
  2.2.1. Set $F_n \leftarrow -\vec{\infty}$
  2.2.2. Repeat the following steps until $i > b$
    2.2.2.1. Set $F_{m_i} \leftarrow F(m_i, \alpha, \beta)$
    2.2.2.2. IF $F_{m_i} \neq -\vec{\infty}$ THEN
      Set $\alpha \leftarrow \alpha +_{\max} F_{m_i}$
      Set $F_n \leftarrow F_n +_{\max} F_{m_i}$
      IF *min_prune*(β,*MAX-to-MIN*(F_{m_i})) is true THEN
        Return $-\vec{\infty}$
    2.2.2.3. Set $i \leftarrow i + 1$
  2.2.3. Return F_n
2.3. IF n is a MIN-node THEN
  2.3.1. Set $F_n \leftarrow +\vec{\infty}$
  2.3.2. Repeat the following steps until $i > b$
    2.3.2.1. Set $F_{m_i} \leftarrow F(m_i, \alpha, \beta)$
    2.3.2.2. IF $F_{m_i} \neq -\vec{\infty}$ THEN
      Set $\beta \leftarrow \beta +_{\min}$ *MAX-to-MIN*(F_{m_i})
      Set $F_n \leftarrow F_n +_{\min}$ *MAX-to-MIN*(F_{m_i})
      IF *max_prune*(α, F_{m_i}) is true THEN
        Return $-\vec{\infty}$
    2.3.2.3. Set $i \leftarrow i + 1$
  2.3.3. Return *MIN-to-MAX*(F_n)

The working of the algorithm is illustrated on the game tree of Fig. 5. The sequence of nodes visited, and the corresponding values of α and β is as shown in Table 1. In MAX-nodes, $F(n)$ denotes the MAX-expression and in MIN-nodes, $F(n)$ denotes the MIN-expression. For convenience of writing, we have written packets containing a single outcome (a, b) as "(a, b)" instead of "$\{(a, b)\}$".

6. Conclusion

The problem of searching game trees under a total order has been well studied in the past. In this paper we have investigated the more general problem of searching a game tree under a partial order.

The contents of this paper focuses on the general issues in partial order game tree searching. The pruning conditions and basic search strategies can be enhanced to cater to the formulations of specific problems. For example, it is likely that if a problem involves many criteria, and each of them are modeled as separate dimensions of the cost vector, then very little pruning is possible as most outcomes become non-dominated. In order to enhance the pruning in such situations a judicious combination of some of the criteria may be called for. In the other extreme case, when all the criteria can be meaningfully combined, the proposed pruning conditions collapses to the pruning conditions of total order game tree search.

Table 1
Sequence of nodes visited by *procedure F* on the game tree of Fig. 5

n	$F(n)$	α	β
n_1, n_2, n_3	–	$-\vec{\infty}$	$+\vec{\infty}$
n_4	$(5,4) +_{\min} (3,10)$	$-\vec{\infty}$	$(5,4) +_{\min} (3,10)$
n_3	–	$\{(5,4),(3,10)\}$	$+\vec{\infty}$
n_5	$(11,5) +_{\min} (5,9)$	$\{(5,4),(3,10)\}$	$(11,5) +_{\min} (5,9)$
n_3	$\{(5,4),(3,10)\}$ $+_{\max} \{(11,5),(5,9)\}$ $= \{(11,5),(5,9)\}$	$\{(11,5),(5,9)\}$	$+\vec{\infty}$
n_2, n_6	–	$-\vec{\infty}$	$(11,5) +_{\min} (5,9)$
n_7	$(12,7) +_{\min} (7,12)$	$-\vec{\infty}$	$(11,5) +_{\min} (5,9)$ $+_{\min} (12,7) +_{\min} (7,12)$ $= (11,5) +_{\min} (5,9)$
n_6[a]	–	$\{(12,7),(7,12)\}$	$(11,5) +_{\min} (5,9)$
n_2	$(11,5) +_{\min} (5,9)$	$-\vec{\infty}$	$(11,5) +_{\min} (5,9)$
n_1, n_8, n_9	–	$\{(11,5),(5,9)\}$	$+\vec{\infty}$
n_{10}[a]	–	$\{(11,5),(5,9)\}$	$(4,5)$
n_9	–	$\{(11,5),(5,9)\}$	$+\vec{\infty}$
n_{11}	$(8,7) +_{\min} (9,9) = (8,7)$	$\{(11,5),(5,9)\}$	$(8,7)$
n_9	$(8,7)$	$\{(11,5),(5,9)\}$	$+\vec{\infty}$
n_8, n_{12}	–	$\{(11,5),(5,9)\}$	$(8,7)$
n_{13}	$(9,9) +_{\min} (8,8) = (8,8)$	$\{(11,5),(5,9)\}$	$(8,7) +_{\min} (8,8) = (8,7)$
n_{12}[a]	–	$\{(11,5),(5,9)\} +_{\max} (8,8)$	$(8,7)$
n_8	$(8,7)$	$\{(11,5),(5,9)\}$	$(8,7)$
n_1	$\{(11,5),(5,9)\} +_{\max} (8,7)$	$\{(11,5),(5,9)\} +_{\max} (8,7)$	$+\vec{\infty}$

[a] Pruning occurs at these nodes.

Acknowledgements

The authors would like to thank the reviewers for their comments, which helped in improving both the contents as well as the style of presentation of the paper.

References

[1] D.E. Knuth and R.W. Moore, An analysis of alpha-beta pruning, *Artif. Intell.* **6** (1975) 293–326.
[2] R.E. Korf, Multi-player alpha-beta pruning, *Artif. Intell.* **48** (1991) 99–111.
[3] C.H. Papadimitriou and M. Yannakakis, Shortest paths without a map, *Theoret. Comput. Sci.* **84** (1991) 127–150.
[4] J. Pearl, *Heuristics* (Addison-Wesley, Reading, MA 1984).
[5] B.S. Stewart and C.C. White, Multiobjective A^*, *J. ACM* **38** (1991) 775–814.
[6] G.C. Stockman, A minimax algorithm better than alpha-beta?, *Artif. Intell.* **12** (1979) 179–196.

ELSEVIER

Artificial Intelligence 82 (1996) 259–272

Artificial Intelligence

Support set selection for abductive and default reasoning

Bart Selman [a,*], Hector J. Levesque [b,1]

[a] AI Principles Research Department, AT&T Bell Laboratories, Murray Hill, NJ 07974, USA

[b] Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4

Received March 1993; revised September 1994

Abstract

Of all the possible ways of computing abductive explanations, the ATMS procedure is one of the most popular. While this procedure is known to run in exponential time in the worst case, the proof actually depends on the existence of queries with an exponential number of answers. But how much of the difficulty stems from having to return these large sets of explanations? Here we explore abduction tasks similar to that of the ATMS, but which return relatively small answers. The main result is that although it is possible to generate some nontrivial explanations quickly, deciding if there is an explanation containing a given hypothesis is NP-complete, as is the task of generating even one explanation expressed in terms of a given set of assumption letters. Thus, the method of simply listing all explanations, as employed by the ATMS, probably cannot be improved upon. An interesting result of our analysis is the discovery of a subtask, we call the Support Set Selection Task, that is not only at the core of generating explanations, but is also at the core of generating extensions in Reiter's default logic. Moreover, it is this subtask that accounts for the computational difficulty of both forms of reasoning. This establishes for the first time a strong connection between computing abductive explanations and computing extensions in default logic.

1. Introduction

Of all the possible ways of computing abductive explanations, the procedure employed by an assumption-based truth maintenance system (ATMS) is one of the most

* Corresponding author. E-mail: selman@research.att.com.

[1] Fellow of the Canadian Institute for Advanced Research. Supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada. E-mail: hector@cs.toronto.edu.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0004-3702(94)00069-7

popular [3,29]. It is therefore somewhat surprising that so little effort has gone into understanding in precise terms the nature of the computational task performed by an ATMS, that is, the *what* for which an ATMS is a *how*.

What do we know in general about this task? It has been known since at least 1985 that in the worst case, any procedure that computes what the ATMS computes will need time that is exponential in the length of its input [24]. This is because there are problems for which the desired set of answers (where intuitively, each answer is a set of assumptions that would *explain* a given condition) is exponentially large. Perhaps this simple fact has discouraged further theoretical analysis into the worst-case difficulty of computing explanations.

But it doesn't tell the whole story. Is the fact that an ATMS can take exponential time *only* due to cases where an exponential number of answers need to be returned? What if instead of generating all the answers, we only required a procedure to reply to simple yes/no questions about them, such as whether or not there is an explanation containing a given assumption? Furthermore, in many (if not most) applications, we expect to be dealing with a very small number of explanations. For example, in circuit diagnosis, explanations involve sets of possibly faulty components [25,30], and one would not expect k components to break down independently, for large k. Is it still the case that generating a small number of explanations is hard? In other words, if an ATMS still runs in exponential time for problems like this (as it apparently does [26]), should we be looking for a different procedure, or is this the best that can be expected?

In this paper, we attempt to answer these questions. In particular, we show that certain natural variations of the ATMS task that do not require enumerating all the answers are nonetheless NP-hard. In our view, this provides concrete evidence that the ATMS is doing as well as can be expected.

But something more fundamental came out of the analysis. We were surprised to discover a strong connection between computing explanations on the one hand, and computing extensions in Reiter's default logic [28], on the other. It turns out that both tasks share a common computational core. Moreover, it is this common subtask that leads to the computational difficulty of both abductive and default reasoning. Apart from the fact that both forms of reasoning use the word "assumption", this is the first result that we know of to show a clear relationship between the computational properties of these two apparently very different forms of reasoning.

The rest of the paper is organized as follows. In the next section, we review Reiter and de Kleer's analysis of the ATMS. Next, we show that while it is always easy to find at least one nontrivial explanation, determining if there is one containing a given assumption, or finding one that is expressed only in terms of a given assumption set is NP-hard. In Section 4, we consider a weaker version of the ATMS task (which we call the Support Selection Task), where explanations are not required to be minimal, and show that it too is NP-hard. In Section 5, we briefly review the definitions from Reiter's default logic, and show that the problem of computing an extension is a variant of the Support Selection Task where we care about maximality instead of minimality. It turns out that making a support set minimal or maximal is easy; it is finding one in the first place that is hard. Finally, some conclusions are drawn.

2. Abduction

In this section, we formally define what constitutes an explanation as computed by the ATMS [29]. We will assume a standard propositional language $\mathcal{L}$ with propositional letters from the set P. We will use p, q, r, s, and t (possibly with subscripts) to denote propositional letters. A *clause* is a disjunction of literals (a *literal* is either a propositional letter, called a positive literal, or its negation, called a negative literal). We will represent a clause by the set of literals contained in the clause. A clause is called a *unit* clause if it only contains a single literal. A clause is *Horn* iff it contains at most one positive literal. A set of Horn clauses will be called a *Horn theory*.

Definition 1 (*Explanation*). Given a set of clauses Σ, called the background theory, and a letter q, an explanation for q is a minimal set of unit clauses α such that

(1) $\Sigma \cup \alpha \models q$, and
(2) $\Sigma \cup \alpha$ is consistent.

For a discussion on the desirability of the above properties, see [22].[2] Instead of expressing explanations as sets of unit clauses, we will often give the logical equivalent form consisting of the conjunction of the literals occurring in the clauses, e.g., we write $p \wedge q \wedge r$ instead of $\{\{p\}, \{q\}, \{r\}\}$.

Example. Let Σ be the set $\{\{p\}, \{q\}, \{\overline{p}, \overline{r}, \overline{s}, t\}\}$. The conjunctions $r \wedge s$ and t are explanations for t. We call t the trivial explanation for t; our interest lies of course in the other, nontrivial explanations.

The notion of explanation defined above is somewhat more general than the one employed in the ATMS. The ATMS only computes a certain subset of these explanations, namely those whose unit clauses are drawn from a distinguished subset of the propositional letters, called *assumptions*. Assumptions stand for the hypotheses that we are willing to consider in the explanations, such as the possible failure of a component in circuit diagnosis. (See [23], for a novel application of abductive explanations in testing.)

Definition 2 (*Assumption-based explanation*). Given a set of clauses Σ, a set of assumptions $A \subseteq P$, and a letter q, called the query, α is an assumption-based explanation iff

(1) α is an explanation of q given Σ, and
(2) α only contains letters from A.

Note that when the assumption set includes all symbols in the language, every explanation is also an assumption-based one. Also, in the ATMS the background theory Σ is Horn.

[2] For a quite different definition of explanation, see [1,27].

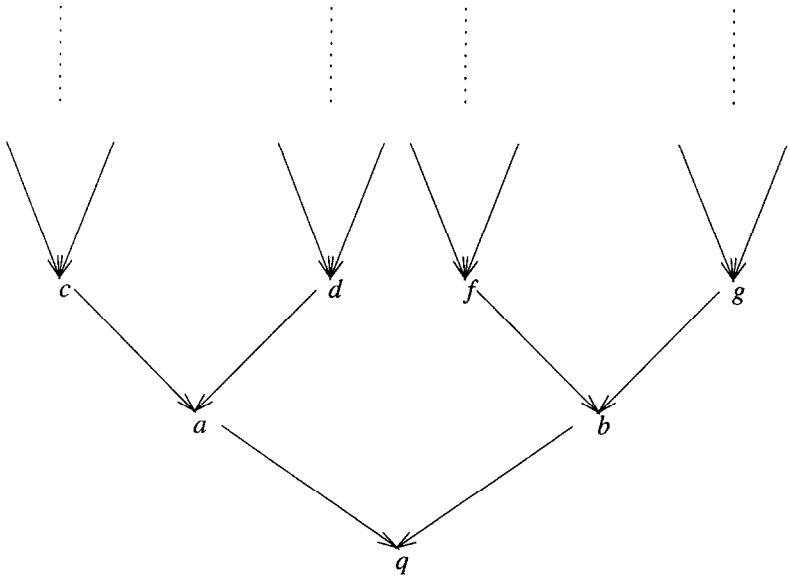


Fig. 1. Theory with exponentially many explanations. The theory contains the clauses $(a \wedge b) \supset q$, $(c \wedge d) \supset a$, $(f \wedge g) \supset b$, etc. Consider explanations for q. We get q, $a \wedge b$, $c \wedge d \wedge b$, $a \wedge f \wedge g$, $c \wedge d \wedge f \wedge g$, etc. The total number of explanations is given by the recurrence relation $f(k) = f^2(k-1) + 1$, where k is the number of levels. So, $f(k) \geqslant 2^{2^{k-2}}$. Since $k = \lceil \log(n) \rceil$, where n is the total number of nodes, it follows that the total number of explanations grows exponentially in n. Example from Reiter (personal communication, 1989).

3. Computing explanations

We will now consider the computational cost of generating explanations. As is well-known, there may be exponentially many explanations for a given letter. McAllester [24] and de Kleer [4] showed this by using an example based on an encoding of the parity function.[3] Another, perhaps somewhat more natural, example is given in Fig. 1. So, clearly, listing all explanations may require exponential time.[4] However, this leaves open the question of what the complexity of finding some explanation is. In particular, what is the complexity of finding a nontrivial one?

In case Σ contains arbitrary clauses, finding any explanation is easily shown to be NP-hard.[5] However, the following theorem shows that when Σ is a Horn theory, a nontrivial explanation (if one exists) can be computed efficiently.

[3] Rutenburg [31] notes that encoding parity requires an exponential size theory [15], which would imply that the number of explanations is only polynomial in the size of the background theory. However, McAllester and de Kleer use auxiliary letters to obtain a theory of polynomial size (in the number of propositional letters).

[4] In fact, there may be exponentially many *assumption-based* explanations, and therefore the worst-case complexity of the ATMS, which lists all of them, is clearly exponential.

[5] Since explanations only exist when Σ is consistent, an explanation procedure can be used to test the satisfiability of a set of clauses.

Theorem 3. *Given a set of Horn clauses Σ and a letter q, a nontrivial explanation for q can be computed in time* $O(kn)$, *where k is the number of propositional letters and n is the number of occurrences of literals in Σ.*

Proof. Here we give only an outline of the algorithm. Consider a clause in Σ of the following form: $\{\overline{q}_1, \ldots, \overline{q}_k, q\}$ with $k \geqslant 1$ (if no such clause exists, return "no nontrivial explanation"). Now, clearly $q_1 \wedge \cdots \wedge q_k$ with Σ implies q. Subsequently, try removing a letter from this conjunction while ensuring that the remaining conjunction together with Σ still implies q (testing can be done in linear time using the Dowling and Gallier [9] procedure, since W is Horn). Repeat this process until no more letters can be removed. If the remaining conjunction is non-empty and combined with Σ is consistent, return that one; otherwise consider another clause containing q and repeat the above procedure. When all clauses containing q have been explored and no explanation is found, return "no nontrivial explanation". (To see that the latter is correct consider the set of clauses W' consisting of a nontrivial explanation combined with the original theory. W' must be consistent and combined with $\overline{q}$ should be inconsistent. Now, to create the inconsistency with $\overline{q}$ at least one of the clauses in W' containing q should be "activated". The negative literals in this clauses will form a superset of some nontrivial explanation and would have been found by the algorithm.) A direct implementation of this approach would give us $O(n^2)$ time complexity; a slightly more sophisticated implementation runs in time $O(kn)$. □

It is clear that the above algorithm only generates certain, very particular explanations—which ones depend on the way the background knowledge Σ is expressed. But if there are *some* nontrivial explanations that are easy to find, could it be that in some sense they are *all* easy to find, even if there are too many to list?[6] One way to look at this question is to consider a procedure that generates only a single explanation, but must return different ones for different arguments. For example, if we can ask for an explanation containing the letters in S_1 but not containing those in S_2, clearly we can generate arbitrary explanations.[7] Unfortunately, the following theorem show that there can be no efficient procedure for this form of "goal-directed" abduction, even if the set S_1 contains only a single literal and S_2 is empty.

Theorem 4. *Given a set of Horn clauses Σ and letters p and q, the problem of generating an explanation for q that contains p is* NP-*complete.*

Proof. First we show that the problem is in NP. We therefore show that a solution can be verified in polynomial time. Consider as a solution candidate the set of unit clauses α. First check that α contains p. Then verify that $\alpha \cup \Sigma$ is consistent and implies q, because W is Horn this can be done in polynomial time. Finally, one can verify the minimality of α, by trying to remove each of its unit clauses (one at a time) and checking whether

[6] Note that, for example, one can efficiently list all models (satisfying truth assignments) of a Horn theory. See also Dechter and Itai [2].

[7] The set S_1 could be used, for example, to identify components that have a high failure rate when doing circuit diagnosis. For a related approach, see [7].

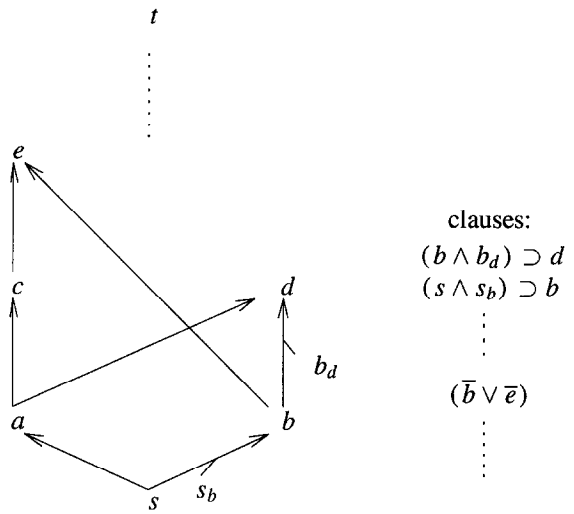


Fig. 2. Example illustrating the construction used in our NP-completeness result for abduction. The pair (b, e) is a forbidden pair.

the remaining set together with Σ still implies q. If no such unit clause exists then α is minimal, and thus a valid explanation containing p.

Next, we show that the problem is as hard as any problem in NP. The proof is based on a reduction from the NP-complete decision problem "path with forbidden pairs" (or PWFP) defined in [16, 17]. An instance of PWFP consists of a directed graph $G = (V, E)$, specified vertices $s, t \in V$, and a collection $C = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ of pairs of vertices from V. The question is: Does there exist a path from s to t in G that contains at most one vertex from each pair in C? This problem remains NP-complete even if we consider only acyclic graphs.

Given an instance of this restricted version of PWFP, we construct a background theory Σ. Identifying the vertices of the graph with propositional letters, Σ contains the following clauses: (1) for each directed edge (x, y), the clause $\{\overline{x_y}, \overline{x}, y\}$ where x_y is a new propositional letter, and (2) for each forbidden pair (a_i, b_i), the clause $\{\overline{a_i}, \overline{b_i}\}$. Fig. 2 illustrates the reduction. Now, consider an explanation for t that contain s. It is relatively straightforward to show that if such an explanation exists, it will consist of a set of unit clauses of the form $\{x_y\}$ that uniquely identify a path from s to t in the original graph. Moreover, because of the clauses in group (2), such a path goes through at most one vertex of each forbidden pair. Thus, any procedure that can generate an explanation for t containing s can also be used to solve PWFP, and so this explanation problem is as hard as any problem in NP. □

Intuitively speaking, Theorem 4 shows that certain explanations will be hard to find, even if our background theory Σ is Horn. And, as can be seen from the reduction, this

result also holds when Σ consists of an acyclic Horn theory.[8]

Let us now consider the influence of an assumption set as used in the ATMS. Recall that the assumption set A is a distinguished subset of the propositional letters and that given a query q, the ATMS will generate only explanations that contain letters from among those in A. Note that the assumption set again allows one to select a certain subset of all possible explanations. This way of selecting certain explanations is related, but *not* identical, to the notion of goal-directed abduction. The following theorem shows that the use of such an assumption set again dramatically increases the complexity of finding a nontrivial explanation (compare with Theorem 3):

Theorem 5. *Given a set of Horn clauses Σ, a set of assumptions A, and a query letter q, finding an assumption-based explanation for q is* NP-*complete.*

Proof. It is again easy to see that the problem is in NP because consistency checking and inference for Horn theories can be done in polynomial time. For the completeness part, we modify the reduction used in the proof of Theorem 4: Add the clause $\{s\}$ to the background theory, and let the assumption set contain all letters of the form x_y. Now, an assumption-based explanation will consist of a subset of the letters in the assumption set, and as above, this set will uniquely identify a path from s to t not containing any forbidden pair. Again, the problem remains NP-complete even for acyclic theories. □

This theorem shows that apart from the fact that the ATMS may have to list an exponential number of explanations, merely finding one of them may require exponential time.

Let us now consider the experimental observation, reported by Provan [26], that the ATMS can exhibit exponential behaviour even if the background theory is such that there are only a few assumption-based explanations for the query letter. Provan argues that such restricted theories have practical significance, for example, in scene interpretation since, although there may many local ambiguities in scene interpretation tasks, there are generally only a few globally consistent interpretations for any given scene.

The following result shows that in fact the intractability observed by Provan is *inherent* in the task, and not simply caused by the particular procedure employed in the ATMS:

Theorem 6. *Given a set of assumptions A, a query q, and a set of Horn clauses Σ such that q has at most one assumption-based explanation, finding this explanation is* NP-*hard under randomized reductions.*

Proof. We use a result by Valiant and Vazirani [36], who show that determining propositional satisfiability remains hard (unless NP = RP, considered unlikely) even if one guarantees that the given instances of SAT has at most one satisfying truth assignment. Since the reduction from SAT to PWFP and the reduction from PWFP

[8] Given a Horn theory Σ, let G be a directed graph containing a vertex for each letter in Σ and an edge from any vertex corresponding to a letter on the left-hand side of a Horn rule to the vertex corresponding to the letter on the right hand-side of that rule. A Horn theory is *acyclic* if and only if the associated graph G is acyclic.

to assumption-based explanations are parsimonious (i.e., the number of solutions is preserved), it follows that even if we guarantee that there is at most one assumption-based explanation for the query letter, finding it still remains hard. So the problem of generating assumption-based explanations even for such special restricted background theories remains intractable. □

One remaining question is whether the problem is still hard if we are guaranteed that there are only a few explanations overall (*including* the non-assumption-based ones) for the query. Note that we can always generate at least one nontrivial explanation (see Theorem 3); we conjecture however that generating $O(n)$ of them is again NP-hard (possibly under randomized reductions), where n is the number of propositional letters in the language.[9]

Finally, given these negative results about the difficulty of dealing with Horn theories, even acyclic ones, one might wonder whether one can trade expressiveness for tractability in abductive reasoning. Fortunately, our reduction in Fig. 2 suggests one further restriction. Note that the clauses containing two negative literals are central to the reduction. Such clauses represent *incompatibilities* between possible hypotheses. Now, when we disallow such clauses, and restrict ourselves to background theories containing only *positive* Horn clauses,[10] an assumption-based explanation can be found in polynomial time.

Theorem 7. *Given a set of positive Horn clauses Σ, a set of assumptions A, and a query letter q, an assumption-based explanation for q can be computed in polynomial time.*

An algorithm would start by simply trying the set of all letters in the assumption set as a potential explanation (i.e., check whether $\Sigma \cup A$ implies q). If that set is not sufficient to infer q, then there exists no assumption-based explanation for q. Otherwise, the algorithm will proceed by minimizing the set by removing letters from it (one at a time). A letter is removed if the remaining assumptions combined with the background theory still implies q. This can be checked in polynomial time since Σ is Horn. This time we do not have to check for consistency (condition (2) in Definition 1), because we only have definite Horn clauses which are always consistent. So, we can identify at least one tractable, albeit weak, form of abductive reasoning. Eshghi [11] has recently identified a somewhat less restricted background theory that also still allows for tractable abduction. Kautz et al. [19] and Khardon and Roth [21] identify a quite different class of tractable abduction problems by introducing a model-based representation. Alternatively, we can also consider some notion of "approximate" explanation, or perhaps some principled form of unsound or incomplete abduction [22]. Incomplete abduction can also be obtained by using an incomplete inference mechanism, such as the satisfiability procedure proposed by Selman et al. [33,34]. (There is a quite natural translation of the NP-complete Horn abduction task into a propositional satisfiability problem.) Another

[9] The problem is closely related to the hypergraph enumeration problem, which status remains open [10].

[10] These are clauses with exactly one positive literal, also called pure Horn clauses.

way of dealing with the intractability of the abduction task would be by translating Horn background theories into a tractable form, using ideas along the lines of [8, 32]. Finally, there has been a substantial amount of work on the use of various heuristics to improve on the worst-case exponential search behaviour of the ATMS. For example, Forbus and de Kleer [13] propose a mechanism for focusing the ATMS on the part of the search space that is directly relevant to the problem under consideration. In [5], de Kleer improves efficiency by exploiting certain locality aspects in the underlying background theory. For a further discussion on heuristic methods for improving the efficiency of the ATMS in practical applications, see [6, 14, 20].

4. The computational core

We have shown that finding an assumption-based explanation is intractable, even when the background theory Σ is an acyclic set of Horn clauses. In this section, we will isolate a subtask, called the Support Selection Task, which lies at the core of the computational difficulties.

Definition 8 (*Support Selection Task*). Given a set of Horn clauses Σ, a set of letters $A \subseteq P$, and a letter q, find a set of unit clauses α, called a support set, such that the following conditions hold:

(1) $\Sigma \cup \alpha \models q$,
(2) $\Sigma \cup \alpha$ is consistent, and
(3) α contains only letters form A.

Note that an assumption-based explanation is simply a *minimal* support set. We first consider the complexity of the Support Selection Task:

Theorem 9. *Given a Horn theory Σ, a set $A \subseteq P$, and a letter q, finding a support set for q is* NP-*complete.*

This result follows directly from a generalization of the reduction used in the proof of Theorem 5. Since the reduction does not rely on finding a minimal set of assumptions to support the query, any support set for the query will identify a path that goes from s to t containing at most one vertex from each forbidden pair.

Because an assumption-based explanation is a minimal support set, finding such an explanation is at least as hard as finding support sets. Hence, the intractability of finding an assumption-based explanation is in fact a direct consequence of Theorem 9. Stated differently, in order to establish the intractability of finding an assumption-based explanation, one need not use the fact that explanations are *minimal*. This is in contrast with the widely held belief that it's the minimality of explanations that makes them difficult to compute.

Furthermore, the minimality requirement does not further increase the computational difficulty of the task, as can be seen from the following argument. Consider a support set α for q. We can minimize this set in polynomial time by removing clauses from it

while each time verifying that the reduced set combined with Σ still implies q. Since Σ is a Horn theory this can be done in polynomial time. Finally, note that the Support Selection Task can be shown to be no harder than any problem in NP, and thus neither is generating an assumption-based explanation.

To summarize, the Support Selection Task is at the core of the ATMS-style abduction task. In the next section, we will see how this task also is at the core of goal-directed default reasoning, thereby establishing a computational connection between abductive and default reasoning.

5. Default reasoning

Default Logic, introduced by Reiter [28], is one of the more prominent formal proposals for representing and reasoning with default information. We will first briefly define Default Logic (see [12,28] for further details), and subsequently consider the complexity of default logic theories.

Reiter formalized default reasoning by extending first-order logic with default rules. A *default theory* is a pair (D, W) where D is a set of default rules and W a set of ordinary first-order formulas. A rule is of the form:

$$\frac{\alpha : \beta}{\gamma}$$

where α is the *prerequisite*, β the *conclusion*, and γ the *justification* of the rule, each of them formulas. A rule is intuitively understood as meaning that if α is known, and β is consistent with what is known, then γ may be inferred.

An *extension* is a *maximal* set of conclusions that can be drawn from a theory. But care must be taken that none of the justifications of the rules used in the construction of an extension conflict with its final contents, and that every formula in the extension can in fact be derived from W and the rules. The formal definition of an extension (from [28, Theorem 2.1]) is therefore rather complex:

Definition 10 (*Extension*). A set of formulas E is an extension for the theory (D, W) if and only if it satisfies the following equations:[11]

$$E_0 = W,$$
and for $i > 0$
$$E_{i+1} = Th(E_i) \cup \left\{ \gamma \mid \frac{\alpha : \beta}{\gamma} \in D,\ \alpha \in E_i, \text{ and } \neg\beta \notin E \right\},$$
$$E = \bigcup_{i=0}^{\infty} E_i.$$

Note the explicit reference to E in the definition of E_{i+1}. *Th* denotes logical closure.

[11] We assume that all formulas are closed, i.e., they do not contain free variables.

5.1. Computing extensions

Kautz and Selman [18] give a detailed analysis of the computational complexity of default reasoning based on Reiter's proposal. They consider a partially ordered space of more and less general propositional default logic theories. For each theory the complexity is determined of the following tasks: finding an extension (*credulous* reasoning), generating an extension that contains a given set of propositions (*goal-directed* reasoning), and the problem of determining what holds in all extensions of a default logic theory (*skeptical* reasoning). To avoid the difficulty of the consistency check needed to determine whether a rule can be applied, Kautz and Selman restrict the default theories to ones in which the set of facts W, the prerequisites, the justifications, and the consequences each consist of a set of unit clauses. Here we will consider a relaxation of these restrictions. In particular, we will allow W to contain Horn clauses.

We will show that even for extremely simple default rules and a Horn theory W goal-directed default reasoning is intractable, and that the computational difficulty is again appropriately characterized by the Support Selection Task.[12] To facilitate our discussion, a rule of the form $:p/p$ will be called an *elementary* default—these rules are the simplest possible defaults.[13]

We have the following result concerning goal-directed default reasoning:

Theorem 11. *Given an acyclic Horn theory W, a set of elementary defaults D, and a letter q, finding an extension of (D, W) that contains q is* NP-*complete.*[14]

This result strengthens a result by Stillman [35], who showed the task is NP-hard for arbitrary Horn theories with general normal unary defaults. But aside from strengthening Stillman's result, our interest in this result arises from the fact that the Support Selection Task lies again at the root of the computational difficulty of the problem, as we will see below.

To prove Theorem 11, we first consider the relation between goal-directed default reasoning and the Support Selection Task. Note that if Σ is a Horn theory and D a set of elementary defaults involving letters from a set A, then each extension of (D, Σ) is of the form $Th(\Sigma \cup \alpha)$, where α is a set of unit clauses drawn from A. Intuitively, α is the set of letters that are added to Σ via the rules in D. We have the following theorem.

Theorem 12. *Let Σ be a Horn theory, q be a letter, $A \subseteq P$ be a set of letters, and let $D = \{:p/p \mid p \in A\}$. Then, $Th(\Sigma \cup \alpha)$ is an extension of (D, Σ) that contains q if and only if α is a* maximal *support set of q.*

[12] Determining the complexity of goal-directed reasoning for such theories was given as one of the central open problems in [18].

[13] Using the terminology of Kautz and Selman [18], they are normal unary defaults with empty prerequisites and positive consequences.

[14] Since an extension consists of an infinite set of formulas because of the logical closure, we are only looking for a set of unit clauses that represents an extension.

The proof of this is somewhat tedious but relatively straightforward. From this theorem, it follows that finding extensions of the default logic theory that contains a given letter q is at least as hard as finding a support set for q. Thus, Theorem 11 follows directly from the fact that the Support Selection Task is NP-complete (Theorem 9).

Furthermore, the fact that extensions correspond to *maximal* support sets does not add further to the difficulty of computing extensions: Given a support set, one can simply try adding additional letters from the assumption set while maintaining consistency, until a maximal set is obtained. Thus, as for assumption-based explanations, the Support Selection Task is the difficult part of goal-directed default reasoning.

It is suggested in [18] that goal-directed reasoning default could be of use in resolution theorem provers that incorporate default information. Our results here suggest that such an integration will most likely run into computational difficulties. Much more promising, are credulous reasoners that search for an arbitrary extension. This task remains tractable for relatively expressive default rules combined with Horn theories. By contrast, skeptical default reasoning, i.e., determining what holds in all extensions of a theory, can be shown to be strictly harder than goal-directed reasoning, and thus our intractability result carries over to skeptical reasoning.

6. Conclusions

In this paper, we have examined the problem of computing abductive explanations. We have shown that given a Horn theory and a letter q, some nontrivial explanation for q can be calculated in polynomial time. However, goal-directed abduction or the use of an assumption set renders the problem intractable, even for acyclic Horn theories. Thus, the exponential worst-case complexity of the ATMS is not just a consequence of having to return an exponential number of answers; generating even one explanation containing letters from the assumption set is inherently difficult. It appears unlikely, therefore, that the efficiency of the ATMS algorithm can be significantly improved.

This work also shows that there is a strong connection between computing explanations and computing extensions in default logic. Our Support Selection Task is at the core of both assumption-based abductive reasoning and goal-directed default reasoning. We need to minimize support sets for the former, and maximize them for the latter, but neither is hard for Horn theories. In both cases, the difficult task is deciding on an appropriate set of assumptions to make.

Acknowledgements

We thank Ray Reiter for bringing to our attention the close relation between default reasoning and the ATMS, and for useful suggestions and discussions.

References

[1] D. Allemang, M.C. Tanner, T. Bylander and Josephson, On the computational complexity of hypothesis

assembly, in: *Proceedings IJCAI-87*, Milan, Italy (1987) 1112–1117.

[2] R. Dechter and A. Itai, The complexity of finding all solutions, UCI Technical Report 92-61, University of California at Irvine, Irvine, CA (1991).

[3] J. de Kleer, An assumption-based TMS, *Artif. Intell.* **28** (2) (1986) 127–162.

[4] J. de Kleer, Problem solving with the ATMS, *Artif. Intell.* **28** (2) (1986) 197–224.

[5] J. de Kleer, Exploiting locality in a TMS, in: *Proceedings AAAI-90*, Boston, MA (1990) 264–271.

[6] J. de Kleer, An improved incremental algorithm for generating prime implicates, in: *Proceedings AAAI-92*, San Jose, CA (1993) 780–785.

[7] J. de Kleer and B.C. Williams, Diagnosis with behavioral modes, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 1324-1330.

[8] A. del Val, Tractable databases: how to make propositional resolution complete through compilation, in: *Proceedings Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR-94)*, Bonn, Germany (1994) 551–561.

[9] W.F. Dowling and J.H. Gallier, Linear-time algorithms for testing the satisfiability of propositional Horn formulae, *J. Logic Programming* **3** (1984) 267–284.

[10] T. Eiter, and G. Gottlob, Identifying the minimal traversals of hypergraph and related problems, Preprint (1991).

[11] K. Eshghi, A tractable class of abduction problems, in: *Proceedings IJCAI-93*, Chambéry, France (1993) 3–8.

[12] D.W. Etherington, Reasoning with incomplete information: investigations of nonmonotonic reasoning, Ph.D. Thesis, Department of Computer Science, University of British Columbia, Vancouver, BC (1986); Revised version appeared as: *Reasoning with Incomplete Information* (Morgan Kaufmann, Los Altos, CA, 1988).

[13] K.D. Forbus and J. de Kleer, Focusing the ATMS, in: *Proceedings AAAI-88*, St. Paul, MN (1988) 193–198.

[14] K.D. Forbus and J. de Kleer, *Building Problem Solvers* (MIT Press, Cambridge, MA, 1993).

[15] M. Furst, J. Saxe and M. Sipser, Parity, circuits, and the polynomial time hierarchy, *Math. Syst. Theory* **17** (1984) 13–27.

[16] H.N. Gabow, S.N. Maheshwari and L. Osterweil, On two problems in the generation of program test paths, *IEEE Trans. Softw. Eng.* (1976) 227–231.

[17] M.R. Garey and D.S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness* (Freeman, New York, 1979).

[18] H.A. Kautz and B. Selman, Hard problems for simple default logics, in: *Proceedings First International Conference on Knowledge Representation and Reasoning (KR-89)*, Toronto, Ont. (1989) 189–197; Extended version in: *Artif. Intell.* **49** (1991) 243–279.

[19] H. Kautz, M. Kearns and B. Selman, Reasoning with characteristic models, in: *Proceedings AAAI-93*, Washington, DC (1993) 34–39.

[20] A. Kean and G. Tsiknis, An incremental method for generating prime implicants implicates, *J. Symbolic Comput.* **9** (1990) 185–206.

[21] R. Khardon and D. Roth, Reasoning with models, in: *Proceedings AAAI-94*, Seattle, WA (1994) 1148–1153.

[22] H. J. Levesque, A knowledge-level account of abduction, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 1061–1067.

[23] S. McIlrait, Generating tests using abduction, in: *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR-94)*, Bonn, Germany (1994) 449–460.

[24] D. McAllester, A widely used truth-maintenance system, AI Lab Memo, MIT, Cambridge, MA (1985).

[25] D. Poole, A methodology for using a default and abductive reasoning system, Technical Report, Department of Computer Science, University of Waterloo, Waterloo, Ont. (1988).

[26] G.M. Provan, Efficiency analysis of multiple-context TMSs in scene representation, in: *Proceedings AAAI-87*, Seattle, WA (1987) 173–177.

[27] J. Reggia, Diagnostic expert systems based on a set-covering model, *Int. J. Man Mach. Stud.* **19** (5) (1983) 437–460.

[28] R. Reiter, A logic for default reasoning, *Artif. Intell.* **13** (1980) 81–132.

[29] R. Reiter and J. de Kleer, Foundations of assumption-based truth maintenance systems, in: *Proceedings AAAI-87*, Seattle, WA (1987) 183–188.

[30] R. Reiter, A theory of diagnosis from first principles, *Artif. Intell.* **32** (1987) 57–96.

[31] V. Rutenburg, Propositional truth maintenance systems: Classification and complexity analysis, *Ann. Math. Artif. Intell.* **10** (1994) 207–231.

[32] B. Selman and H.A. Kautz, Knowledge compilation using Horn approximations, in: *Proceedings AAAI-91*, Boston, MA (1991) 904–909.

[33] B. Selman and H.A. Kautz, Domain-independent extensions to GSAT: solving large structured satisfiability problems, in: *Proceedings IJCAI-93*, Chambéry, France (1993) 290–295.

[34] B. Selman, H.J. Levesque and D. Mitchell, A new method for solving hard satisfiability problems, in: *Proceedings AAAI-92*, San Jose, CA (1992) 440-446.

[35] J. Stillman, It's not my default: the complexity of membership problems in restricted propositional default logics, in: *Proceedings AAAI-90*, Boston, MA (1990) 571–578.

[36] R.G. Valiant and V.V. Vazirani, NP is as easy as detecting unique solutions, *Theor. Comput. Sci.* **47** (1986) 85–93.

ELSEVIER

Artificial Intelligence 82 (1996) 273–302

Artificial Intelligence

On the hardness of approximate reasoning ★

Dan Roth *

Aiken Computation Laboratory, Harvard University, 33 Oxford St., Cambridge, MA 02138, USA

Received April 1994; revised October 1994

Abstract

Many AI problems, when formalized, reduce to evaluating the probability that a propositional expression is true. In this paper we show that this problem is computationally intractable even in surprisingly restricted cases and even if we settle for an approximation to this probability.

We consider various methods used in approximate reasoning such as computing degree of belief and Bayesian belief networks, as well as reasoning techniques such as constraint satisfaction and knowledge compilation, that use approximation to avoid computational difficulties, and reduce them to model-counting problems over a propositional domain.

We prove that counting satisfying assignments of propositional languages is intractable even for Horn and monotone formulae, and even when the size of clauses and number of occurrences of the variables are extremely limited. This should be contrasted with the case of deductive reasoning, where Horn theories and theories with binary clauses are distinguished by the existence of linear time satisfiability algorithms. What is even more surprising is that, as we show, even approximating the number of satisfying assignments (i.e., "approximating" approximate reasoning), is intractable for most of these restricted theories.

We also identify some restricted classes of propositional formulae for which efficient algorithms for counting satisfying assignments can be given.

1. Introduction

Investigating the computational cost of tasks that are of interest to AI has been argued [22, 39] to be essential to our understanding and our ability to characterize these

★ Supported by NSF grants CCR-89-02500 and CCR-92-00884 and by DARPA AFOSR-F4962-92-J-0466. A preliminary version of this paper appeared in the Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93), Chambery, France.

* E-mail: danr@das.harvard.edu.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0004-3702(94)00092-1

tasks and to finding knowledge representation systems adequate for them. The problem discussed most extensively in this context is the problem of propositional satisfiability, the typical NP-hard problem, which is of special concern to AI because of its direct relationship to deductive reasoning. Many other forms of reasoning, including default reasoning, planning and others which make direct appeal to satisfiability, have also been shown to be NP-hard. In practice, there is a fundamental disagreement about the implications of this. There is no debate, however, that something has to be given up: restrict the form of the statements in the knowledge base, settle for approximate output and so on. One consequence of the intensive research in that direction is the identification of restricted languages for which propositional satisfiability can be solved efficiently (e.g., Horn theories).

In this paper we consider a related problem, that of counting satisfying assignments of propositional formulae. We argue that the role played by satisfiability problems in many AI problems in which *deduction* is of special concern, is replaced by that of counting satisfying assignments when *approximate reasoning* techniques are used. To support this argument we show that various methods used in approximate reasoning, such as computing degree of belief and Bayesian belief networks, are equivalent, computationally, to solving *counting* problems. We also show that considering the problem of counting solutions is a valuable approach in evaluating techniques that use approximations in an effort to avoid computational difficulties, such as constraint satisfaction and knowledge compilation,

We analyze the computational complexity of counting satisfying assignments of propositional languages, and prove that this is intractable even for Horn and monotone formulae, and even when the size of clauses and number of occurrences of a variable in the formula are extremely limited. This should be contrasted with the case of deductive reasoning, where Horn theories and theories with binary clauses are distinguished by the existence of linear time algorithms for their satisfiability. What is more surprising is that, as we show, even approximating the number of satisfying assignments (that is, "approximating" approximate reasoning), is intractable for most of those restricted theories. We identify some restricted classes of propositional formulae for which we develop efficient algorithms for counting satisfying assignments.

While our positive results can sometimes be used to find tractable languages for the approximate reasoning technique discussed, we believe that the main contribution of this paper is the presentation of these widely applicable and surprising hardness results. This implies that research should be directed away from investigating structural constraints of the "world" and towards investigating distributional constraints, and suggests that we reconsider how we model the reasoning problem. We discuss these issues further in Section 5.

In the next section we give background material from the computational complexity of counting problems, and in Section 3 we present our positive and negative results on exact and approximate counting of satisfying assignments. The main results are presented in Section 4, where we put the model-counting results in the context of various approximate reasoning techniques, by reducing those techniques to counting problems. Proofs of the technical results appear in the appendix.

2. The computational complexity of counting problems

We give in this section a brief overview of the computational complexity of counting problems and the related problems of approximate counting and random generation of solutions. For a detailed discussion consult [10, 14, 37, 38].

With a large number of decision problems we can naturally associate a counting problem. For example, counting the number of satisfying assignments of a Boolean formula, counting the number of perfect matchings in a bipartite graph and counting the number of cycles in a graph. Clearly, the counting version is at least as hard as the decision problem but in many cases, even when the decision problem is easy, no computationally efficient method is known for counting the number of distinct solutions. The class #P was introduced by Valiant [37, 38] in an effort to explain these phenomena. A counting problem belongs to #P if there is a non-deterministic algorithm such that for any instance I of the associated decision problem, the number of "guesses" that lead to acceptance of I is equal to the number of distinct solutions to I, and such that the algorithm is polynomial in the size of I.[1] As usual, the hardest problems in the complexity class are *complete* in the class.

In particular, it was shown that counting the number of satisfying assignments of a CNF formula as well as the counting versions of many other NP-complete problems are complete for #P. More significantly, it was also shown that the counting versions of many problems in P are also complete for the same class. Examples of the latter include counting the number of satisfying assignments of a DNF formula, counting the number of cycles in a graph and many other problems [29, 37, 38].

Problems that are #P-complete are at least as hard as NP-complete problems, but probably much harder. Evidence to the hardness of problems in #P is supplied by a result of [36] which implies that one call to a #P oracle suffices to solve any problem in the polynomial hierarchy in deterministic polynomial time. This may serve also as indication that #P is outside of the polynomial hierarchy. It is therefore natural to consider the problem of approximate counting. The notion of approximation we use is that of *relative approximation* [14, 15, 35]. Let M, M' be two positive numbers and $\delta \geqslant 0$. We say that M' *approximates* M within δ when

$$M'/(1+\delta) \leqslant M \leqslant M'(1+\delta).$$

Indeed, approximating a solution to a #P problem might be easier than finding an exact solution. In fact, it is no harder than solving NP-hard problems [35]. For example, there exists a polynomial time randomized algorithm that approximates the number of satisfying assignments of a DNF formula within any constant ratio [14, 15]. It is possible, though, for a #P-complete problem, even if its underlying decision problem is easy, to resist even an efficient *approximate solution*. An example for that was given in [14], and in this paper we exhibit a similar phenomenon. We prove, for various propositional languages for which solving satisfiability is easy, that it is NP-hard to approximate the number of satisfying assignments even in a very weak sense.

[1] In [37] the definition is given in terms of "counting Turing machines".

We use the notion of relative approximation to discuss probabilities as well. It is worth noticing therefore that this notion of approximation is preserved when computing ratios of quantities. Assume we can approximate M_1 and M_2 to within δ. That is, we can find M_1', M_2' such that $M_1'/(1+\delta) \leqslant M_1 \leqslant (1+\delta)M_1'$ and $M_2'/(1+\delta) \leqslant M_2 \leqslant (1+\delta)M_2'$. Then,

$$\frac{M_1'}{M_2'}\frac{1}{(1+\delta)^2} \leqslant \frac{M_1}{M_2} \leqslant (1+\delta)^2\frac{M_1'}{M_2'}.$$

Thus, this yields a relative approximation of the ratio M_1/M_2 as well (within $2\delta+\delta^2$). In particular, when computing the conditional probability $P(Y=y \mid X=x)$, of the event $Y=y$ given evidence $X=x$, since

$$P(Y=y \mid X=x) = \frac{P(Y=y, X=x)}{P(X=x)}$$

we conclude that:

Proposition 2.1. *The complexity of computing relative approximation of the conditional probability $P(Y=y \mid X=x)$ is polynomially related*[2] *to that of computing relative approximation of the unconditional probability $P(Y=y)$.*

We note that a related class of problems of interest to AI, that of randomly generating solutions from a uniform distribution, was shown in [14] to be equivalent to randomized approximate counting, for a wide class of problems. (All natural problems considered here, e.g., finding satisfying assignments of Boolean formulae and various graph problems are in this class.) It is therefore enough, from the computational complexity point of view, to consider the problems of exact and approximate counting, as we do here.

3. Summary of model-counting results

In this section we summarize our results on exact and approximate counting of satisfying assignments of propositional languages. Those include hardness results for exact and approximate counting and positive results for exact counting. Complete proofs of the results are given in the appendix.

Let $\#(SAT, \mathcal{L})$ ($\tilde{\#}(SAT, \mathcal{L})$) denote the problem of counting (approximating, respectively) the number of satisfying assignments of a given formula from the propositional language $\mathcal{L}$. Given the problem $\#(SAT, \mathcal{L})$, a problem hierarchy is obtained whenever we place additional restrictions or relaxations on the allowed instances. Given propositional languages $\mathcal{L}_1$ and $\mathcal{L}_2$, define $\mathcal{L}_1 \subseteq \mathcal{L}_2$ if every instance of $\mathcal{L}_1$ is also an instance of $\mathcal{L}_2$. (e.g., HORN $\subseteq$ CNF.) Clearly, if we can solve the problem $\#(SAT, \mathcal{L}_2)$ we can solve the problem $\#(SAT, \mathcal{L}_1)$; to prove hardness results it is therefore enough to

[2] By that we mean that a procedure that relatively approximates unconditional probabilities and is, say, polynomial in δ, yields a procedure that can relatively approximate conditional probabilities, and is also polynomial in δ (and of course, vice versa).

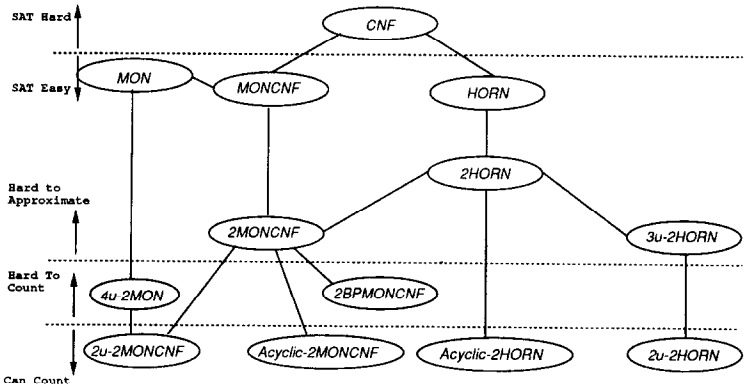


Fig. 1. Complexity of (approximately) counting satisfying assignments.

consider the most restricted languages. The same argument holds for the corresponding approximation problem.

We use the following notations and conventions in denoting propositional languages: if LANG is a class of Boolean formulae and k, l are integers, then kLANG denotes the subclass of formulae in LANG in which a clause consists of up to k literals; $l\mu$-LANG denotes the class of all LANG formulae in which no variable occurs more than l times. l is the *degree* of the formulae. For example, 2MONCNF consists of monotone CNF formulae with clauses of length 2; 3μ-2HORN consists of Horn formulae with clauses of length 2 in which no variable occurs more than 3 times.

- SAT: Any Boolean formulae.
- MON: Boolean formulae in which all variables are unnegated (monotone formulae).
- CNF: Boolean formulae in Conjunctive Normal Form.
- MONCNF: Monotone CNF.
- HORN: A CNF in which clauses have up to one unnegated variable (Horn clauses).
- 2BPMONCNF: A 2MONCNF in which the set of variables can be divided into two sets, and every clause contains one variable from each.
- Acyclic-2MONCNF: Given a 2MONCNF theory ϕ, let G be an undirected graph containing a vertex for every variable in ϕ and an edge connecting two vertices if and only if the corresponding variables appear in the same clause. ϕ is Acyclic-2MONCNF if this corresponding graph is acyclic.
- Acyclic-2HORN: Given a 2HORN theory ϕ, let G be a directed graph containing a vertex for every literal in ϕ and an edge from every vertex corresponding to a literal in the body of a rule (i.e., negative variable in the clause representation of the rule) to the vertex corresponding to a literal in the head of a rule (i.e., positive variable in the clause representation of the rule). Two special nodes T and F are added for clauses with empty body or empty head. ϕ is Acyclic-2HORN if every connected component of this corresponding graph is a directed tree.

Fig. 1 summarizes our results; it presents a hierarchy of propositional languages along with a classification according to the complexity of $\#(SAT, \mathcal{L})$ and $\tilde{\#}(SAT, \mathcal{L})$.

Based on the above comment these results imply similar results on other, less restricted languages.[3]

It is noticeable that for various propositional languages having efficient algorithms for satisfiability, and even for very restricted versions of these (e.g., 3μ-2HORN), exact counting is complete for #P. In fact, for the case of Horn theories, the situation is fully understood, and we give an efficient algorithm for the only possible case, 2μ-2HORN. The situation for approximate counting is even more surprising: for very restricted classes of Horn theories (e.g., 3μ-2HORN) it is NP-hard to approximate the number of satisfying assignments even within, say, $2^{n^{9/10}}$ (for formulae over n variables). Similar results hold for 2MONCNF theories, for which the bounded degree case is open. Our positive results virtually complete the complexity picture and can be directly applied in some of the reasoning techniques considered.

3.1. Statements of results

We now formally state the technical results outlined above. We state the results only for some of the important languages; results for other languages can be easily deduced by inclusion, as described above. Proofs are given in the appendix.

Theorem 3.1 (Hardness of Exact Counting). *Let $\Sigma \in \mathcal{L}$ be a propositional formula on n variables. If $\mathcal{L}$ is one of the following propositional languages, counting the number of satisfying assignments of Σ is complete for* #P:

(1) $\mathcal{L}$ = 2MONCNF [38],
(2) $\mathcal{L}$ = 2BPMONCNF [29],
(3) $\mathcal{L}$ = 2HORN,
(4) $\mathcal{L}$ = 3μ-2HORN,
(5) $\mathcal{L}$ = 4μ-2MON.

Theorem 3.2 (Hardness of Approximation). *Let $\Sigma \in \mathcal{L}$ be a propositional formula on n variables, and let $\varepsilon > 0$ be any constant. If $\mathcal{L}$ is one of the following propositional languages, approximating the number of satisfying assignments of Σ to within $2^{n^{1-\varepsilon}}$ is NP-hard*:

(1) $\mathcal{L}$ = 2MONCNF,
(2) $\mathcal{L}$ = 3μ-2HORN.

Theorem 3.3 (Positive results). *Let $\Sigma \in \mathcal{L}$ be a propositional formula on n variables. If $\mathcal{L}$ is one of the following propositional languages, then there exists an efficient algorithm for counting the number of satisfying assignments of Σ.*

(1) $\mathcal{L}$ = 2μ-2MONCNF,
(2) $\mathcal{L}$ = 2μ-2HORN,

[3] Notice that we place the language 2HORN above 2MONCNF even though 2HORN does not contain 2MONCNF. 2HORN contains, however, 2anti-MONCNF (where all the variables in each formula are negated rather than unnegated) and thus, clearly, all the counting results that hold for 2MONCNF hold also for 2anti-MONCNF.

(3) $\mathcal{L}$ = Acyclic-2MONCNF,
(4) $\mathcal{L}$ = Acyclic-2HORN.

4. Reducing approximate reasoning to counting

In this section we consider various techniques for approximate reasoning and show that in each case inference is equivalent to solving a counting problem. We start by considering the case of computing degree of belief, the underpinning of approximate reasoning. We then consider Bayesian belief networks, reasoning with approximate theories and constraint satisfaction problems. Finally, we discuss some previous work that relates approximate reasoning to counting problems, for which our results here also apply.

4.1. Degree of belief

The inference of a degree of belief is a generalization of deductive inference which can be used when the knowledge base is augmented by, e.g., statistical information, or in an effort to avoid the computationally hard task of deductive inference.

Consider a knowledge base consisting of a propositional theory Σ and assume we would like to assign a *degree of belief* to a particular statement α. This situation is natural in various AI problems such as planning, expert systems and others, where the actions an agent takes may depend crucially on this degree of belief. In [24] it is suggested that the kind of reasoning used in expert systems is the following: "we are given a knowledge base of facts (possibly, with their associated probabilities); we want to compute the probability of some sentence of interest. ... According to *probabilistic logic*, the probability of a sentence is the sum of the probabilities of the sets of possible worlds in which that sentence is true ...".

Indeed, the general approach to computing degree of belief is that of assigning equal degree of belief to all basic "situations" consistent with the knowledge base, and computing the fraction of those which are consistent with the query. Much work has been done on how to apply this principle, and how to determine what are the basic situations [1,2,4].

We consider here the question of *computing* the degree of belief in a restricted case, in which the knowledge base consists of a propositional theory and contains no statistical information. The hardness results we get in this restricted case just highlight the computational difficulties in the more general cases.[4] Using the above approach, all possible models of the theory are given equal weight and we are interested in the computational complexity of computing the degree of belief of a propositional formula, that is, the fraction of models that are consistent with a propositional query.

[4] The first-order version of this problem was considered in [12] where it was shown that almost all problems one might want to ask are highly undecidable. In some cases, though, it was shown that the asymptotic conditional probabilities exist, and can be computed.

Given a propositional theory Σ on n variables, the *probability that Σ is satisfied*, P_Σ, is computed over the uniform distribution over the set $\{0,1\}^n$ of all possible assignments of Σ.

$$P_\Sigma = Prob\{\Sigma \equiv T\} = |\mathcal{M}(\Sigma)|/2^n,$$

where $\mathcal{M}(\Sigma)$ denotes the set of all satisfying assignments of Σ, and $|\mathcal{M}(\Sigma)|$ denotes its size, and T stands for the truth value.

Given a propositional theory Σ and a propositional statement α, the *degree of belief in α*, is the *conditional probability* of α with respect to Σ, $P_{\alpha|\Sigma}$, that is, the fraction of satisfying assignments of Σ that satisfy α:

$$P_{\alpha|\Sigma} = Prob\{\alpha \wedge \Sigma \equiv T \mid \Sigma \equiv T\} = \frac{|\mathcal{M}(\alpha \wedge \Sigma)|}{|\mathcal{M}(\Sigma)|}.$$

The observation that $|\mathcal{M}(\alpha)| = P_{\alpha|p \vee \bar{p}}$ for any variable p, together with the discussion in Section 2 (Proposition 2.1) implies:

Proposition 4.1. *The complexity of computing (approximating) the degree of belief in a propositional statement with respect to a propositional theory, is polynomially related to the complexity of computing (approximating) the number of models of a propositional statement.*

Together with the results in Theorem 3.1 and Theorem 3.2 we have:

Theorem 4.2. *The problem of computing the degree of belief in a propositional statement (over n variables) with respect to a propositional theory, is complete for* #P.[5] *For every fixed $\varepsilon > 0$, approximating this probability within $2^{n^{1-\varepsilon}}$ is NP-hard.*

4.2. Bayesian belief networks

Bayesian belief networks provide a natural method for representing probabilistic dependencies among a set of variables and are considered an efficient and expressive language for representing knowledge in many domains [13]. We consider here the class of *multiple connected belief network*, i.e., networks that contain at least one pair of nodes (variables) that have more than one undirected path connecting them. It has been argued that the expressiveness of these networks is required for representing knowledge in several domains, like medicine. We first present briefly a general class of belief networks and the associated inference problem and then show how to reduce the inference problem to that of counting satisfying assignments of a propositional formula. For definitions and an elaborate discussion of Bayesian belief networks, the expressiveness of this representation, and the type of inference one can utilize using it, see [26].

[5] Strictly speaking the problem of computing the degree of belief is not in #P, but easily seen to be equivalent to a problem in this class. We keep the same loose interpretation in the rest of the paper.

A Bayesian belief network (causal network) consists of a graphical structure augmented by a set of probabilities. The graphical structure is a directed acyclic graph (DAG) in which nodes represent random variables (domain variables) and edges represent the existence of direct causal influence between the linked variables. A conditional probability is associated with the group of edges directed toward each node (and not with each single edge). Prior probabilities are assigned to source nodes (i.e., any node without any incoming edge). We represent the belief network as a triple (V, E, P), where V is the set of vertices (variables), E is the set of edges and P is the set of probabilities. In particular, P consists of prior probability functions, $P(X_i = x_i)$, for every source node X_i and conditional probabilities functions, $\{P(X_i \mid Y_{ij})\}_{Y_{ij} \in p_{X_i}}$, for each node X_i with a set p_{X_i} of direct predecessors. (We use the notation $P(X_i = x_i)$ and $P(x_i)$ inadvertently, when it is clear that we refer to the variable X_i.) See the construction in the proof of Theorem 4.3 for an example of a belief network. In general, not every probability distribution can be represented by a Bayesian belief network. However, given a DAG it is easy to specify consistently the conditional probabilities. One needs only to make sure that the conditional probabilities attached satisfy, for every node X_i, $\sum_{x_i} P(X_i = x_i \mid p_{X_i}) = 1$.

For complexity analysis, we take our complexity parameter to be n, the number of nodes in the belief network. Notice that the conditional probabilities tables associated with the network might be exponential in n. Practitioners of Bayesian belief networks try to avoid this case, of course. In our reduction the conditional probabilities tables will have concise representations, polynomial in the number of nodes in the network, and in this sense one can think of our complexity measure as if it is the total size of network, including the conditional probabilities tables.

The general *inference problem* using belief network is that of calculating the posterior probability $P(S_1 \mid S_2)$, where S_1 (S_2) is either a single instantiated variable or a conjunction of instantiated variables. The most restricted form of probabilistic inference, determining $P(Y = T)$ for some propositional variable Y (with no explicit conditioning information), was analyzed by Cooper [5] who proved that it is NP-hard. This hardness result for the exact inference problem shows that one cannot expect to develop general-purpose algorithms for probabilistic inference that have a polynomial running time and therefore there is a need to divert attention toward trying to construct *approximation algorithms* for probabilistic inference. Our results show that this is not the case:

Theorem 4.3. *The problem of computing the probability that a node in a Bayesian belief network is true is complete for* #P. *Moreover, for every fixed* $\varepsilon > 0$, *approximating this probability within* $2^{n^{1-\varepsilon}}$ (*where n is the size of the network*) *is NP-hard.*

Proof. The proof is based on the reduction from [5]. The two major differences are that (1) we reduce the problem of *counting* satisfying assignments of a propositional formula to that of computing the probability that a node in a Bayesian belief network is true, and (2) based on the results from Section 3 we can start our reduction from a restricted propositional formula, yielding a more restricted network topology.

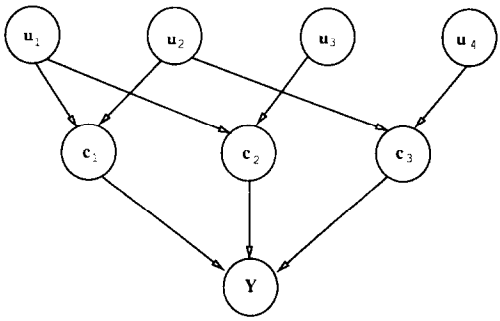


Fig. 2. The belief network generated from the 2MONCNF formula Σ.

In the following we reduce the problem of counting satisfying assignments of a 2MONCNF[6] formula to that of computing the probability that a node in a Bayesian belief network is true. Since our reduction preserves the number of satisfying assignments this reduction holds for the problem of approximating the probability as well.

Consider an instance of 2MONCNF, $\Sigma = \{c_1, c_2, \ldots, c_m\}$ where c_i are clauses on a set $U = \{u_1, u_2, \ldots, u_n\}$ of n Boolean variables. We construct a belief network $BN = (V, E, P)$ containing variable Y such that

$$2^n P(Y = T) = |\mathcal{M}(\Sigma)|.$$

To construct $BN = (V, E, P)$ we show how to define the vertices V, the edges E and the set of prior and conditional probabilities P. V is defined to be a set of $n + m + 1$ vertices, one for each variable u_i, one for every clause c_j and one for Y. The set of edges E consists of up to $3m$ edges: a variable u_i is connected to all clauses c_j in which it appears (i.e., total of up to $2m$ edges, since $\Sigma \in$ 2MONCNF); Y is connected to all clauses c_j. Fig. 2 depicts the BN generated using the above procedure for the instance of 2MONCNF in which $U = \{u_1, u_2, u_3, u_4\}$, and

$$\Sigma = \{(u_1 \vee u_2), (u_1 \vee u_3), (u_2 \vee u_4)\}.$$

The set of probabilities P is defined as follows: Each of the source nodes u_i, $1 \leqslant i \leqslant n$, is given a prior probability of $1/2$ to be T. For incoming edges to the node corresponding to the clause c_j we define the conditional probability such that the node becomes T only when it is satisfied by the assignment to the variables in the clause. Formally, if $c_j = u_{j1} \vee u_{j2}$ $(1 \leqslant j \leqslant m)$, define the conditional probabilities by:

$$P(c_j = T \mid u_j^1 = v_1, u_j^2 = v_2) = \begin{cases} 1, & \text{if the assignment } u_j^1 = v_1,\ u_j^2 = v_2 \text{ satisfies } c_j, \\ 0, & \text{otherwise.} \end{cases}$$

[6] This is not possible in [5], since the results there hinge on the hardness of solving satisfiability, which can be done in polynomial time for 2MONCNF.

Finally, the conditional probability for the edges coming into the node Y is defined by

$$P(Y = T \mid c_1, c_2, \ldots, c_m) = \begin{cases} 1, & \text{if } c_1 = T,\ c_2 = T, \ldots, c_m = T, \\ 0, & \text{otherwise.} \end{cases}$$

It is easy to see that the structure (V, E, P) defined is indeed a Bayesian belief network. Also, it is clear that the construction of a belief network from a given 2MONCNF instance can be accomplished in time that is polynomial in the size of the 2MONCNF instance.[7]

We now compute the probability $P(Y = T)$. Let $u = (u_1, \ldots, u_n)$ be an assignment of the n input variables (that is, $u \in \{0,1\}^n$), and $c = (c_1, \ldots, c_m)$ be an assignment of the m clauses (that is, $c \in \{0,1\}^m$).

By the construction above we then have that

$$P(Y = T) = \sum_{s=0}^{2^n-1} \sum_{t=0}^{2^m-1} P(Y = T \mid c = t) P(c = t \mid u = s) P(u = s). \tag{1}$$

Suppose Σ is satisfiable, and let $s_1, s_2, \ldots, s_k$ be the satisfying assignments. Clearly, for $i = 1, \ldots, k$, $P(u = s_i) = 1/2^n$. Also, by the definition of the conditional probability for $i = 1, \ldots, k$, we have that $P(c = c(s_i) \mid u = s_i) = P(Y = T \mid c = c(s_i)) = 1$, and for any other assignment, these terms are equal to 0. Thus, the internal sum in Eq. (1) is equal to $1/2^n$ when s is a satisfying assignment of Σ, and is equal to 0 otherwise. We get,

$$P(Y = T) = \frac{|\mathcal{M}(\Sigma)|}{2^n}.$$

Applying now the results in Theorem 3.1 and Theorem 3.2 completes the proof. □

We have considered the computational complexity of computing the probability of a node in a Bayesian belief network being true. To compute a conditional probability, that is, $P(Y = y \mid X = x)$, where X, Y might be sets of nodes in the network, we notice that

$$P(Y = y \mid X = x) = \frac{P(Y = y, X = x)}{P(X = x)}.$$

It is clear that exact computation of the conditional probability is as hard as computing the unconditional probability (taking, e.g., X to be a single source node). Based on Proposition 2.1 this is the case also for the problem of approximating the conditional probability, and therefore we can conclude:

Theorem 4.4. *The problem of computing the conditional probability of a node given evidence in a Bayesian belief network, is complete for* #P. *Moreover, for every fixed* $\varepsilon > 0$, *approximating this probability within* $2^{n^{1-\varepsilon}}$ (*where* n *is the size of the network*) *is NP-hard.*

[7] This relies on the fact that we can define the conditional probabilities concisely. In general, every node can be associated with a conditional probability table that is exponential in the size of the network.

Finally we note that, as in [5], this reduction can be modified to hold for restricted network topology (limited in-degree, out-degree, etc.). Further restrictions to the topologies of the network can be utilized if we reduce problems of counting satisfying assignments of syntactically restricted CNF formulae to that of computing the probability that a node in the network is true. In light of the results in Section 3, this can yield even stronger hardness results. Recently, Dagum and Luby [6] presented an even stronger result, implying the hardness of computing an *absolute* approximation of probabilities in Bayesian networks. The results here are different in that we show that the inference is equivalent to counting, and combined with the results in Section 3, it implies hardness results even for restricted network topologies.

4.3. Reasoning with approximate theories

The theory of reasoning with *approximate theories* was introduced by Selman and Kautz in a series of papers [17, 18, 32] as a new approach to developing efficient knowledge representation systems.

The goal is to speed up inference by replacing the original theory by two theories that belong to a different propositional language $\mathcal{L}$ and approximate the original theory. One approximate theory implies the original theory (a lower bound) and the other one is implied by it (an upper bound). While reasoning with the approximations instead of the original theory does not guarantee exact reasoning, it can sometimes provide a quick (but not necessarily complete, see below) answer to the inference problem. This can happen when $\mathcal{L}$ allows for efficient deduction, e.g., if $\mathcal{L}$ is the class of propositional Horn formulae.[8] Of course, computing the approximations is a hard computational problem, and this is why it is suggested as an "off-line" *compilation process*. Some computational aspects of computing theory approximations and reasoning with them are studied also in [3, 11, 21]. In the following we concentrate on discussing Horn approximation.

For notational convenience, when no confusion can arise, we identify in this section the propositional theory Σ with the set of its models (satisfying assignments). Observe that the connective "implies" ($\models$) used between Boolean functions (propositional formulae) is equivalent to the connective "subset or equal" ($\subseteq$) used for subsets of models. That is, $\Sigma_1 \models \Sigma_2$ if and only if $\Sigma_1 \subseteq \Sigma_2$.

Consider a propositional theory Σ. The Horn theories Σ_{lb}, Σ_{ub} are a Horn lower bound and Horn upper bound, respectively, of Σ, if and only if

$$\Sigma_{lb} \models \Sigma \models \Sigma_{ub}$$

or, equivalently, in subset notations,

$$\Sigma_{lb} \subseteq \Sigma \subseteq \Sigma_{ub}.$$

Σ_{glb} and Σ_{lub}, the greatest Horn lower bound and least Horn upper bound, respectively, of Σ, are called *Horn approximations* of the original theory Σ.

[8] The implication problem for Horn theories can be solved in linear time in the combined length of the theory and the query. This remains true for even a broader class of queries such as DNF formulae where each disjunct contains at most one negative literal and arbitrary CNF formulae.

In order to answer $\Sigma \models \alpha$, we use a Horn approximation based inference procedure in the following way: (1) test if $\Sigma_{lub} \models \alpha$. If the answer to (1) is "yes", then the inference procedure answers "yes", $\Sigma \models \alpha$. Otherwise, (2) test if $\Sigma_{glb} \models \alpha$. If the answer to (2) is "no", then the inference procedure answers "no", $\Sigma \not\models \alpha$. Otherwise, the inference procedure returns "don't know".

Aside from the two computational problems related to Horn approximations, namely, computing the approximations and the question of the size of the formula representing the approximations (see e.g., [3,17,18,21,32]) a third major question, that is harder to analyze, is the question of evaluating the utility of reasoning with the approximate theories. Clearly, if for a given query we have either $\Sigma_{lub} \models \alpha$ or $\Sigma_{glb} \not\models \alpha$, the answer to the question $\Sigma \models \alpha$ is correct. The total performance of the inference procedure is determined, though, by how many queries are answered "don't know", forcing the procedure to resort to the original inference problem in order to answer the query.

Consider a theory Σ, and let Σ_{lub} be its least upper bound approximation.

Proposition 4.5. *The number of queries for which the reasoning with approximate theories returns "don't know" is at least exponential in* $|\Sigma_{lub} \setminus \Sigma|$.

Proof. Let $S = \Sigma_{lub} \setminus \Sigma$. For every subset $s \subset S$ define the query $\alpha_s = \Sigma \cup s$ (that is, the set of models of α consists of the models of Σ and the models in the set s). Then, for all $s \subset S$, $\Sigma_{glb} \models \alpha_s$ (since $\Sigma_{glb} \models \Sigma$), and $\Sigma_{lub} \not\models \alpha_s$. Therefore, for all the $2^{|S|} - 1$ queries α_s, reasoning with approximate theories returns "don't know". □

In [16] it is shown that, for a family of propositional languages $\mathcal{L}$ which consists of kHorn formulae (all Horn formulae with up to k literals in a clause), one can construct examples of theories Σ for which $|\Sigma_{lub} \setminus \Sigma|$ is exponential in the number of variables, where Σ_{lub} is the least upper bound of Σ in $\mathcal{L}$. (Surprisingly, one can even construct $(k+1)$Horn theories with these properties.) Using Proposition 4.5, this leads to a double exponential number of queries for which reasoning with approximate theories returns "don't know". In [21] tools are developed that allow for a construction of such examples for every language $\mathcal{L}$ with respect to which we want to consider theory approximation. We briefly describe one example, for the case of Horn approximation:

Consider the theory

$$\Sigma = (x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge \cdots \wedge (x_{n-1} \vee x_n).$$

It is easy to see that the number of models of Σ is $3^{n/2}$. However, the least upper approximation of Σ with respect to Horn, Σ_{lub}, can be shown to contain all the models in $\{0,1\}^n$, that is, it is of size 2^n. This can be argued from the fact that the set of models of any Horn formula is closed under intersection (bitwise "and") (see, e.g., [9]). Therefore, the size of $\Sigma_{lub} \setminus \Sigma$ is exponential in the number n of variables.

This question is partially addressed in [11], where learning techniques are used to find a locally-optimal approximation. However, in [11], as is done in general in the theory of Horn approximations, an approximation is defined in terms of *containment*,

(that is, logical strength), and there is no guarantee that this approximation is "close" to the optimal one, nor that the optimal one approximates the original theory within any reasonable bound, in the sense that it answers some fraction of the queries correctly.

Taking the "counting" approach, as we suggest in this paper, can shed some light on this problem. As argued above, a more reasonable way to estimate the utility of reasoning with approximate theories is to define it in terms of proximity in the number of models, since this correlates positively with the number of queries answered correctly (i.e., not answered with "don't know") by the approximations. The examples presented indicate that, when no restriction are imposed on the queries, even syntactically similar propositional languages (e.g., $(k+1)$Horn formulae and kHorn formulae) can be far enough in terms of the number of models to produce unacceptable behavior.

The argument presented above, as exhibited in Proposition 4.5, shows that the heart of the problem is the fact that the queries presented to the reasoner are unrestricted. Thus it motivates an investigation in the direction of reasoning with restricted queries, where it might be possible to avoid these difficulties. Indeed, in [19] an experimental analysis is presented in which, under severe restrictions on the classes of queries allowed, reasoning with approximate theories is shown to succeed on a large percentage of the queries. In [21] a general analysis is developed and it is shown, in particular, that reasoning with the least upper bound of a theory Σ with respect to $\mathcal{L}$ is always correct if and only if the queries belong to the language $\mathcal{L}$.

4.4. Constraint satisfaction problems

Constraint satisfaction problems (CSP) provide a convenient way of expressing declarative knowledge, by focusing on local relationships among entities in the domain.

A *constraint satisfaction problem* [7] involves a set of n variables $x_1, \ldots, x_n$ having domains $D_1, \ldots, D_n$, where each D_i defines the set of values that variable x_i may assume. An *n-ary relation* on these variables is a subset of the Cartesian product $D_1 \times D_2 \times \cdots \times D_n$. A *constraint* between variables is a subset of the Cartesian product of their domains. The set of all n-tuples satisfying all the constraints are the solutions to the CSP. The problem is either to find all the solutions, or one solution.

In general, constraint satisfaction problems are hard, as a generalization of satisfiability of CNF formulae. A CNF formula is a constraint satisfaction problem in which $D_i = \{0, 1\}$ for each i, and each clause is the relation consisting of all the tuples for which at least one literal is 1. The set of solutions of the CSP is the set of satisfying assignments of the formula. In particular, if we consider a network of binary constraints[9] over $D_i = \{0, 1\}$, as is usually done, the problem can be represented as a 2SAT formula.

Finding all the solutions is clearly an enumeration problem, and based on the results in Section 3, even the associated counting problem is #P-complete for almost all nontrivial

[9] Not every n-ary relation can be represented by a network of binary constraints with n variables [23].

cases. [10]

However, even finding one solution of a constraint satisfaction problem is known to be hard in general, as discussed above, and different heuristics techniques have been used to find approximate solutions. To discuss those under the counting point of view presented here, we first observe that the above discussion implies that computing (approximating) the number of solutions of a (binary) CSP problem is at least as hard as computing (approximating) the number of solutions of a 2MONCNF formula.

Together with the results in Theorem 3.1 and Theorem 3.2 we have:

Theorem 4.6. *The problem of computing the number of solutions of a (binary) CSP problem is complete for* #P. *For every fixed* $\varepsilon > 0$, *approximating this number within* $2^{n^{1-\varepsilon}}$ *is NP-hard.*

Search techniques were traditionally used to solve CSPs. These techniques require, in the worst case, exponential search time, and analyzing those techniques in order to get some performance guarantees is usually hard.

We exemplify how the counting point of view taken here can be used to evaluate one class of heuristics [8] and restrict its feasibility. Dechter and Pearl suggest to use counting to guide the search according to an estimate of the confidence we have that a specific solution can be extended further to a full solution.

More specifically, it is suggested to (1) simplify the pending subproblems (i.e., make some simplifying assumptions about the continuing portion of the search graph), (2) count the number of consistent solutions in each simplified subproblem, and (3) order the candidates according to these counts and use them to decide among the pending subproblems.

The intuition behind the heuristics is that "the number of solutions found in the simplified version of the problem is a good estimate to the number of solutions in the original problem and thus is indicative of the chance to retain at least one surviving solution."

The results we present in Section 3 therefore restrict the utility of these heuristics in two ways: First, the simplified subproblem must be a tree, or another syntactically constraint structure (see Theorem 3.3), in order to be able to get a count of the solutions of the simplified version. More significantly, in case the original problem possesses a non-trivial structure, the number of solutions of the simplified version is not indicative at all to the number of solutions of the original problem. If it were, this could be used to approximate the number of solutions of the original problem, which we have shown to be hard. To summarize:

Corollary 4.7. *Using counting heuristics to constraint satisfaction problem is computationally intractable.*

[10] We comment, though, that Valiant's results ([38], Fact 7) imply that under simple conditions (e.g., when finding one solution is easy and the problem satisfies a form of self-reducibility), enumerating the solutions is polynomial in their number even when the counting problem is hard. These conditions trivially hold for Horn formulae, and therefore for subclasses of CSP as well.

On the other hand, the positive results (Theorem 3.3) can be used to identify restricted domains for which these counting techniques can be shown useful.

4.5. Related results

The most related result to the work presented here is a reduction of yet another approximate reasoning technique to a counting problem. Orponen [25] shows, by reduction, that the problem of computing Dempster's rule of combination, the main tool in the Dempster–Shafer theory of evidence is at least as hard as the problem of computing the number of satisfying assignments of a propositional CNF formula.[11] Those results can be now strengthened using Theorem 3.1 and Theorem 3.2. As immediate results we get that (1) even the approximate version of Dempster's rule of combination is hard to compute and (2) the hardness result for the Dempster–Shafer theory still holds even if we severely restrict the *basic probability assignments* allowed as evidence in the Dempster–Shafer theory.

5. Discussion

We have put results on the complexity of counting and approximating the number of satisfying assignments of propositional formulae in the context of various approximate reasoning techniques. The significance of this approach was illustrated by showing that various, supposedly different methods in approximate reasoning are equivalent to counting. It is worth noticing, for example, that while there is an active discussion in the approximate reasoning community as for differences in the semantical basis of the Dempster–Shafer theory and the Bayesian approach (see, e.g., [27,31,33]) we show here that there is one computational problem underlying both approaches: computing inference is equivalent to counting satisfying assignments of a theory. Moreover, we have shown that this approach is valuable in evaluating techniques that use approximations in an effort to avoid computational difficulties. This was exemplified by analyzing heuristics used in constraint satisfaction problems and the utility of reasoning with approximate theories. We believe that the approach developed here can be found useful in the analysis of other problems of interest to AI.

Our hardness results indicate that for the problems of computing degree of belief, probabilistic reasoning and other approximate reasoning techniques, one cannot expect to develop general-purpose algorithms that have a polynomial running time. Moreover, even computing approximate inference was proved to be intractable.

These results do not rule out the possibility of developing efficient algorithms that apply in restricted cases, as our positive results suggest; identifying more positive results and investigating how they apply to various techniques might be one direction to extend this work. We mention in particular the problem of (approximately) counting the number of satisfying assignments in bounded degree 2MONCNF formulae. The problem is left

[11] A similar result, using a different reduction, was proved independently by Provan [30]. We thank Greg Provan for bringing [25,30] to our attention.

open (see Fig. 1) and its solution might be used to develop efficient algorithms for constraint satisfaction problems, for example. The positive results presented here are important therefore not only for pointing out the tractability frontiers, but also since they provide a collection of techniques that can be used to further enhance our understanding of these problems and develop new results, possibly, for other problems of interest to AI.

However, the extent to which the hardness results apply (very restricted propositional languages, restricted topologies of Bayesian networks, etc.), imply that research should be directed away from investigating structural constraints on the "world" and towards investigating distributional constraints, or limiting our reasoning tasks rather than the "world" we reason about. The first might include constraining the distributions we can represent in our belief networks (e.g., [28]), while the second could imply studying restrictions on the type of queries we need to respond to. This is motivated also by the results in Section 4.3 that suggest that a possible approach to allow for efficient reasoning might be to constrain the *queries* (rather than the "world"). Indeed, partly motivated by these results, in [21] it is shown how constraining the queries gets around the difficulties presented in Section 4.3 and leads to correct reasoning with approximate theories.

A possible interpretation of the surprising and widely applicable results presented here is that we need to reconsider the way we model the reasoning problem. One way to get around the difficulties presented here is to allow the reasoner other ways to access the "world", instead, or in addition to the fixed (formula-based, Bayesian network-based, etc.) knowledge-based approach that we analyze here. Promising results in this direction are presented in [20].

Appendix A. Proofs

In this section we formally state and prove the results outlined in Section 3. The results are stated only for some of the important languages in Fig. 1 and results for other languages can be easily deduced by inclusion, as described in Section 3. The results in this section are summarized in three theorems: in Section A.2 (Theorem 3.1) we prove results on the hardness of exact counting. In Section A.3 (Theorem 3.2) we prove results on the hardness of approximate counting. In Section A.4 (Theorem 3.3) we give positive results on exact counting by describing efficient algorithms for counting satisfying assignments for formulae in those languages.

A.1. Preliminaries

For a Boolean function f we denote by $\mathcal{M}(f)$ the set of satisfying assignments of f and by $|\mathcal{M}(f)|$ its size. We denote $\{1, 2, \ldots, n\} = [n]$.

Lemma A.1. *The problem of counting the number of satisfying assignments of a* 2MONCNF *is equivalent to the problem of counting the number of independent sets in a graph.*

Proof. Given a graph $G(V,E)$ we associate with it a monotone CNF Φ_G on variables $\{x_1,\ldots,x_n\}$ as follows:

$$\Phi_G = \bigwedge_{(v_i,v_j)\in E} (x_i \vee x_j).$$

The inverse mapping is defined in the same manner: given a formula $\Phi \in$ 2MONCNF on $\{x_1,\ldots,x_n\}$, construct a graph $G_\Phi(V,E)$ on $\{v_1,\ldots,v_n\}$ as follows:

$$(v_i,v_j) \in E \quad \text{iff } (x_i \vee x_j) \text{ is a clause in } \Phi.$$

Now, let $S_I = \{v_j : j \in I\}$ be an independent set in G, then the assignment defined by

$$x_i = \begin{cases} 0, & \text{if } i \in I, \\ 1, & \text{otherwise,} \end{cases}$$

satisfies Φ_G. The reason is that in every clause $x_i \vee x_j$ at least one of the variables is assigned to 1, since otherwise, by the definition of Φ_G, $(v_i,v_j) \in E$, but both v_i and v_j are in the independent set S_I.

For the other direction, assume that x is a satisfying assignment of Φ, and let $I = \{i \in [n] : x_i = 0\}$, then $S_I = \{v_j : j \in I\}$ is an independent set in G_Φ. The reason is that by the definition of the graph G_Φ, no two vertices which share an edge are in I, since otherwise we have a clause in Φ that x does not satisfy. □

Corollary A.2. *The problem of counting the number of satisfying assignments of a* $k\mu$-2MONCNF *is equivalent to the problem of counting the number of independent sets in a graph of degree* k.

Proof. The mapping defined in Lemma A.1 maps graphs of degree k to $k\mu$ formulae. □

A.2. Exact counting

Theorem 3.1 (Hardness of Exact Counting). *Let $\Sigma \in \mathcal{L}$ be a propositional formula on n variables. If $\mathcal{L}$ is one of the following propositional languages, counting the number of satisfying assignments of Σ is complete for* #P:

(1) $\mathcal{L}$ = 2MONCNF,
(2) $\mathcal{L}$ = 2BPMONCNF,
(3) $\mathcal{L}$ = 2HORN,
(4) $\mathcal{L}$ = 3μ-2HORN,
(5) $\mathcal{L}$ = 4μ-2MON.

Proof. (1) and (2) are well known: (1) is proved in [38]; (2) is from [29]. We can get (3) from (2) by negating all the variables in one of the bipartite sets.

To prove (4), given a formula Φ in 2HORN we rewrite it, without changing the number of solutions, as a 3μ-2HORN formula. Let Φ be a 2HORN formula on $\{x_1,\ldots,x_n\}$ and assume x_i appears $m(i)$ times in Φ (negated or unnegated). For every $i \in [n]$

introduce $m(i)$ new variables, $\{x_i^{(j)}\}_{j=1,m(i)}$ and replace the jth appearance of x_i in Φ by $x_i^{(j)}$ to get Σ, a μ-2HORN formula. We then conjoin Σ with $\bigwedge_i \Gamma_i$, where Γ_i is the following 2μ-2HORN formula:

$$\Gamma_i = x_i^{(1)} \to x_i^{(2)} \to \cdots \to x_i^{(m-1)} \to x_i^{(m)} \to x_i^{(1)}.$$

Clearly

$$\Phi \equiv \Sigma \wedge \bigwedge_i \Gamma_i.$$

Thus, the number of satisfying assignments of the 3μ-2HORN formula $\Sigma \wedge \bigwedge_i \Gamma_i$ is equal to the number of satisfying assignments of the original 2HORN formula, and the counting problems for those languages are therefore equivalent.

For (5) we use a different rewriting technique. Given a 3μ-2CNF formula Φ we rewrite it, while preserving the number of satisfying assignments as $\Sigma \wedge \neg\Gamma$ where Σ and Γ are both monotone, and Σ and $\Sigma \wedge \Gamma$ are both in 4μ-2MON. Since $|\Sigma \wedge \neg\Gamma| = |\Sigma| - |\Sigma \wedge \Gamma|$, the hardness of exact counting for 4μ-2MON formulae results from the hardness of counting for 3μ-2CNF formulae (cf. (4)).

In rewriting Φ, given a variable x_i, which appears both negated and unnegated in Φ, we replace its (up to 2) unnegated occurrences by y_i and its (up to 2) negated occurrences by z_i. The resulting formula is a 3μ-2MONCNF formula Φ'. To force that $\forall i$, $\overline{y_i} = z_i$ we denote

$$\Phi'' = \bigwedge_i (y_i \vee z_i), \qquad \Gamma = \bigvee_i (y_i \wedge z_i).$$

It is clear that

$$\Phi \equiv \Phi' \wedge \Phi'' \wedge \neg\Gamma = \Sigma \wedge \neg\Gamma.$$

Since $\Sigma = \Phi \wedge \Phi''$ is a 3μ-2MONCNF formula and Γ is a 1μ-2MON formula (in a DNF form), the result follows. □

A.3. Approximate counting

Theorem 3.2 (Hardness of Approximation). *Let $\Sigma \in \mathcal{L}$ be a propositional formula on n variables. If $\mathcal{L}$ is one of the following propositional languages, approximating the number of satisfying assignments of Σ to within a factor of $2^{n^{1-\varepsilon}}$, for any fixed ε, is NP-hard.*

(1) $\mathcal{L}$ = 2MONCNF,
(2) $\mathcal{L} = 3\mu$-2HORN.

Proof. To get (2) from (1) we use the rewriting technique for nonmonotone clauses as in (4) of Theorem 3.1. (This technique leaves the number of solutions the same but might increase the number of variables up to n^2. This can easily be handled as we do below.) Notice that the rewriting technique used in (5) of that theorem does not extend for approximations.

The next lemma provides the main step in the proof of (1). The proof is based on the "blow-up" technique developed in [14]. A different version of this lemma appears also in [34].

Lemma A.3. *For any ε, approximating the number of independent sets of a graph on n vertices within $2^{n^{1-\varepsilon}}$ is NP-hard.*

Proof. We use the "blow-up" technique introduced in [14], to reduce the problem of approximating the number of independent sets in G to the k-INDEPENDENT-SET problem [10]. Given $G(V,E)$, where $|V| = n$, we construct a graph $G'(V',E')$ such that approximating the number independent sets in G' to within $2^{n^{1-\varepsilon}}$ can be used to solve k-INDEPENDENT-SET in G. G' is defined as follows: each vertex $v \in V$ is blown-up to a "cloud" $c(v)$ of m vertices in G'. If $(u,v) \in E$, in G' we construct a complete bipartite graph on $c(v) \cup c(u))$; otherwise, there are no edges connecting $c(v)$ to $c(u))$. Formally,

$$V' = \{v^j : v \in V;\ j \in \{1,\ldots,m\}\},$$

$$E' = \{(v^i,u^j) : (v,u) \in E;\ i,j \in \{1,\ldots,m\}\}.$$

Assume now that G contains an independent set I of size k. Then, $I' = \{v^j : v \in I;\ j \in \{1,\ldots,m\}\}$ is an independent set of size km in G'. Since all the subsets of an independent set are also independent sets, there are are least $N_{\min} = 2^{km}$ independent sets in G'.

On the other hand, if G contains no independent set of size k, an independent set in G', contains vertices from up to $k-1$ "clouds", since otherwise, the corresponding vertices in G (the "projection" of the clouds) generate in G an independent set of size larger than $k-1$. In particular, the largest independent set in G' is of size $\leqslant (k-1)m$ (there might be, however, many different independent sets of that size). Thus, there are no more than $N_{\max} = \binom{n}{k-1}2^{(k-1)m}$ independent sets in G'.

Finally, let $k = n/2$; in this case, k-INDEPENDENT-SET is NP-hard [10, p. 194]. Given $\varepsilon > 0$, choose r large enough such that $1-(r-2)/(r+1) < \varepsilon$, and let $m = n^r$. The "blow-up" graph G' is of size $|V'| = nm = n^{r+1}$. We have that,

$$\sqrt{N_{\min}/N_{\max}} = \sqrt{\frac{2^{km}}{\binom{n}{k-1}2^{(k-1)m}}} = \sqrt{\frac{2^m}{\binom{n}{k-1}}} = \sqrt{\frac{2^{n^r}}{\binom{n}{n/2-1}}}$$
$$\geqslant \sqrt{\frac{2^{n^r}}{2^{n/2}}} = 2^{(2n^r-n)/4} \geqslant 2^{n^{r-2}} = 2^{|V'|^{(r-2)/(r+1)}}.$$

Notice also that this "blow-up" procedure is polynomial in the size of the original graph. Therefore, if we can approximate the number of independent sets in G' within $2^{|V'|^{1-\varepsilon}} < 2^{|V'|^{(r-2)/(r+1)}}$ we can use this approximation to decide whether the graph G' has more than $N_{\min}$ or less than $N_{\max}$ independent sets. As argued above, this leads to deciding $n/2$-INDEPENDENT SET. $\square$

We have proved in Lemma A.1 that counting the number of satisfying assignments of a 2MONCNF is equivalent to the problem of counting the number of independent sets in a graph. This, together with Lemma A.3 implies the theorem. □

A.4. Positive results

Theorem 3.3 (Positive results). *Let $\Sigma \in \mathcal{L}$ be a propositional formula on n variables. If $\mathcal{L}$ is one of the following propositional languages, there exists an efficient algorithm for counting the number of satisfying assignments of Σ.*

(1) $\mathcal{L} = 2\mu$-2MONCNF,
(2) $\mathcal{L} = 2\mu$-2CNF,
(3) $\mathcal{L} =$ Acyclic-2MONCNF,
(4) $\mathcal{L} =$ Acyclic-2HORN.

Proof. We prove (1) by developing a closed form formula that is easy to evaluate for the number of independent sets in graphs of degree 2. For the other cases we develop efficient algorithms. (3) is the problem of counting independent sets of trees, for which we give an efficient recursive algorithm. The algorithms for (2) and (4) are more elaborate. In both cases we start by constructing chains of the form $x_1 \to x_2 \to \cdots \to x_r$, from the original theory. We then show that the original theory can be represented as a composition of these chains, and develop compositions rules that allow us to count the number of satisfying assignments of the composite chains. The difference between (2) and (4) is the type of compositions allowed. We note that (1) and (3) are subcases of (2) and (4), respectively, but we give for them separate proofs, since those cases are considerably simpler.

Proof of (1). Based on Corollary A.2 it is enough to count the number of independent sets of a degree-2 graph. In the following we consider the empty set to be an independent set.

Let G be a connected graph of maximal degree 2 on n vertices. G can be either a *cycle*, in case all its vertices are of degree 2, or an *arm* if exactly 2 of its vertices are of degree 1. We have:

Lemma A.4. *Let IS_n^c denote the number of independent sets in a cycle of length n, and IS_n^a the number of independent sets in an arm of length n. Then,*

(i) $IS_n^a = 1 + \sum_{j=1}^{\lceil n/2 \rceil} \binom{n-j+1}{j}$,
(ii) $IS_n^c = 1 + \sum_{j=1}^{\lfloor n/2 \rfloor} \binom{n-j}{j} + \binom{n-j-1}{j-1}$.

Proof. We denote by $IS_{n,j}^a$ (respectively, $IS_{n,j}^c$) the number of independent sets of size j in an arm (respectively, cycle) of length n. $IS_{n,j}^a(v)$ counts those independent sets that contain a fixed vertex, v, in the cycle.

(i) The problem of computing $IS_{n,j}^a$ reduces to the following combinatorial problem: find the number of selections of j integers from the set $\{1, \ldots, n\}$, such that no two consecutive numbers are selected. To count this number, consider any

selection of j different numbers from among $\{1,\ldots,n-j+1\}$. The mapping which adds 0 to the first number selected, 1 to the second, $\ldots, j-1$ to the jth, is a 1–1 correspondence between those selections and the legal selections we count. Thus, we get:

$$IS^a_{n,j} = \binom{n-j+1}{j}.$$

To get the total number we sum on j, $j \leqslant \lceil n/2 \rceil$, and add 1 for the empty set.

(ii) We claim that $IS^c_{n,j} = IS^a_{n-1,j} + IS^a_{n-1,j}(v)$. To see that, consider the cycle as an arm with end points v_1, v_n. (I.e, v_1, v_n are adjacent in the cycle.) The first term corresponds to the independent sets of the cycle that do not contain, say, v_n, while the second term correspond to those that contain it. (The latter group cannot contain v_1, so we get them by shifting each set that contains v_1 by one place). Also, $IS^a_{n-1,j}(v) = IS^a_{n-3,j-1}$ as we can just add two adjacent vertices as prefix, one that is not selected and v. We get:

$$IS^c_{n,j} = \binom{n-j}{j} + \binom{n-j-1}{j-1}.$$

To get the total number we sum on j, $j \leqslant \lfloor n/2 \rfloor$, and add 1 for the empty set. □

As an immediate consequence of Lemma A.4 we get:

Lemma A.5. *Let $G(V,E)$ be a graph of maximal degree 2, and assume G has r connected components, of sizes $n_1,\ldots,n_r$, respectively. The number of independent sets in G is*

$$\prod_{i=1}^{r} IS^x_{n_i},$$

where $x \in \{a,c\}$ depends on whether the component is an arm or a cycle.

Proof of (2). All the clauses in a 2μ-2CNF theory are of the form $l_1 \to l_2$ where every literal l_i might be a variable $x \in X_n$ or its negation, and every *variable* appears no more than twice in the given theory. (I.e., either a literal appears twice and its negation never appears in the theory or that the literal and its negation appear once each.) Notice that in the implication representations every clause $\overline{l_1} \vee l_2$ has two equivalent representation, $l_1 \to l_2$ and $\overline{l_2} \to \overline{l_1}$. Since it will be convenient to use the implication representation we assume that we hold both representations and use the one that it more convenient. If a literal l appears both as an antecedent and as a consequent in two clauses, e.g., $l_1 \to l_2$ and $l_2 \to l_3$ we can combine then the chain $l_1 \to l_2 \to l_3$, which now contains the only occurrence of $var(l_2)$ in the theory.

We call the theory $C = l_1 \to l_2 \to \cdots \to l_r$ a *simple chain*. In this case, l_1 is the *antecedent* of C and l_r its *consequent*. The antecedent and the consequent are the only

literals of degree 1 in the theory; all other literals are of degree 2. The antecedent and the consequent literals might have the same underlying variable, but this cannot be the case for other literals in the chain. We say that a simple chain $l_1 \to l_2 \to \cdots \to l_r$ in a 2μ-2CNF theory is maximal if it cannot be extended, that is, the theory contains no other clause with l_1 as consequent and no other clause with l_r as antecedent. Notice that two maximal simple chains C_1 and C_2 in a theory cannot have as consequents l_1 and $\overline{l_1}$, respectively. The reason in that in this case we can "reverse" C_2 and negate all its literals, to get an equivalent chain that has l_1 as antecedent. The new chain can be concatenated to C_1, contradicting its maximality. A simple chain is called a *cycle* if $var(x_1) = var(x_r)$; in this case all variables are of degree 2.

Let $C_1, C_2, \ldots, C_k$ be maximal simple chains in a 2μ-2CNF theory Σ. Assume C_1 and C_2 both have l_1 as antecedent. In this case we can *compose* the simple chains, and say that $C_1 \wedge C_2$ is a *composite chain*. Similarly, C_1 and C_2 can be composed if they have a common consequent. Notice that if a literal l appears in two maximal simple chains, it must appear in it both as an antecedent or as a consequent (being internal to both contradicts the degree requirement while being a consequent in one and an antecedent in the other contradicts the maximality of the chain). Thus, we can repeat this process of composition until there are no two chains in Σ, simple or composite, that share a *variable*.

If two chains share both antecedents and consequents, composing them results in a *closed* composite chain. Every composite chain C that is not closed has exactly two literals of degree 1, say l_i and l_j. We decide arbitrarily to denote $l_i = t(C)$, the *tail* of C and $l_j = h(C)$, the *head* of C if $i < j$. The tail and head of a composite chain can be both antecedents, both consequents or any other combination.

Given a chain C_1 on variables $\{x_1, x_2, \ldots, x_k\}$ (i.e., all k variables appear in C_1), denote by N_{C_1} the number of assignments of $\{x_1, x_2, \ldots, x_k\}$ that satisfy C_1. Likewise, for $b \in \{0, 1\}$ denote by $N_{C_1|h=b}, N_{C_1|t=b}$ the number of assignments of $\{x_1, x_2, \ldots, x_k\}$ that satisfy C_1, given that we force the head (respectively, tail) literal to 0 or 1, and by $N_{C_1|t=b_1,h=b_1}$ when we force both head and tail to some value in $\{0, 1\}$.

Notice that given the values N_C, $N_{C|t=1}$, $(N_{C|h=1})$ and $N_{C|t=1,h=1}$ one can determine all the possible values $N_{C|t=*,h=*}$. Therefore, it will be necessary to compute all these values for a composite chain. For a closed chain, it will be enough to compute N_C, since the chain will not be composed any more.

Given a 2μ-2CNF theory, to count the number of its satisfying assignments we first decompose it to simple chains and cycles. The number of satisfying assignments of these simple theories is given in Lemma A.7. As argued above, if a variable x is common to two chains it must be an antecedent in both or a consequent in both. We derive, in Lemma A.8 and Lemma A.9 a composition rule that shows how to compute the number of satisfying assignments of the conjunction of two theories, under the restriction that these two theories are part of a 2μ-2CNF theory. This composition rule is applied also for conjuncting composite chains, until there are no more compositions to be made. At this point the theory is represented as a conjunction of disjoint theories, and we use Lemma A.6 to compute its total number of satisfying assignments. We now describe the algorithm in some more details, and then prove some lemmas that show its correctness.

A.4.1. Algorithm: **Count-2μ-2HORN**

Let $\Sigma = (l_{i_1} \vee l_{j_1}) \wedge \cdots \wedge (l_{i_m} \vee l_{j_m})$ be a CNF theory in 2μ-2CNF such that $var(l_{i_k}), var(l_{j_k}) \in X_n$. The following procedure counts the number of satisfying assignments of Σ:

Construct simple chains:

- Represent each clause [12] as

$$l_1 \vee l_2 \equiv \overline{l_1} \rightarrow l_2 \equiv \overline{l_2} \rightarrow l_1.$$

- Fix an order of the clauses. Start from the first clause and greedily combine $l_i \rightarrow l_j$ and $l_j \rightarrow l_k$ to $l_i \rightarrow l_j \rightarrow l_k$. (That is, for each clause, check if one of its representations can be combined in that way, if any.) Go on until you end up with a maximal simple chain.
- Starting from the next available clause, repeat the above procedure, using only clauses that are not already part of a previously constructed chains. Go on until no more combination can be made. (I.e, no variable occurs both as consequent and as an antecedent.) Make sure in this process that the constants T and F are never internal to a chain.
- For each simple chain of the form

$$C = l_{i_1} \rightarrow l_{i_2} \rightarrow \cdots \rightarrow l_{i_r},$$

compute, using Lemma A.7 the value of $N_C, N_{C|t=1}, N_{C|h=1}, N_{C|t=1,h=1}$.

Notice that in the above process no more than n combination steps are required (since the degree of a variable in Σ is at most 2, and that the resulting chains are uniquely defined.

Combine chains:

- Given Σ, represented as a conjunction of simple chains, combine chains C_1 and C_2 if they have a common variable as antecedent, as consequent, or both.
- At each combination step, resulting in a composite chain C, compute, using Lemma A.8 and Lemma A.9, the values of $N_C, N_{C|t=1}, N_{C|h=1}, N_{C|t=1,h=1}$.
- Go on until there are no chains with common variables. No more than n combinations steps are required, since the degree of a variable in Σ is at most 2. The process results in disjoint composite chains.

Compute the number of satisfying assignments:

- Let $\{C_1, C_2, \ldots, C_k\}$ be the set of disjoint composite chains given by the previous stage. Let N_{C_i} be the number of assignments that satisfy C_i, counted only over the

[12] Clauses of length 1 can also be represented in this way, e.g., $x \equiv (T \rightarrow x)$ and $\bar{x} \equiv (x \rightarrow F)$. The fact that the constants T and F might appear in the theory more than twice will not affect the correctness of the algorithm (see remark on that later).

variables in C_i. If only r variables from X_n are used in $\{C_1, C_2, \ldots, C_k\}$ Then, using Lemma A.6 the number of satisfying assignments of Σ is

$$|\mathcal{M}(\Sigma)| = 2^{n-r} \prod_{j=i}^{k} N_{C_j}.$$

A.4.2. Correctness

It is clear that the maximal simple chains constructed by the algorithm are unique, and so are the composite chains. It is also clear that the construction is efficient. We just need to show how to derive the number of satisfying assignments in that process. We show that in the next lemmas.

Lemma A.6. *Let $\Sigma = C_1 \wedge C_2 \wedge \cdots \wedge C_k$ be a formula on X_n such that for each i, x_i appears in exactly one of the conjuncts C_j. Let N_{C_i} be the number of independent sets of C_i and N_Σ the number of independent sets of Σ. Then,*

$$N_\Sigma = \prod_{i=1}^{k} N_{C_i}.$$

Proof. Clear from the fact that C_i are variable-disjoint. □

The initial step in computing the number of satisfying assignments is given by the next lemma:

Lemma A.7. *Consider the simple chain*

$$C = l_1 \to l_2 \to \cdots \to l_r.$$

(i) *If all the underlying variables in C are different then $N_C = r + 1$, $N_{C|t=1} = 1$, $N_{C|h=1} = r$, $N_{C|t=1,h=1} = 1$.*
(ii) *If $l_1 = l_r$ then $N_C = 2$.*
(iii) *If $l_1 = \overline{l_r}$ then $N_C = 1$.*

Proof. For (i) we note that if $l_i = 1$ then $l_j = 1$ for $j > i$, so we need to consider only the first index i such that $l_i = 1$. There are r possibilities for that and one satisfying assignment in which all variables are 0. The other statements follow similarly (here $l_1 = t(C); l_r = h(C)$). In (ii), either all literals are 1 or all are 0. In (iii) we must have that: $l_1 = l_2 = \cdots = l_{r-1} = 0$. □

The next lemma shows how to compute the number of satisfying assignments when composing two chains.

Lemma A.8. *Let C_1, C_2 be two composite chains that have a exactly on variable, x, in common. We assume, without loss of generality, that the common variable is the tail of both C_1 and C_2, that the tail of the composite chain $C_1 \wedge C_2$ is the degree-1 variable coming from C_1 and the head is the degree-1 coming from C_2.*

(i) *If x appears as a positive (negative, respectively) variable in both chains it must be either an antecedent in both chains or a consequent in both. In this case,*
- $N_{C_1 \wedge C_2} = N_{C_1|t=1} N_{C_2|t=1} + N_{C_1|t=0} N_{C_2|t=0}$,
- $N_{(C_1 \wedge C_2)|t=1} = N_{C_1|t=1,h=1} N_{C_2|t=1} + N_{C_1|t=0,h=1} N_{C_2|t=0}$,
- $N_{(C_1 \wedge C_2)|h=1} = N_{C_1|t=1} N_{C_2|t=1,h=1} + N_{C_1|t=0} N_{C_2|t=0,h=1}$,
- $N_{(C_1 \wedge C_2)|t=1,h=1} = N_{C_1|t=1,h=1} N_{C_2|t=1,h=1} + N_{C_1|t=0,h=1} N_{C_2|t=0,h=1}$.

(ii) *Assume x appears as a positive variable in C_1 and as a negative variable in C_2. In this case x must be an antecedent in one of the chains (say, C_1), and consequent in the other. We have:*
- $N_{C_1 \wedge C_2} = N_{C_1|t=1} N_{C_2|t=0} + N_{C_1|t=0} N_{C_2|t=1}$,
- $N_{(C_1 \wedge C_2)|t=1} = N_{C_1|t=1,h=1} N_{C_2|t=0} + N_{C_1|t=0,h=1} N_{C_2|t=1}$,
- $N_{(C_1 \wedge C_2)|h=1} = N_{C_1|t=0} N_{C_2|t=1,h=1} + N_{C_1|t=1} N_{C_2|t=0,h=1}$,
- $N_{(C_1 \wedge C_2)|t=1,h=1} N_{C_1|t=1,h=1} N_{C_2|t=0,h=1} + N_{C_1|t=0,h=1} N_{C_2|t=1,h=1}$.

Proof. The proof is immediate from the notational assumption made and the observation that all the possible satisfying assignments are counted that way, and no other satisfying assignment is possible. □

Lemma A.9. *Let C_1, C_2 be two composite chains that have exactly two variables, x, y, in common. We assume, without loss of generality, that the literals whose variable is x are in the tail of both C_1 and C_2, and those whose variable is y are in the head of both chains. Since the result of this composition is a closed chain it is sufficient to compute $N_{C_1 \wedge C_2}$.*

(i) *If both x and y appear as a positive (negative, respectively) variables in both chains (i.e., each must be either an antecedent in both chains or a consequent in both) we have:*

$$N_{C_1 \wedge C_2} = \sum_{b_1, b_2 \in \{0,1\}} N_{C_1|t=b_1,h=b_2} N_{C_2|t=b_1,h=b_2}.$$

(ii) *If both occurrences of x are positive and y appears once as a positive variable and once negated we have:*

$$N_{C_1 \wedge C_2} = \sum_{b_1, b_2 \in \{0,1\}} N_{C_1|t=b_1,h=b_2} N_{C_2|t=b_1,h=\overline{b_2}}.$$

(iii) *If both x and y appear as positive variable in one chain and negated in the other, then we have:*

$$N_{C_1 \wedge C_2} = \sum_{b_1, b_2 \in \{0,1\}} N_{C_1|t=b_1,h=b_2} N_{C_2|t=\overline{b_1},h=\overline{b_2}}.$$

As an example, consider the case of composing two simple maximal chains

$$C_1 \wedge C_2 = (x_1 \to x_2 \to \cdots \to x_{r_1-1} \to z) \wedge (y_1 \to y_2 \to \cdots \to y_{r_2-1} \to z),$$

where the x_i's are different from the y_j's, z is the tail variable in both chains, and x_1, y_1 are the tail and the head, respectively of the composite chain. It is easy to see that

$$N_{C_1 \wedge C_2} = r_1 r_2 + 1,$$
$$N_{C_1 \wedge C_2 | t=1} = N_{C_1 \wedge C_2 | x_1=1} = r_2,$$
$$N_{C_1 \wedge C_2 | h=1} = N_{C_1 \wedge C_2 | y_1=1} = r_1,$$
$$N_{C_1 \wedge C_2 | t=1, h=1} = 1.$$

If also $x_1 = y_1$ then $N_{C_1 \wedge C_2} = (r_1 - 1)(r_2 - 1) + 2$.

Proof. The proof is immediate from the notational assumption made and the observation that all the possible satisfying assignments are counted that way, and no other satisfying assignment is possible. □

With the observations that the computation above can be done in time polynomial in the size of the formula, this completes the correctness proof for algorithm **Count-2μ-2CNF**. We note that in the case of clauses that contain a constant, T or F, since all of them appear either as tail or head of a chain the algorithm can handle multiple occurrences of them. This in fact is true in general. The composition rule holds if we require only that the degree of variables that appear internal to a simple chain is at most 2, while the degree of all other variables, those whose all occurrences in the theory iseither positive or negative, is not restricted. The problem is that in this case every composite chain can have more than two possible connection points, and the number of N_*'s we need to keep track of, in order to implement the algorithm, grows exponentially. Therefore, we can allow no more than a logarithmic (in n) number of variables with unrestricted degree.

Proof of (3). By Lemma A.1, given $\Sigma \in$ Acyclic-2MONCNF, it is sufficient to count the number of independent sets in the corresponding graph which is, by the definition of an Acyclic-2MONCNF formula (Section 3), an acyclic graph. We first consider the case of a connected acyclic graph, a tree.

Lemma A.10. *Let T be a tree on n vertices. The number of independent sets of T can be computed in time* $O(n)$.

Proof. Let T be a tree with root r. For a vertex $x \in T$, we denote by T_x the subtree of T with x as root. $c(x)$ denotes the set of all vertices which are children of x in T, and $gc(x)$ the grandchildren of x in T. We denote by IS_x the number[13] of independent sets of the tree rooted at x. Among these independent sets, $IS_x(x)$ denotes the number of those which contain the root x, and $IS_x(\not{x})$ denotes the number of those which do not contain the root x.

Notice that, for all $y \in c(x)$ and any independent set I of T_x, $I_y = \{z \in I \cap T_y\}$ is an independent set in T_y. We use this in the next claim to represent the number of independent sets in T_x in terms of the number of independent sets of subtrees of T_x.

[13] As before, we count the empty set as one of the independent sets.

Count-IS-Tree(T):

For all $x \in R_0$, $IS_x = 2$.

For all $x \in R_1$, $IS_x = 1 + 2^{|c(x)|}$.

For $i = 3, 4, \ldots, r$ do:

For all $x \in R_i$, $IS_x = \prod_{z \in gc(x)} IS_z + \prod_{y \in c(x)} IS_y$.

End

Fig. A.1. Counting independent sets of a tree.

Claim A.11. *For $IS_x, IS_x(x), IS_x(\not{x})$ as defined above, we have:*

(i) $IS_x(\not{x}) = \prod_{y \in c(x)} IS_y$,

(ii) $IS_x(x) = \prod_{z \in gc(x)} IS_z$,

(iii) $IS_x = IS_x(x) + IS_x(\not{x}) = \prod_{z \in gc(x)} IS_z + \prod_{y \in c(x)} IS_y$.

Proof. If $c(x) = \{y_1, y_2, \cdots, y_k\}$ and $\{I_{y_i} \subseteq T_{y_i}\}_{i=1}^k$ any collection of independent sets (in the respective trees) then $\bigcup_{i=1}^k I_{z_i}$ is an independent set in T_x that does not contain x. For the other direction, clearly any independent set $I \subseteq T_x$ that does not contain x can be decomposed uniquely as above. For (ii), similarly, if I is an independent set in T_x and $x \in I$, then clearly $\forall z \in gc(x)$, $I_z = \{z \in I \cap T_z\}$ is an independent set in T_z. For the other direction, if $gc(x) = \{z_1, z_2, \ldots, z_k\}$ and $\{I_{z_i} \subseteq T_{z_i}\}_{i=1}^k$ any collection of independent sets in T_{z_i}, then clearly $\bigcup_{i=1}^k I_{z_i} \cup \{x\}$ is an independent set in T_x, since it contains no vertex from $c(x)$. (iii) is immediate from (i) and (ii). □

We now present an algorithm, **Count-IS-Tree**, that computes the number of independent sets of a given tree. We denote by $r(x)$ the *rank* of the vertex $x \in T$. The rank of x is defined as follows: If x is a leaf, $r(x) = 0$. If x is an internal node in the tree we define, $r(x) = 1 + \max_{y \in c(x)} r(y)$. We denote by R_i the set of all vertices $x \in T$ such that $r(x) = i$, and assume w.l.o.g. that the tree T is representated as a collection of its sets R_i. In the algorithm, we compute the number IS_x of independent sets of a tree rooted at $x \in R_i$, given the values IS_y for all $y \in R_j$, for $j < i$.

For the boundary conditions, notice that if the tree contains a single vertex x, then $IS_x = 2$, and if the children of the root are leaves, then $IS_x = 1 + 2^{\#\text{leaves}}$. Thus, the correctness of the algorithm follows from Claim A.11 and the discussion above, and this completes the proof of the lemma. □

Since an acyclic undirected graph is a union of disjoint trees, using Lemma A.6 completes the proof.

Proof of (4). As in the proof of (3) we assume that in the graph that corresponds to the 2HORN formula Σ every connected component is a *tree*. The counting algorithm is very similar to the one presented for the acyclic monotone case. We prove a claim that is analog to Claim A.11, and use it to count the satisfying assignments as in the algorithm **Count-IS-Tree(T)** above.

Claim A.12. *Let N_x denote the number of assignments that satisfy the conjunction of clauses that correspond to a tree rooted at x (with respect to these variables only). $N_x(0)$ (respectively, $N_x(1)$) denotes those assignments in which x is assigned 0 (respectively, 1). We have that:*

(i) $N_x(0) = \prod_{y \in c(x)} N_y$,
(ii) $N_x(1) = \prod_{y \in c(x)} N_y(1)$,
(iii) $N_x = N_x(0) + N_x(1) = \prod_{y \in c(x)} N_y + \prod_{y \in c(x)} N_y(1)$.

Proof. To prove (i) we observe that since x is assigned 0, there are no restrictions on the satisfying assignments of the tree rooted at $y \in c(x)$. Since the subtrees rooted at different elements of $c(x)$ are disjoint, we get the result. We get (ii) by observing that an assignment satisfies the formula corresponding to T_x, where x is assigned 1, iff all $y \in c(x)$ are assigned 1. (iii) is immediate from (i) and (ii). □

Noticing that if the corresponding tree is of depth 1, the number of satisfying assignments is $1 + 2^{\#\text{leaves}}$, serves as the boundary condition for the procedure, that uses Claim A.12 to count the number of satisfying assignments of the Acyclic-2HORN formula Σ. Similar to the algorithm **Count-IS-Tree(T)** in the proof of (3) above we get an algorithm that computes the number of satisfying assignments efficiently. □

Acknowledgements

I am very grateful to Les Valiant for very helpful discussions and for his comments on an earlier draft of this paper. I would also like to thank Karen Daniels, Roni Khardon and Salil Vadhan for their comments on an earlier draft of this paper.

References

[1] F. Bacchus, *Representing and Reasoning with Probabilistic Knowledge: A Logical Approach to Probabilities* (MIT Press, Cambridge, MA, 1990).
[2] F. Bacchus, A. Grove, J.Y. Halpern, and D. Kolle, From statistics to beliefs, in: *Proceedings AAAI-92*, San Jose, CA (1992) 602–608.
[3] M. Cadoli, Semantical and computational aspects of Horn approximations, in: *Proceedings IJCAI-93*, Chambery, France (1993) 39–44.
[4] R. Carnap, *Logical Foundations of Probability* (University of Chicago Press, Chicago, IL, 1950).
[5] G.F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks, *Artif. Intell.* **42** (1990) 393–405.
[6] P. Dagum and M. Luby, Approximating probabilistic inference in Bayesian belief networks is NP-hard, *Artif. Intell.* **60** (1993) 141–153.
[7] R. Dechter, Constraint networks, in: G.S. Shapiro, ed., *Encyclopedia of Artificial Intelligence* (Wiley, New York, 1992).
[8] R. Dechter and J. Pearl, Network-based heuristics for constraint-satisfaction problems, *Artif. Intell.* **34** (1988) 1–38.
[9] R. Dechter and J. Pearl, Structure identification in relational data, *Artif. Intell.* **58** (1992) 237–270.
[10] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, CA, 1979).

[11] R. Greiner and D. Schuurmans, Learning useful Horn approximations, in: *Proceedings International Conference on the Principles of Knowledge Representation and Reasoning*, Cambridge, MA (1992) 383–392.
[12] A. Grove, J.Y. Halpern and D. Koller, Asymptotic conditional probabilities for first-order logic, in: *Proceedings 24th ACM Symposium of the Theory of Computing* (1992) 294–305.
[13] S. Holtzman, *Intelligent Decision Systems* (Addison-Wesley, Reading, MA, 1989).
[14] M.R. Jerrum, L.G. Valiant and V.V. Vazirani, Random generation of combinatorial structures from a uniform distribution, *Theoret. Comput. Sci.* **43** (1986) 169–188.
[15] R. Karp and M. Luby, Monte-Carlo algorithms for enumeration and reliability problems, in: *Proceedings 24th IEEE Symposium of Foundations of Computer Science* (1983) 56–64.
[16] H. Kautz, M. Kearns and B. Selman, Horn approximations of empirical data, *Artif. Intell.* **74** (1995) 129–146.
[17] H. Kautz and B. Selman, A general framework for knowledge compilation, in: *Proceedings International Workshop on Processing Declarative Knowledge*, Kaiserslautern, Germany (1991).
[18] H. Kautz and B. Selman, Forming concepts for fast inference, in: *Proceedings AAAI-92*, San Jose, CA (1992) 786–793.
[19] H. Kautz and B. Selman, An empirical evaluation of knowledge compilation by theory approximation, in: *Proceedings AAAI-94*, Seattle, WA (1994) 155–161.
[20] R. Khardon and D. Roth, Learning to reason, in: *Proceedings AAAI-94*, Seattle, WA (1994) 682–687; also: Technical Report TR-02-94, Aiken Computation Lab., Harvard University, Cambridge, MA (1994).
[21] R. Khardon and D. Roth, Reasoning with models, in: *Proceedings of the National Conference on Artificial Intelligence* (1994) 1148–1153. Submitted for publication. Full version: Technical Report TR-01-94, Aiken Computation Lab., Harvard University (1994).
[22] H. Levesque, Making believers out of computers, *Artif. Intell.* **30** (1986) 81–108.
[23] U. Montanari, Networks of constraint: fundamental properties and applications to picture processing, *Inf. Sci.* **7** (1974) 95–132.
[24] N.J. Nilsson, Probabilistic logic, *Artif. Intell.* **28** (1986) 71–87.
[25] P. Orponen, Dempster's rule of combination is #P-complete, *Artif. Intell.* **44** (1990) 245–253.
[26] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, San Mateo, CA, 1988).
[27] J. Pearl, Reasoning with belief functions: an analysis of compatibility, *Int. J. Approximate Reasoning* **4** (1990) 343–389.
[28] D. Poole, Average-case analysis of a search algorithm for estimating prior and posterior probabilities in Bayesian networks with extreme probabilities, in: *Proceedings IJCAI-93*, Chambery, France (1993) 607–612.
[29] J.S. Provan and M.O. Ball, The complexity of counting cuts and of computing the probability that a graph is connected, *SIAM J. Comput.* **12** (4) (1983) 777–788.
[30] M.G. Provan, A logical-based analysis of Dempster–Shafer theory, *Int. J. Approximate Reasoning* **4** (1990) 451–498.
[31] M.G. Provan, The validity of Dempster–Shafer belief functions, *Int. J. Approximate Reasoning* **6** (1992) 389–399.
[32] B. Selman and H. Kautz, Knowledge compilation using Horn approximations, in: *Proceedings AAAI-91*, Anaheim, CA (1991) 904–909.
[33] G. Shafer, Perspectives of the theory and practice of belief functions, *Int. J. Approximate Reasoning* **4** (1990) 323–362.
[34] A. Sinclair, Randomized algorithms for counting and generating combinatorial structures, Ph.D. Thesis, Department of Computer Science, University of Edinburgh (1988).
[35] L. Stockmeyer, On approximation algorithms for #P, *SIAM J. Comput.* **14** (1985) 849–861.
[36] S. Toda, On the computational power of PP and $\oplus$P, in: *Proceedings 30th IEEE Symposium of Foundations of Computer Science* (1989) 514–519.
[37] L.G. Valiant, The complexity of computing the permanent, *Theoret. Comput. Sci.* **8** (1979) 189–201.
[38] L.G. Valiant, The complexity of enumeration and reliability problems, *SIAM J. Comput.* **8** (1979) 410–421.
[39] L.G. Valiant, A theory of the learnable, *Commun. ACM* **27** (11) (1984) 1134–1142.

ELSEVIER

Artificial Intelligence 82 (1996) 303–330

Artificial Intelligence

Stochastic modelling of Genetic Algorithms

David Reynolds*, Jagannathan Gomatam

Department of Mathematics, Glasgow Caledonian University, Cowcaddens Road, Glasgow G4 0BA, UK

Received July 1994

Abstract

This paper presents stochastic models for two classes of Genetic Algorithms. We present important distinctions throughout between classes of Genetic Algorithms which sample with and without replacement, in terms of their search dynamics. For both classes of algorithm, we derive sufficient conditions for convergence, and analyse special cases of Genetic Algorithm optimisation. We also derive a long-run measure of crossover bias for optimisation via Genetic Algorithms, which has practical implications with respect to the choice of crossover operators. For a class of Genetic Algorithms, we provide theoretical underpinning of a class of empirically derived results, by proving that the algorithms degenerate to randomised, cost-independent search as mutation probabilities increase. For an alternative class of Genetic Algorithms, we show that degeneration accompanies excessive crossover rates. In formulating the models, important definitions are introduced which capture in simple form the probabilistic properties of the genetic operators, which provides models which are independent of solution encoding schemes.

1. Introduction

Genetic Algorithms (GAs) are a set of heuristic search algorithms which have been applied with success to a wide variety of combinatorial optimisation problems, including many that are NP-hard [3,10]. Theoretical investigations into the reasons for this success are divided mainly into two camps; those who formulate and analyse deterministic mathematical models for GAs [2, 10], and those who investigate the application of stochastic models based on the theory of Markov Chains [4, 5, 7–9, 15, 17, 18]. In this paper we shall present a number of stochastic models for GAs.

* Corresponding author. E-mail: dre@gcal.ac.uk.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0004-3702(94)00091-3

We begin by giving a brief introduction to GAs in Section 2, making important distinctions between large classes of GAs, those which create new populations of solutions from old ones by sampling with replacement, and those which sample solutions without replacement. In Section 3 we present a brief introduction to the theory of homogeneous Markov Chains [4, 5, 15, 18], and then proceed to summarise current stochastic models for GAs [7–9, 17]. The work reported in [7] was previously unknown to the authors; in this paper we generalise and extend the results reported there. In Sections 4 and 5 we introduce new definitions for describing the probabilistic properties of the GA operators and use these definitions to formulate representation-independent stochastic models for the two classes of GAs mentioned. The models are then used to investigate various properties of GAs. We analyse special cases of optimisation via GAs, and show that, given certain choices of mutation and crossover rates, both GAs reduce to a search of the solutions where no role is played by the cost function, despite the presence of a selection operator. For these cases, we show that both GAs converge in one generation. We present bounds for the expected numbers of optimal and other solutions within the population in the long run which is based on the probability of creation of solutions under crossover, for the class of GAs which operate on the basis of sampling with replacement. Finally, we discuss the potential for further analysis of the models contained in this paper with respect to obtaining the long-run probability distribution of a generalised GA [7] and for analysing their rates of convergence towards this distribution.

2. Definitions of GAs

The objective of this section is to formulate definitions for GAs, together with various assumptions and concepts which are necessary for the formulation of mathematical models to describe their behaviour. We stipulate that our optimisation problem has $k > 0$ candidate solutions, where k is an integer. We assume that we have assigned a bijective mapping from the set of solutions to the integers $\{1, \ldots, k\}$, and that to each solution i $(1 \leqslant i \leqslant k)$ there corresponds a well-defined cost (or fitness), f_i, which is restricted by $0 < f_i < \infty$. Thus, to the set of solutions $s = \{s_1, s_2, \ldots, s_k\}$ there corresponds a set of fitness values $\boldsymbol{f} = \{f_1, \ldots, f_k\}$. For the purposes of formulating and developing the models, we merely assume that we have assigned a natural index labelling system to the solutions, e.g., in a binary string encoding [10] each solution would be represented in our analysis as the base 10 version of its binary representation; thus, we do not assume any particular encoding scheme for solutions.

A GA for solving finite and discrete maximisation problems of the type mentioned is characterised by possessing the following features [3, 10]:

(i) a scheme for encoding solutions to the optimisation problem to be solved (chromosomes);

(ii) an evaluation function that rates each solution, assigning a positive cost, or fitness to each one;

(iii) an initialisation procedure for generating an initial population of size N of candidate solutions;
(iv) a set of operators used to manipulate the genetic composition of the population between generations;
(v) a set of parameter values, such as stopping_criteria, etc.

This algorithm, together with the genetic operators, has been implemented in many different ways [6]. For example, there are many different representation schemes [6] for candidate solutions to optimisation problems, the most common being binary strings of fixed length L. Also, the fitness evaluation/selection, mutation and crossover operators have all been implemented in different ways [6], including the recent introduction of a time-varying mutation operator [7]. Further, the order in which these operators are applied in order to generate new populations has also been varied [6]. However, we can divide into two classes the majority of existing GAs; those algorithms which form new populations of solutions by sampling solutions with replacement from the current population and then implementing the genetic operators, and those which sample solutions without replacement and then implement the genetic operators. A pseudo-code description for GAs which use sampling with replacement is:

```
GA/WI(N, s, f, fitness selection, mutation, crossover)
begin
  Initialise with population of size N at generation t = 0
  REPEAT
    WHILE |Pop(t + 1)| < N
    BEGIN
      Select (p ≥ 2) parent solutions with replacement from Pop(t) by
        fitness selection
      Carry out mutation of the copies of the selected parents
      Combine the mutant parents to form (c ≥ 1) child solutions by
        crossover
      Place the child solutions in the new population Pop(t + 1)
    END
    t := t + 1
  UNTIL stopping_criteria_reached
end
```

Here and in what follows, *mutation* and *crossover* are pre-defined operators.

We shall consider *fitness selection* above and in what follows to be implemented by the common roulette wheel selection [10]. Further, for GA/WI, we shall consider the selected parent solutions to be replaced immediately after selection into the old population (i.e., into $Pop(t)$), and that we subsequently operate on copies of the parents. For GAs which operate on the basis of sampling without replacement, a pseudo-code description is:

```
GA/WO(N s,f, fitness selection, mutation, crossover)
begin
```

```
    Initialise with population of size N, N even, at generation t = 0
    REPEAT
      Select N parent solutions with replacement from Pop(t) by fitness
        selection, to form population Pop'(t)
      Carry out mutation to each solution in Pop'(t) to form population
        Pop''(t)
      Carry out crossover pairwise between solutions in Pop''(t) to form N new
        child solutions, to form population Pop'''(t)
      Set Pop(t + 1) := Pop'''(t)
      t = t + 1
    UNTIL stopping_criteria_reached
  end
```

It is important to note that both algorithms contain a sampling with replacement *fitness selection* operator, although in GA/WO the *mutation* and *crossover* operators are implemented on the basis of sampling from the current population without replacement. According to [9], GAs which contain implementations of the latter two operators without replacement are more common in GA applications. Note that we have not described the above algorithms with respect to the choice of solution representation scheme, of which there are many [6]. The models we formulate will be independent of representation.

3. Current stochastic models

We present a brief introduction to Markov Chains. We then proceed to discuss the current approaches to the stochastic modelling of GAs. The reader is referred to [5,15,18] for further details.

3.1. Homogeneous Markov Chain theory

Definition 3.1. A stochastic process (i.e., a sequence of random variables over time, $X(t)$) is called a Markov process if it exhibits Markov dependence, i.e., if

$$\Pr[X(t) \leq x \mid X(t_n) = x_n, \ldots, X(t_0) = x_0] \tag{3.1}$$

$$= \Pr[X(t) \leq x \mid X(t_n) = x_n] \tag{3.2}$$

$$= F(x_n, x; t_n)\,, \tag{3.3}$$

where $t > t_n > \cdots > t_0$. A realisation of $X(t)$ over time is called a Markov Chain.

The Markov Chains we shall discuss in this paper are realisations of random variables defined over discrete state-spaces and discrete time. If a Markov process has parameters which do not vary over time, the process is said to be homogeneous, otherwise, inhomogeneous. In the homogeneous case, the probabilities of

transiting from state i to state j between time steps $t_0 = m$ and $t_1 = n$ depend only on $t_1 - t_0$ and are defined to be

$$P_{ij}^{(m,n)} = \Pr[X_n = j \mid X_m = i] . \tag{3.4}$$

A Markov Chain is said to be irreducible if the corresponding one time-step transition probability matrix is irreducible. If the chain has all of its states i, $1 \leq i \leq S$, s.t. $P_{ii} > 0$, then the chain is said to be aperiodic. For irreducible and aperiodic finite length Markov Chains, the following two theorems hold:

Theorem 3.2 [15, p. 47]. *Let P be the transition probability matrix of an irreducible, aperiodic finite Markov Chain. Then there exists an N s.t. for all $n \geq N$, the n-step transition probability matrix P^n has no zero elements.*

Theorem 3.3 [15, p. 88]. *Let P be the transition probability matrix of an irreducible, S-state finite Markov Chain. Then*

$$\lim_{t \to \infty} P^t = \begin{bmatrix} \boldsymbol{\pi} \\ \boldsymbol{\pi} \\ \vdots \\ \boldsymbol{\pi} \end{bmatrix} , \tag{3.5}$$

where $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_S]$ with $0 < \pi_j < 1$ and $\boldsymbol{\pi} \cdot [1, \ldots, 1]_{1 \times S} = 1$.

The following theorem shows that irreducible and aperiodic Markov Chains converge towards stationarity at a geometric rate (i.e., possess geometric ergodicity, [15]).

Theorem 3.4 [15, p. 90]. *Let P be the transition matrix of an irreducible, aperiodic S-state Markov Chain. Let the limiting probabilities be defined as in* (3.5). *Then there exists constants c and r with $c > 0$ and $0 < r < 1$ s.t.*

$$P_{ij} = \pi_j + e_{ij}^{(n)} , \tag{3.6}$$

where

$$|e_{ij}^{(n)}| \leq c r^n . \tag{3.7}$$

When $P > 0$, $c = 1$.

Finally, the following theorem presents the stationary distribution of an irreducible and aperiodic Markov Chain in terms of the corresponding transition probability matrix:

Theorem 3.5 [15, p. 92]. *Given the transition probability matrix P of an aperiodic and irreducible, S-state finite Markov Chain, there exists a unique probability vector $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_S]$ with $\boldsymbol{\pi} \cdot [1, \ldots, 1]_{1 \times S} = 1$, and*

$$\Pi P = \Pi = P \Pi , \tag{3.8}$$

where Π is a matrix where each row vector is identical to $\boldsymbol{\pi}$. The probability vector $\boldsymbol{\pi}$ gives the stationary distribution of the process.

Notice that (3.8) implies $\boldsymbol{\pi} P = \boldsymbol{\pi}$. In the rest of the paper, we shall refer to (3.8) as the invariant equation. We proceed in the next subsection to discuss current approaches to the stochastic modelling of Genetic Algorithms.

3.2. General stochastic models for GAs

Most current models formulated to describe the probabilistic search carried out by GAs are mainly based on quite general matrix properties, and the properties of the binary string encoding scheme. Thus, the genetic operators are cast as irreducible and primitive stochastic matrices, and it can be shown that, for example, due to the properties of these matrices a stationary distribution holds for certain classes of GA [7–9, 17]. However, since these models generally consider the action of the mentioned operators upon the states of the algorithm, i.e., the populations, it is quite difficult to obtain results regarding the actual solutions to the optimisation problem [17]. Thus, the transition probability matrices of many current models are based in the large part upon the irreducibility and primitivity properties of certain genetic operators, and this means that the models are not formulated on the basis of costs of solutions, the probability that a particular solution is created upon *mutation*, *crossover*, and other algorithm parameters. Thus, little information can be extracted with respect to the search dynamics of the corresponding algorithms. For example, the main stochastic model of [17] is

$$P = C \cdot M \cdot S \tag{3.9}$$

where P is the transition probability matrix of the underlying stochastic process of the Genetic Algorithm, and C, M and S are irreducible matrices which correspond to the three genetic operators acting upon the population states of the algorithm. The stochastic model contained in [8] is essentially an abstract generalisation, which also appears to be over-simplified [17], as is the main model of [9]. Note that in [8, 17] a globally optimal GA is given as one which maintains the currently optimal solution from generation to generation (i.e., elitism [10]). In [17] the conclusion is reached that the global optimality of this algorithm depends essentially on this "algorithmic trick". We verify this conclusion explicitly in this paper by carrying out the analyses of special cases mentioned previously.

3.3. Explicit stochastic models for GAs

In [7], a description is given of the application and analysis of modelling a binary encoded GA as a sequence of random vectors [4,11]. Thus, using these models, it is shown in [7] that a stationary distribution exists for certain classes of GAs, at least for those which implement the genetic operators on the basis of sampling with replacement, i.e., instances of GA/WI under a binary encoding

scheme [10]. The stationary distribution of the GAs described in [7] is given in terms of the characteristic equations of modified transition probability matrices. However, the main work of [7] is concerned with the analysis of time-varying mutation probabilities within Genetic Algorithms, and does not generalise to both classes of Genetic Algorithm under discussion here. In this paper we present and formalise the distinction between algorithms GA/WI and GA/WO by formulating stochastic models for each algorithm which show explicitly their differing search properties; these differing properties have not yet been discussed in the literature. The models are then used to present explicitly the stationary distribution of both of these GAs in certain cases, and to provide practical guidance for the parameterisation of such algorithms.

4. Stochastic model for GA/WI

The objective of this section is to formulate a stochastic model for the first class of algorithms, GA/WI, i.e., those which create new populations of solutions purely on the basis of sampling with replacement from the current population and then operating on copies of the sampled solutions with the mentioned genetic operators.

4.1. *K-solution, 1-operator* GA/WI(N, s)

We present a stochastic model for GA/WI(N, s), i.e., with no fitness selection, *mutation* or *crossover* operators. We begin by formulating essential definitions.

Definition 4.1. Let $(X_i)_t$ represent the numbers of solution i within the population GA/WI at time t. Then $(X_1, X_2, \ldots, X_k)_t$ represents the numbers of all solutions in the population at time t, where $X_1 + X_2 + \cdots + X_k = N$ and X_i is a positive integer.

Definition 4.2. Let

$$S^{[k]} = \{X = (X_1, X_2, \ldots, X_k)\colon\ X_1 + X_2 + \cdots + X_k = N,\ X_i \text{ a positive integer}\}$$

represent the set consisting of all valid states of algorithm GA/WI.

Thus, since GA/WI samples solutions with immediate replacement and then implements the genetic operators on copies of the sampled solutions, no distinction is made between the permutations of solutions within the population states of $S^{[k]}$. Since the numbers of each solution within the population is a random variable, the collection $(X_1, X_2, \ldots, X_k) \in S^{[k]}$ is a random vector [11], subject to $X_1 + X_2 + \cdots + X_k = N$. Therefore, the states of GA/WI are random vectors. Essentially, the analysis of the properties of GA/WI with respect to

heuristic search for optimisation reduces, in certain cases, to the analysis of the probability distributions of the given random variables/vectors.

In order to formulate transition probability matrices and corresponding convergence results for GA/WI, we make the following definition of matrix multiplication for general matrices which are indexed by vectors;

Definition 4.3. Let $A_{XY} \in \Re$ and $B_{XY} \in \Re$ be $|S^{[k]}| \times |S^{[k]}|$ matrices, where $\boldsymbol{X},\boldsymbol{Y} \in S^{[k]}$. Then we define as the $\boldsymbol{IJ}$th element of the matrix product AB the real number

$$\sum_{\boldsymbol{D} \in S^{[k]}} A_{\boldsymbol{ID}} B_{\boldsymbol{DJ}} = \sum_{i=1}^{|S^{[k]}|} A_{\boldsymbol{ID}_i} B_{\boldsymbol{D}_i \boldsymbol{J}} \quad \forall \boldsymbol{I}, \boldsymbol{J} \in S^{[k]}, \tag{4.1}$$

where by summation over $\boldsymbol{D}_i$ is meant the summation over the list of vectors formed under an arbitrary bijective mapping between $\{1, 2, \ldots, |S^{[k]}|\}$ and $S^{[k]}$. Thus $\boldsymbol{D}$ is a dummy vector in (4.1). Only the existence of the (arbitrary) mapping is important, and there are $|S^{[k]}|!$ such mappings [19]. When successive matrix multiplications are carried out for such matrices, it is assumed that the same mapping is used for indexing purposes throughout.

From this general definition, we make the following definition for premultiplication of an $|S^{[k]}| \times |S^{[k]}|$ matrix by a $1 \times |S^{[k]}|$ column vector.

Definition 4.4. Let $q_{\boldsymbol{Y}} \in \Re$ and $B_{\boldsymbol{XY}} \in \Re$ where B is an $|S^{[k]}| \times |S^{[k]}|$ matrix and $\boldsymbol{q}$ is a $1 \times |S^{[k]}|$ column vector, where $\boldsymbol{X},\boldsymbol{Y} \in S^{[k]}$. Then we define as the $\boldsymbol{J}$th element of the matrix product $\boldsymbol{q}B$ the real number

$$\sum_{\boldsymbol{D} \in S^{[k]}} q_{\boldsymbol{D}} B_{\boldsymbol{DJ}} = \sum_{i=1}^{|S^{[k]}|} q_{\boldsymbol{D}_i} B_{\boldsymbol{D}_i \boldsymbol{J}} \quad \forall \boldsymbol{J} \in S^{[k]}, \tag{4.2}$$

where by summation over $\boldsymbol{D}_i$ is meant the summation over the list of vectors formed under an arbitrary bijective mapping between $\{1, 2, \ldots, |S^{[k]}|\}$ and $S^{[k]}$, as in Definition 4.3.

This notation, which enables us to ignore the particular mapping used and retain vector indexing notation for the elements of the transition matrices which are to follow, is useful in terms of estimating marginal probability distributions from transition matrices [15], as will become clear. It is usual when formulating stochastic models for GAs to show the existence of the bijective mappings described in the previous definitions, and then use these to index the states of the GA [7–9]. However, in this paper we do not require this restriction, preferring to retain the identity of solutions within the states of GA/WI; this then gives useful information on the long run search properties of the GAs with respect to *both* solutions and states.

A result is given in [7, 17] concerning the number of states of a GA, in terms of

the chosen representation scheme, i.e., fixed length binary strings. We now present the following general result.

Proposition 4.5. *GA/WI with population size N, where the order in which the population is stored is irrelevant (sampling with replacement GA), operating on a solution space of size $|s| = k$ searches a state-space of size $|S^{[k]}|$, where*

$$|S^{[k]}| = \binom{N+k-1}{k-1}. \tag{4.3}$$

Proof. A population configuration $(X_1, X_2, \ldots, X_k)$ is a valid state of GA/WI if and only if

$$X_1 + X_2 + \cdots + X_k = N \tag{4.4}$$

by definition of $\boldsymbol{X} \in S^{[k]}$. Thus the total number of valid states is given by the total number of distinct solutions of this equation in non-negative integers, which is a well-known problem from combinatorics, with solution as stated [19]. □

Therefore, the number of states of GA/WI is a function of the solution space and the population size alone, regardless of the chosen solution encoding scheme. It will be shown later that a fundamental difference between GA/WI and GA/WO is the number of states of each algorithm; GAs which contain operators which do not sample with replacement (i.e., GA/WO-type algorithms) will be shown to have more states than those which do not contain such operators.

With the necessary definitions of the underlying random variables (solutions) and random vectors (states) of the following stochastic model in place, we now proceed to calculate the conditional probability distributions of these entities; for GA/WI(N, s), with random sampling of solutions, the probability that solution i is sampled from population $\boldsymbol{X} = (X_1, X_2, \ldots, X_k) \in S^{[k]}$ is X_i/N where $1 \leqslant i \leqslant k$. Therefore, the probability of $\boldsymbol{Y} = (Y_1, Y_2, \ldots, Y_k) \in S^{[k]}$ instances of solutions $(1, \ldots, k)$ at time $t+1$ given $\boldsymbol{X} = (X_1, X_2, \ldots, X_k)$ instances at time t is

$$\begin{aligned} &\Pr\{(Y_1, Y_2, \ldots, Y_k)_{t+1} \,|\, (X_1, X_2, \ldots, X_k)_t\} \\ &\quad = P_{\boldsymbol{XY}} = \frac{N!}{Y_1! Y_2! \cdots Y_k!} \left(\frac{X_1}{N}\right)^{Y_1} \left(\frac{X_2}{N}\right)^{Y_2} \cdots \left(\frac{X_k}{N}\right)^{Y_k}. \end{aligned} \tag{4.5}$$

This equation gives the conditional probability distribution of the random vector $(Y_1, Y_2, \ldots, Y_k)$ at time $t+1$. Note that the state-space of the Markov process defined by (4.5) is the same as the space defined in Definition 4.2, i.e., it consists of all distinct vectors $(X_1, X_2, \ldots, X_k) \in S^{[k]}$. Each solution has a one-step conditional probability distribution given by [11]

$$\Pr\{(Y_i)_{t+1} \,|\, (X_1, X_2, \ldots, X_k)_t\} = \sum_{Y_1} \sum_{Y_2} \cdots \sum_{Y_{i-1}} \sum_{Y_{i+1}} \cdots \sum_{Y_k} P_{\boldsymbol{XY}}, \tag{4.6}$$

where $\boldsymbol{X}, \boldsymbol{Y} \in S^{[k]}$. Substituting (4.5) into (4.6) shows that the numbers of solution

i, $1 \leq i \leq k$, within the population at time t, $(X_i)_t$ is a random variable with a binomial distribution. In [7] certain basic properties of the model (4.5) are investigated for the case of binary representation schemes.

4.2. *Incorporating fitness-based selection*: GA/WI(N, s, f, *roulette wheel*)

We consider the standard roulette-wheel-based selection procedure [3,10]. Since each solution has a strictly positive cost, or fitness value, i.e., to $(X_1, X_2, \ldots, X_k) \in S^{[k]}$ there corresponds a fitness vector, $f = (f_1, \ldots, f_k)$ with $f_i > 0$, $1 \leq i \leq k$, if we select from the population at time t (with replacement) on the basis of roulette-wheel fitnesses then we have that the probability that solution i is sampled from population $X \in S^{[k]}$ is

$$r_{xi} = \frac{X_i f_i}{\sum_{j=1}^{k} X_j f_j}, \tag{4.7}$$

so that

$$\Pr\{(Y_1, Y_2, \ldots, Y_k)_{t+1} \mid (X_1, X_2, \ldots, X_k)_t\} = \frac{N!}{\prod_{i=1}^{k} Y_i!} \prod_{i=1}^{k} r_{xi}^{Y_i}, \tag{4.8}$$

where by r_{xi} is meant the relative fitness of solution i conditional upon population X. Since the fitness of a solution is assumed fixed, we have again that this model defines a time-homogeneous first-order Markov process, over the states defined as before by $X = (X_1, X_2, \ldots, X_k) \in S^{[k]}$. This model is also discussed in [7] for the special case of binary string representation of solutions [10]. In the next section, we discuss the problem of introducing the GA *mutation* and *crossover* operators into this model, in a form which leaves the model representation-independent.

4.3. *Incorporating mutation and crossover:* GA/WI(N, s, f, *roulette wheel*, *mutation*, *crossover*)

Since our models are to be representation-independent, we make the following definitions, which capture in simple form the stochastic properties of the genetic operators *mutation* and *crossover*.

Definition 4.6. Let η be a row-stochastic mutation matrix, i.e., η_{ij} stores the probability that solution i transforms to solution j under *mutation*, i.e.,

$$\sum_{j=1}^{k} \eta_{ij} = 1. \tag{4.9}$$

For example, for a binary representation with strings of fixed length L and using the fixed mutation probabilities $p_m \leq 1/2$, used in [6, 10], we have that

$$\eta_{ij} = p_m^{H(i,j)}(1-p_m)^{L-H(i,j)}, \quad \eta_{ij} = \eta_{ji} > 0 \tag{4.10}$$

[17], where $H(i, j)$ represents the Hamming distance between the binary string encodings of solutions i and j [17]. To both incorporate *crossover* and to remain representation-independent, we define a crossover probability matrix, which gives the probability that a fixed number of (ordered) parent solutions will generate any fixed number of (ordered) child solutions under the action of some *crossover* operation. Thus, this matrix is indexed by ordered pairs, triples, etc. The "minimal" crossover probability matrix is defined below.

Definition 4.7. Let $C_{ij,m}$, $j \geqslant i$, be a row-stochastic crossover probability matrix, i.e.,

$$\sum_{m=1}^{k} C_{ij,m} = 1 \quad \forall 1 \leqslant i \leqslant j \leqslant k, \tag{4.11}$$

that stores the probability with which solutions i, j, will produce solution m under the action of *crossover* ($1 \leqslant i \leqslant j \leqslant k$, $1 \leqslant m \leqslant k$). Moreover, we define $C_{ij,i} > 0$ and $C_{ij,j} > 0$ (non-zero probability of parental retention). Note that the rows of C are indexed by ordered pairs of parent solutions, the columns by single child solutions.

An example is given in Appendix A of a crossover probability matrix for a binary string encoding of solutions, where the string length is 2. Note that we force the parent solutions to be gathered in an ordered pair, which forces uniqueness of reference.

With this definition in place, it is now possible to see that this particular *crossover* is a 2-parent 1-child crossover operation, as implemented in, for example, [14], i.e., 2-parent solutions will always generate only one child (which may be one of the parents). It was noticed that in fact it is quite possible to define an X-parent Y-child crossover operation [6], where $X \geqslant 2$ and $Y \geqslant 1$. Therefore, it should be pointed out that descriptions of *crossover* in the literature are mere instances of an X-parent Y-child crossover operation, i.e., the C matrix can be easily extended to triples, etc., of both parents and children; the essential requirement is that the pairs, triples, must be distinct so that the solution production probabilities under *crossover* are not over/underestimated within the C matrix. Since our definition is the simplest possible, we shall develop a Markov Chain model for GA/WI which uses this definition. As usual, the previous two definitions apply to the given genetic operators regardless of solution encoding scheme.

With these definitions of *mutation* and *crossover* in place, we now proceed to calculate the probability that solution i, $1 \leqslant i \leqslant k$, is produced given $X = (X_1, X_2, \ldots, X_k) \in S^{[k]}$, under the action of roulette-wheel fitness selection of two parents (with replacement), then *mutation* of one parent and then *crossover* between the mutant and non-mutant to produce solution i; this is clearly given by, in terms of the unions and intersections of probabilistic events,

$$\Pr\left\{\bigcup_{j\in s}\bigcup_{d\in s}\left(j_{\mathrm{fs}}\cap d_{\mathrm{fs}}\cap\left\{\bigcup_{b\in s} j\underset{\eta}{\rightarrow}b\cap b,\, d\underset{C}{\rightarrow}i\right\}\right)\right\} \tag{4.12}$$

$$=\Pr\left\{\bigcup_{j\in s} j_{\mathrm{fs}}\cap\bigcup_{b\in s} j\underset{\eta}{\rightarrow}b\cap\bigcup_{d\in s} d_{\mathrm{fs}}\cap d,\, b\underset{C}{\rightarrow}i\right\} \tag{4.13}$$

(where subscript "fs" refers to fitness selection) which has probability

$$p_{xi}=\sum_{j=1}^{k}\frac{X_j f_j}{X\cdot f}\sum_{b=1}^{k}\eta_{jb}\left[\sum_{d=1}^{b}\frac{X_d f_d}{X\cdot f}C_{db,i}+\sum_{d=b+1}^{k}\frac{X_d f_d}{X\cdot f}C_{bd,i}\right], \tag{4.14}$$

where by p_{xi} is meant the probability of generation of solution i by GA/WI, conditional upon population X, where $1\leqslant i\leqslant k$. This gives the stochastic model for GA/WI, with the given operators, as

$$P_{XY}=\frac{N!}{Y_1!Y_2!\cdots Y_k!}\prod_{i=1}^{k}(p_{xi})^{Y_i}, \tag{4.15}$$

where $Y\in S^{[k]}$. Note that the independence of parent selection gives (4.13) from (4.12). The row-stochasticity of P follows from that of the given p_{xi} terms (which follows by the previous two definitions) and the properties of the multinomial distribution [12]. With *mutation* of both parents, the stochastic model would be obtained in a similar manner, for $X, Y\in S^{[k]}$,

$$P_{XY}=\frac{N!}{Y_1!Y_2!\cdots Y_k!}\prod_{i=1}^{k}(p_{xi})^{Y_i}, \tag{4.16}$$

$$p_{xi}=\sum_{j=1}^{k}\frac{X_j f_j}{X\cdot f}\sum_{d=1}^{k}\frac{X_d f_d}{X\cdot f}\sum_{b=1}^{k}\eta_{jb}\sum_{e=1}^{k}\eta_{de}C^{*}_{eb,i},$$

$$C^{*}_{eb,i}=\begin{cases}C_{eb,i}, & e\leqslant b,\\ C_{be,i}, & e>b.\end{cases}$$

4.4. *Analysis of the models for* GA/WI(N, s, f, *roulette wheel, mutation, crossover*)

We proceed in this section to present three results relating to the search properties of GA/WI. The first result presents sufficient conditions for GA/WI to converge to a stationary distribution; the second result presents a special case of GA/WI search which has important implications for Genetic Algorithm parameterisation and representation issues, and the third result presents a long-run approximation result for GA/WI with respect to the implementation of *crossover*. We first show the usefulness of the independence from solution encoding schemes of the models by presenting the following for model (4.15). A convergence theorem for (4.16) may be obtained similarly.

4.4.1. Parameter choice for convergence

Theorem 4.8. *GA/WI with $\eta > 0$, and C as defined, converges in limit towards a stationary probability distribution $\boldsymbol{q}$ where q_X is strictly positive for all $X \in S^{[k]}$, i.e.,*

$$\lim_{t \to \infty} P^t = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{q} \\ \vdots \\ \boldsymbol{q} \end{bmatrix}. \tag{4.17}$$

Also, convergence takes place at least geometrically, i.e., there exists $0 \leq r < 1$, s.t.

$$P_{XY}^{(t)} = q_Y + e_{XY}^{(t)} \tag{4.18}$$

for all states $X, Y \in S^{[k]}$, where

$$|e_{XY}^{(t)}| \leq r^t . \tag{4.19}$$

Finally, $\boldsymbol{q}$ is independent of the initial choice of population $X^{(0)}$.

Proof. Suppose we are given arbitrary states $X, Y \in S^{[k]}$ where we transit from X to Y in one time step according to the appropriate probability in $\boldsymbol{P}$ given by (4.15). Then, there are two possible cases;

Case 1: $X_I = 0$, *some* $1 \leq I \leq k$. Then there exists at least one component X_j in X, with corresponding relative fitness r_{xj} greater than zero, since the population is of fixed size N. Thus,

$$\begin{aligned} p_{xI} &= r_{xj}\eta_{jI}r_{xj}C_{jI,I} + \cdots > 0 \quad (j \leq I), \\ p_{xI} &= r_{xj}\eta_{jI}r_{xj}C_{Ij,I} + \cdots > 0 \quad (j > I). \end{aligned} \tag{4.20}$$

Case 2: $X_I > 0$, $1 \leq I \leq k$. Then

$$p_{xI} = r_{xI}\eta_{II}r_{xI}C_{II,I} + \cdots > 0 . \tag{4.21}$$

Therefore, for arbitrary states $X, Y \in S^{[k]}$ the entry in $\boldsymbol{P}$ consists of a product of strictly positive numbers, since I was also arbitrary, which implies that $\boldsymbol{P}$ is strictly positive. The result follows by application of Theorems 3.2 and 3.3. □

Note that *mutation* is often implemented for binary-encoded GAs in such a way as to force $\eta > 0$, as mentioned previously [7, 17]. In the next section we show that it is possible to parameterise GA/WI in a way which completely removes bias towards solutions based on cost.

4.4.2. Special case

Proposition 4.9. *GA/WI has a random-search-type algorithm over the solution space as a special case, i.e., there exists at least one choice for C, s.t. the underlying stochastic process of GA/WI is a Markov Chain in which any given $k-1$ solutions*

can be represented by independent and identically distributed random variables. Further, the Markov Chain under this choice of C converges to stationarity immediately, i.e., in one generation.

Proof. Set $C_{bd,t} = 1/k\ \forall 1 \leq b \leq d \leq k,\ 1 \leq t \leq k$. Then,

$$p_{xi} = 1/k \quad \forall 1 \leq i \leq k\,, \quad \forall \boldsymbol{X} \in S^{[k]} \tag{4.22}$$

by substitution in (4.14) so that

$$P_{\boldsymbol{XY}} = \frac{N!}{Y_1!Y_2!\cdots Y_k!}\frac{1}{k^N}\,, \tag{4.23}$$

where $\boldsymbol{Y} \in S^{[k]}$. Since the stationary distribution of a homogeneous stochastic process, if it exists, is unique, we can "guess"

$$q_{\boldsymbol{X}} = \frac{N!}{X_1!X_2!\cdots X_k!}\frac{1}{k^N}\,, \tag{4.24}$$

and check by substitution in (3.8)

$$\sum_{\boldsymbol{X}\in S^{[k]}} q_{\boldsymbol{X}} P_{\boldsymbol{XY}} = \sum_{\boldsymbol{X}\in S^{[k]}} \left[\frac{N!}{X_1!X_2!\cdots X_k!}\frac{1}{k^N}\frac{N!}{Y_1!Y_2!\cdots Y_k!}\frac{1}{k^N}\right] = P_{\boldsymbol{XY}} \tag{4.25}$$

and

$$\sum_{\boldsymbol{X}\in S^{[k]}} q_{\boldsymbol{X}} = 1\,, \qquad q_{\boldsymbol{X}} > 0 \quad \forall \boldsymbol{X}\,, \tag{4.26}$$

where $\boldsymbol{X} \in S^{[k]}$. Thus the chosen $\boldsymbol{q}$ is indeed the stationary distribution of GA/WI with $C = [1/k]$. To show that the GA converges in one generation, we observe that

$$(P^2)_{\boldsymbol{XY}} = \sum_{\boldsymbol{M}\in S^{[k]}} \frac{N!}{M_1!M_2!\cdots M_k!}\frac{1}{k^N}\frac{N!}{Y_1!Y_2!\cdots Y_k!}\frac{1}{k^N} = P_{\boldsymbol{XY}}\,, \tag{4.27}$$

and so $\boldsymbol{P}$ is its own limiting matrix, by repeated substitution. To show that the random variables X_i, $1 \leq i \leq k$, have the same marginal distributions, observe that each of these random variables has a marginal distribution at stationarity which is binomially distributed with probability parameter $1/k$, i.e., at stationarity, the unconditional probability of obtaining j of solution i is given by

$$\Pr[X_i = j] = \binom{N}{j}\left(\frac{1}{k}\right)^j\left(1-\frac{1}{k}\right)^{N-j}\,, \quad t = 1, 2, \ldots \tag{4.28}$$

(by use of (4.6)) and this holds for all $1 \leq i \leq k$. Notice that, since $X_1 + X_2 + \cdots + X_k = N$, $\forall \boldsymbol{X} \in S^{[k]}$, only $k-1$ of the given random variables are independent [11], as claimed, and this completes the proof. □

Again, an analogous result holds for model (4.16), for GA/WI with *mutation* of both parents. We describe the algorithm with the previous *crossover* operator

choice as "random search" since the random variables X_i, $1 \leq i \leq k$, are all identically distributed at all times during the search (since the algorithm converges immediately), i.e., there is no selection bias towards any solution based on cost or otherwise. In fact, there are other choices for C (and η) which provide this result also; we sought only to prove an existence result for the special case. The above result has important implications for the parameterisation and solution encoding schemes of Genetic Algorithms with respect to the implementation of *crossover*. Essentially, GA/WI-type Genetic Algorithms which do not relate the probability of child production under *crossover* to any of the properties of the given parents are degenerate algorithms with respect to the search for optimal solutions; thus, we strongly conjecture that a fundamental reason for the observed success of binary-encoded Genetic Algorithms depends to a certain extent on the properties of the binary encoding scheme. In practical terms, the binary encoding scheme for solutions automatically prevents the degeneration of GA/WI-type GAs to (multidimensional) random search algorithms. In the next section, we extract information from the model (4.15) which also provides a measure of crossover bias with respect to the choice of *crossover* operators.

4.4.3. *Long run approximation result*

We now present a general result for GA/WI, where *mutation* of only one parent is carried out (4.15). As before, analogous results exist for the case where *mutation* of both parents is carried out (4.16). To proceed, we formulate definitions of certain parameters of GA/WI, which capture fundamental properties of these operators.

Definition 4.10. Let $\theta_t = \min C_{bd,t}\ \forall 1 \leq b \leq d \leq k,\ 1 \leq t \leq k$.

Thus, θ_t is the minimum component of the vector which is the tth column of C. Note that θ_t may vary across columns of C (solutions).

Definition 4.11. Let $\varepsilon_t = \max C_{bd,t}\ \forall 1 \leq b \leq d \leq k,\ 1 \leq t \leq k$.

Thus, ε_t is the maximum component of the vector which is the tth column of C. Note that ε_t may also vary across columns of C (solutions).

Definition 4.12. Let μ_t^∞ be the long run expected numbers of solution t in the population, i.e., the expected value of random variable X_t within the stationary distribution $\boldsymbol{q}$ where $1 \leq t \leq k$.

Definition 4.13. Let e be a matrix which rows store the population vectors in the same order in which they index $\boldsymbol{P}$ and $\boldsymbol{q}$, so that e is of size $|S^{[k]}| \times |s|$. Then $\boldsymbol{e}_t$ is a vector containing X_t of solution t across the populations.

With these definitions, we state and prove the following approximation guarantee for GA/WI with *mutation* of one parent.

Theorem 4.14. *For GA/WI, with mutation of one parent only* (4.15) *and with* $\eta > 0$, *the following holds*:

$$N\theta_t \leq \mu_t^\infty \leq N\varepsilon_t \,. \tag{4.29}$$

Proof. By definition of the expected value of a random variable within a multivariate distribution [11,12],

$$\mu_t^\infty = \sum_{X \in S^{[k]}} q_X \cdot X_t = \boldsymbol{q} \cdot \boldsymbol{e}_t = (\boldsymbol{q}P) \cdot \boldsymbol{e}_t = \boldsymbol{q} \cdot (P \cdot \boldsymbol{e}_t) = \boldsymbol{q} \cdot \boldsymbol{a} \,,$$

$$a_X = \left[\sum_{Y \in S^{[k]}} \frac{N!}{Y_1! \cdots Y_k!} \prod_{i=1}^{k} (p_{xi})^{Y_i} Y_t \right]_{1 \times |S^{[k]}|} \tag{4.30}$$

(using the fact that $\boldsymbol{q}$ is a stationary distribution for $\boldsymbol{P}$). By the formula for the expected value of the multinomial distribution this gives

$$\mu_t^\infty = N \sum_{X \in S^{[k]}} q_X \left\{ \sum_{j=1}^{k} \frac{X_j f_j}{\boldsymbol{X} \cdot \boldsymbol{f}} \sum_{b=1}^{k} \eta_{jb} \left[\sum_{d=1}^{b} \frac{X_d f_d}{\boldsymbol{X} \cdot \boldsymbol{f}} C_{db,t} + \sum_{d=b+1}^{k} \frac{X_d f_d}{\boldsymbol{X} \cdot \boldsymbol{f}} C_{bd,t} \right] \right\} , \tag{4.31}$$

so that the result follows by simple substitution of the given θ_t and ε_t into (4.31).

By substituting the parameters from Proposition 4.9, we see that in these cases the bounds are exact, i.e., they return $\mu_t^\infty = N/k$, as expected. Again, this justifies the use of the term "random search" within that proposition, since in the long run we expect equal numbers of all solutions within the population. Note that an analogous result holds for model (4.16).

Therefore, this long run approximation guarantee can be used to analyse certain cases of *crossover* together with a given population size; it can be seen that the lower bound on μ_t^∞ increases in value as either the population size N becomes larger (for $\theta_t > 0$) or θ_t increases, and the upper bound on μ_t^∞ decreases as either the population size or ε_t decreases; neither of these facts is intuitively obvious from the definition of GA/WI-type algorithms in Section 2. Thus, these bounds highlight the existence of *crossover bias* within GAs, since the bounds given above are independent of the costs of solutions. The existence of this bias is of fundamental importance in terms of optimisation via GAs, since for a given optimisation problem k is generally very large and the costs f are not known beforehand; the existence of crossover bias, for example towards lower cost solutions (depending on the given encoding scheme and choice of crossover operator), for example towards lower costs solutions is generally undesirable. Another important point is that these bounds do not depend on any particular representation scheme, i.e., they may prove useful for the analysis of non-binary encoding schemes, where the choice of *crossover* operator is not immediately apparent [6,10].

Finally, we note that we are justified in calculating the expected values of the

stationary distribution, since, the stochastic process converges in limit (by Theorem 4.8).

5. Explicit stochastic model for GA/WO(N, s, f, *roulette wheel*, *mutation*, *crossover*)

We now proceed to formulate a stochastic model for GA/WO; we proceed at a slightly faster pace than for GA/WI, partly because the resultant model is of a simpler form than for the latter, and partly because some analogous concepts with respect to vector-indexed matrix multiplication etc. need not be made here, for brevity. We present important results for GA/WO-type Genetic Algorithms with respect to the size of the state-space through which they search, sufficient conditions for convergence in limit of these algorithms, and a special case analysis which underpins results observed experimentally, in that excessive mutation rates cause a degradation in performance of the algorithm with respect to the discovery of higher fitness solutions [6,10]. We also present important results which show the relationship between GA/WO- and GA/WI-type Genetic Algorithms; we show that GA/WI and GA/WO possess differing state-spaces, that they possess differing transition probability matrices (and therefore differing search properties); finally, we show that for algorithms such as GA/WO, the *order* in which the solutions are stored within the population fundamentally affects the search dynamics of the algorithm.

5.1. Definitions of GA/WO states/operators

In GA/WO, each of the genetic operators acts upon the population separately (as opposed to GA/WI) so that each in effect is a stochastic process which produces a new population. In particular, for the *mutation* and *crossover* operators, since they are effected to *specific locations* within the population (*mutation* is applied singly to specific locations, *crossover* pairwise) we merely assume that the solutions are stored within the population in list-like fashion, as in the GA implemented in [6]. Thus, we introduce new definitions for the states of GA/WO;

Definition 5.1. Let $i = (i_1, \ldots, i_N)_t$ store the order in which the solutions are contained in the population of GA/WO at time t, where $1 \leq i_m \leq k$, $1 \leq m \leq N$.

Thus, every component of the GA/WO state vectors is some solution (whereas every component of the GA/WI state vectors *counts* some solution).

Definition 5.2. Let $S'^{[N]} = \{i = (i_1, \ldots, i_N): 1 \leq i_m \leq k, 1 \leq m \leq N\}$. Thus, $S'^{[N]}$ is the set consisting of all valid populations of GA/WO.

These definitions lead to:

Proposition 5.3. *GA/WO with population size N, operating on a solution space of size* $|s| = k$ *searches a state-space of size* $|S'^{[N]}|$, *where*

$$|S'^{[N]}| = k^N . \tag{5.1}$$

Proof. The total number of population states of GA/WO is given by the total number of *distinct* permutations of the population states of GA/WI, i.e., by

$$\sum_{X \in S^{[k]}} \binom{N}{X} , \tag{5.2}$$

since, every state vector $X \in S^{[k]}$ of GA/WI will have the bracketed term in (5.2) distinct corresponding permutations, which, within GA/WO, are all distinct states; then the result (5.1) follows from a modification of the well-known result given in [1, p. 49]. □

Thus, we present the following inequality between GA/WI and GA/WO:

Theorem 5.4. *Let* $|S^{[k]}|$ *be given by* (4.3), *and* $|S'^{[N]}|$ *by* (5.1). *Then* $|S'^{[N]}| > |S^{[k]}|$ *for* $k > 1$ *and* $N > 1$.

Proof. Observe that $Nk > k + N - 1$ for $k,N > 1$. Thus,

$$k \cdot 2k \cdots Nk > k(k+1) \cdots (k+N-1) , \tag{5.3}$$

so that

$$|S'^{[N]}| = k^N > \frac{k(k+1) \cdots (k+N-1)}{N!} = |S^{[k]}| , \tag{5.4}$$

which completes the proof. □

Note that $k > 1$ corresponds to nontrivial optimisation, and $N > 1$ to a nontrivial Genetic Algorithm. Thus, the number of states of GA/WO is shown here to be strictly larger than that of GA/WI. This fact has been previously ignored within the literature; some authors give the left-hand side of (5.4) as the size of the Genetic Algorithm state-space [17]; others give the right-hand side [7]. Here, we have shown which Genetic Algorithms possess which state-space, with respect to the *implementation* of the genetic operators; the state-space of GA/WO being strictly larger than that of GA/WI.

As is evident from the definitions of GA/WI and GA/WO, the single-child crossover probability matrix is insufficient to describe the stochastic properties of the GA/WO *crossover* operator (which produces two children from two parents). Therefore, we introduce:

Definition 5.5. Let $C_{ab,gh}$ be a row-stochastic crossover probability matrix, i.e.,

$$\sum_{g=1}^{k}\sum_{h=g}^{k} C_{ab,gh} = 1 \quad \forall 1 \leq a \leq b \leq k\,,$$

that stores the probability with which solutions a, b will produce solutions g, h under the action of *crossover* for $1 \leq a \leq b \leq k$, $1 \leq g \leq h \leq k$. Moreover, we define $C_{ij,ij} > 0$ (since whether or not two solutions are combined is the result of a probabilistic test [6, 10]).

Note that we force both the parent and child solutions to be gathered in an ordered pair, which again forces uniqueness of reference. The 2-parent, 2-child crossover probability matrix has the following useful property.

Proposition 5.6.

$$\sum_{g=1}^{k}\sum_{h=g}^{k} C_{ab,gh} = \sum_{g=1}^{k}\sum_{h=1}^{g} C_{ab,hg} = 1 \quad \forall 1 \leq a \leq b \leq k\,.$$

Proof. Setting $m = k - g + 1$, $1 \leq g \leq k$, $n = k - h + 1$, $1 \leq h \leq g \leq k$, gives

$$\begin{aligned}\sum_{g=1}^{k}\sum_{h=1}^{g} C_{ab,hg} &= \sum_{m=1}^{k}\sum_{n=m}^{k} C_{ab,(k-n+1)(k-m+1)} \\ &= \sum_{g=1}^{k}\sum_{h=g}^{k} C_{ab,gh} = 1 \quad \forall 1 \leq a \leq b \leq k\,,\end{aligned}$$

since the middle sum is merely a rearrangement of the outer two sums, and this completes the proof. □

This then gives the following useful property:

Proposition 5.7.

$$\sum_{g=1}^{k}\sum_{h=g+1}^{k} C_{ab,gh} = \sum_{g=1}^{k}\sum_{h=1}^{g-1} C_{ab,hg} \quad \forall 1 \leq a \leq b \leq k\,.$$

Proof. Each sum in the above is just the sums written in Proposition 5.6, excluding the term

$$\sum_{g=1}^{k} C_{ab,gg}$$

and this completes the proof. □

With these definitions and results in place, which describe in simple form the stochasticity of the GA/WO *crossover* operator, we formulate a definition which relates the selection operator of GA/WI to that of GA/WO (since both use

sampling with replacement selection operators). It is then used to formulate a transition matrix for the selection operator of GA/WO.

Definition 5.8. Let $(X_{i,b})_t$ store the numbers of solutions b within the population i of GA/WO at time t, i.e.,

$$X_{i,b} = d \Leftrightarrow \exists m_1, \ldots, m_d\,,\ m_y \neq m_z \quad \forall 1 \leq y \neq z \leq d$$
$$\text{s.t. } i_{m_1} = \cdots = i_{m_d} = b\,.$$

Thus, $(X_{i,1}, \ldots, X_{i,k})$ is a many-to-one mapping from the states of GA/WO to the states of GA/WI; it counts the total occurrences of solutions i_a, $1 \leq a \leq N$, within $i = (i_1, \ldots, i_N)$ of GA/WO so that each $(i_1, \ldots, i_N) \in S'^{[N]}$ has a unique counting vector $(X_{i,1}, \ldots, X_{i,k})$ but any given $(X_1, X_2, \ldots, X_k) \in S^{[k]}$ in Definition 4.2 can represent

$$\binom{N}{X} \tag{5.5}$$

different $(i_1, \ldots, i_N) \in S'^{[N]}$ vectors [1].

5.2. *Stochastic model for GA/WO(N, s, f, roulette wheel, mutation, crossover)*

We present in this section a stochastic model for GA/WO; we begin by presenting a stochastic model for the selection operator. An important point to be made is that populations $j \in S'^{[N]}$ which have the same counting vector $Y \in S^{[k]}$ will have the same probability of production from $i \in S'^{[N]}$, because within the GA/WO fitness selection operator the order in which the solutions are selected is unimportant (i.e., their ordering within population i); only the order in which they are *placed* within $j \in S'^{[N]}$ is important, by Definition 5.2, and so different permutations of $j \in S'^{[N]}$ thus have equal probability of occurrence, due to the sampling with replacement roulette-wheel operator (which implies independence between the selection events, as in GA/WI). Therefore, the relative fitness of solution b within population $i \in S'^{[N]}$ is given by

$$r_{xb} = \frac{X_{i,b} f_b}{\sum_{d=1}^{N} f_{i_d}} \quad 1 \leq b \leq k\,, \tag{5.6}$$

so that

$$F_{ij} = \Pr\{(j_1, j_2, \ldots, j_N) \mid (i_1, i_2, \ldots, i_N)\} = \prod_{b=1}^{k} r_{xb}^{Y_{j,b}} \tag{5.7}$$

is the transition matrix for the selection operator of GA/WO, where $i, j \in S'^{[N]}$ and where $X, Y \in S^{[k]}$ are the counting vectors for populations i and j respectively.

The transition matrix for the GA/WO *mutation* operator is of a simple form;

since *mutation* is carried out independently in list-like manner to the population, it also transforms population $\boldsymbol{i}$ to population, $\boldsymbol{j}$, $\boldsymbol{i},\boldsymbol{j} \in S'^{[N]}$,

$$M_{ij} = \Pr\{(j_1, \ldots, j_N) \mid (i_1, \ldots, i_N)\} = \prod_{d=1}^{N} \eta_{i_d j_d} . \tag{5.8}$$

The construction of the transition matrix for the *crossover* operator of GA/WO is non-trivial; therefore we make a definition which relates *crossover* probabilities to the C matrix;

Definition 5.9. Let $C^*_{ab,cd}$ be defined as

$$C^*_{ab,cd} = \begin{cases} C_{ab,cd}, & a \leqslant b, \quad c \leqslant d, \\ C_{ba,cd}, & a > b, \quad c \leqslant d, \\ C_{ab,dc}, & a \leqslant b, \quad c > d, \\ C_{ba,dc}, & a > b, \quad c > d, \end{cases}$$

i.e., $C^*_{ab,cd}$ gives the appropriate value within C for *crossover* between arbitrarily labelled solutions.

Since the *crossover* operator of GA/WO transforms population $\boldsymbol{i}$ to population $\boldsymbol{j}$, $\boldsymbol{i}, \boldsymbol{j} \in S'^{[N]}$ by the creation of solutions pairwise, the following equation relates the crossover probabilities to the child solutions placed within $\boldsymbol{j} \in S'^{[N]}$,

$$\begin{aligned} &\Pr\{(j_{2d-1}, j_{2d}) = (g, h) \mid (i_{2d-1}, i_{2d}) = (a, b)\} \\ &\quad = \Pr\{(j_{2d-1}, j_{2d}) = (h, g) \mid (i_{2d-1}, i_{2d}) = (a, b)\} , \end{aligned} \tag{5.9}$$

where $1 \leqslant d \leqslant N/2$, i.e., either ordering of the new pair of child solutions within the relevant locations of $\boldsymbol{j}$ is stipulated as equally likely, where each term in the above (5.9) is an ordered pair of solutions. Thus, we define:

Definition 5.10. Let $f(x, y)$ be defined as

$$f(x, y) = \begin{cases} 1, & x = y, \\ \frac{1}{2}, & x \neq y. \end{cases}$$

Note that f is symmetric about $x = y$. Thus, with all necessary definitions in place, observe that

$$\begin{aligned} &\Pr\{(j_{2d-1}, j_{2d}) = (g, h) \mid (i_{2d-1}, i_{2d}) = (a, b)\} \\ &\quad = \Pr\{ab \underset{C}{\rightarrow} gh \cap (j_{2d-1}, j_{2d}) = (g, h)\} \\ &\quad = f(g, h) C^*_{ab,gh} , \end{aligned} \tag{5.10}$$

where each term in brackets is an ordered pair. Thus, the general formula for the

conditional probability of obtaining (j_{2d-1}, j_{2d}) from (i_{2d-1}, i_{2d}) under crossover is

$$\Pr\{(j_{2d-1}, j_{2d}) \mid (i_{2d-1}, i_{2d})\} = f(j_{2d-1}, j_{2d}) C^*_{i_{2d-1}i_{2d}, j_{2d-1}j_{2d}}, \tag{5.11}$$

$1 \leqslant d \leqslant N/2$. Denoting by Q the transition matrix for the given *crossover* operator, we thus have

$$\begin{aligned} Q_{ij} &= \Pr\{(j_1, \ldots, j_N \mid (i_1, \ldots, i_N)\} \\ &= \Pr\left\{\bigcap_{d=1}^{N/2} [(i_{2d-1}, i_{2d}) \underset{C}{\rightarrow} (j_{2d-1}, j_{2d})]\right\} \\ &= \prod_{d=1}^{N/2} [f(j_{2d-1}, j_{2d}) C^*_{i_{2-1}i_{2d}, j_{2d-1}j_{2d}}], \end{aligned} \tag{5.12}$$

since the pairwise crossover events are independent of one another. To complete the construction of Q, we state and prove the following:

Proposition 5.11. *Let Q be defined as in* (5.12). *Then Q is a row-stochastic matrix, i.e.,*

$$\sum_{j \in S'^{[N]}} Q_{ij} = 1 \quad \forall i \in S'^{[N]}.$$

Proof. Observe that

$$\begin{aligned} \sum_{j \in S'^{[N]}} Q_{ij} &= \sum_{j_1=1}^{k} \sum_{j_2=1}^{k} \cdots \sum_{j_N=1}^{k} \left[\prod_{d=1}^{N/2} [f(j_{2d-1}, j_{2d}) C^*_{i_{2d-1}i_{2d}, j_{2d-1}j_{2d}}]\right] \\ &= \sum_{j_1=1}^{k} \sum_{j_2=1}^{k} f(j_1, j_2) C^*_{i_1 i_2, j_1 j_2} \cdots \sum_{j_{N-1}=1}^{k} \sum_{j_N=1}^{k} f(j_{N-1}, j_N) C^*_{i_{N-1}i_N, j_{N-1}j_N}, \end{aligned} \tag{5.13}$$

and so the general term is

$$\begin{aligned} &\sum_{j_{2d-1}=1}^{k} \sum_{j_{2d}=1}^{k} f(j_{2d-1}, j_{2d}) C^*_{i_{2d-1}i_{2d}, j_{2d-1}j_{2d}} \\ &\quad = \frac{1}{2} \sum_{j_{2d-1}=1}^{k} \sum_{j_{2d}=j_{2d-1}+1}^{k} C_{i_{2d-1}i_{2d}, j_{2d-1}j_{2d}} + \frac{1}{2} \sum_{j_{2d-1}=1}^{k} \sum_{j_{2d}=1}^{j_{2d-1}-1} C_{i_{2d-1}i_{2d}, j_{2d}j_{2d-1}} \\ &\qquad + \sum_{j_{2d-1}=1}^{k} C_{i_{2d-1}i_{2d}, j_{2d-1}j_{2d-1}} \end{aligned}$$

(via the properties of $f(x, y)$ and assuming $i_{2d-1} \leqslant i_{2d}$ without loss of generality, since the sum is over j components), where $1 \leqslant d \leqslant N/2$. By Proposition 5.7, the first two sums are equal, and, added to the third sum, this gives

$$\sum_{j_{2d-1}=1}^{k} \sum_{j_{2d}=1}^{k} f(j_{2d-1}, j_{2d}) C^*_{i_{2d-1} i_{2d}, j_{2d-1} j_{2d}}$$
$$= \sum_{j_{2d-1}=1}^{k} \sum_{j_{2d}=j_{2d-1}+1}^{k} C_{i_{2d-1} i_{2d}, j_{2d-1} j_{2d}} + \sum_{j_{2d-1}=1}^{k} C_{i_{2d-1} i_{2d}, j_{2d-1} j_{2d-1}} = 1$$

by Definition 5.5. Since d was arbitrarily chosen, (5.12) is shown to be a finite product of 1's, and so Q is indeed row-stochastic. □

Note that F is easily shown to be row-stochastic using the properties of the multinomial distribution, and the above analysis for Q can be repeated in a simpler form for M, to show that it is also row-stochastic. Thus, the overall stochastic model for GA/WO is

$$P_{ij} = (F \cdot M \cdot Q)_{ij} = \sum_{a \in S'^{[N]}} \left(\sum_{b \in S'^{[N]}} (F_{ib} M_{ba}) Q_{aj} \right) \quad \forall i, j \in S'^{[N]}, \tag{5.14}$$

where F, M and Q are row-stochastic matrices defined in (5.7), (5.8) and (5.12), and vector-indexed matrix multiplication is used throughout. Thus, it can be seen that the transition matrices of GA/WI (i.e., (4.15) and (4.16)) and GA/WO (i.e. (5.14)) differ; generally, the transition matrix of GA/WO is of higher order than that of GA/WI. Since knowledge of the transition matrix of a stochastic process is sufficient to predict the behaviour of that process at all time steps, given any initial probability distribution of the corresponding Markov Chain [15], this implies that GA/WO and GA/WI are quite distinct search algorithms; the models (4.15) and (4.16) and (5.14) capture this distinction in a concise form. Notice also that distinct orderings of the *solutions* with $i \in S'^{[N]}$ are essentially differing states of GA/WO, with corresponding differing transition probabilities within (5.14); thus, the *order* in which the solutions are stored within the populations of GA/WO determines the search dynamics of this algorithm.

5.3. *Analysis of model of GA/WO(N, s, f, roulette wheel, mutation, crossover)*

In this section we present two fundamental analyses of the model given in (5.14). We show that, if the *mutation* operator is implemented in the usual way [6, 17] that GA/WO also possesses a stationary distribution; we then show that, if each row of the mutation probability matrix is the discrete uniform distribution over the set of solutions $s = \{s_1, s_2, \ldots, s_k\}$, then the search conducted by GA/WO converges immediately to cost-independent search.

5.3.1. *Parameter choice for convergence*

We proceed to present a similar result to that of Section 4.4.1, in that we derive sufficient conditions for GA/WO to converge in limit towards a stationary distribution. We first formulate definitions essential to the method of proof:

Definition 5.12. Let $f_{\min} = \min\{f_1, \ldots, f_k\} > 0$.

Thus, $f_{\min}$ is equal to the cost of the solution(s) with lowest cost.

Definition 5.13. Let $f_{\max} = \max\{f_1, \ldots, f_k\} > 0$.

Thus, $f_{\max}$ is equal to the cost of the solution(s) with largest cost.

Definition 5.14. Let $\eta_{\min} = \min_{i,j} [\eta_{ij}]$.

Thus, $\eta_{\min}$ is the smallest entry in the mutation probability matrix. It is greater than zero when $\eta > 0$.

Definition 5.15. Let $C_{\min} = \min_{i \leqslant j} [C_{ij,ij}] > 0$.

Thus, $C_{\min}$ is the smallest probability that a pair of parents is retained under the GA/WO *crossover* operator. It is greater than zero by Definition 5.5.

Theorem 5.16. *GA/WO with $\eta > 0$, and C as defined, converges towards a stationary probability distribution, $\boldsymbol{q}'$, where q'_i is strictly positive for all $\boldsymbol{i}$, where $\boldsymbol{i}$ is a valid population state of the algorithm, i.e.,*

$$\lim_{t \to \infty} P^t = \begin{bmatrix} \boldsymbol{q}' \\ \boldsymbol{q}' \\ \vdots \\ \boldsymbol{q}' \end{bmatrix}. \tag{5.15}$$

Also, convergence takes place at least geometrically, i.e., there exists $0 < r < 1$ s.t.

$$P_{ij}^{(t)} = q'_j + e_{ij}^{(t)}, \tag{5.16}$$

where

$$|e_{ij}^{(t)}| \leqslant r^t. \tag{5.17}$$

Finally, $\boldsymbol{q}'$ is independent of the initial choice of population $\boldsymbol{i}_{(0)}$.

Proof. Observe that

$$P_{ij} = (F \cdot M \cdot Q)_{ij} \geqslant F_{ii} M_{ij} Q_{jj} \quad \forall \boldsymbol{i}, \boldsymbol{j} \in S'^{[N]}, \tag{5.18}$$

where vector-indexed multiplication is carried out in the general sense of Definitions 4.3 and 4.4. Now

$$F_{ii} = \prod_{b=1}^{k} r_{xb}^{X_{i,b}},$$

so that $r_{xb} \geqslant f_{\min}/(N f_{\max})$ where at least one component of $\boldsymbol{i}$ has value b, $\forall 1 \leqslant b \leqslant k$ and $\forall \boldsymbol{X} \in S^{[k]}$, since the numerator of the given fraction is smaller in fitness than all other solutions, and the denominator is the cost of the population with maximum fitness; thus,

$$F_{ii} = \prod_{b=1}^{k} r_{xb}^{X_{i,b}} \geqslant \left(\frac{f_{\min}}{Nf_{\max}}\right)^N > 0 \quad \forall i \in S'^{[N]}, \tag{5.19}$$

$$M_{ij} \geqslant (\eta_{\min})^N > 0 \quad \forall i, j \in S'^{[N]} \tag{5.20}$$

(since $\eta > 0$)

$$Q_{jj} \geqslant \left(\frac{C_{\min}}{2}\right)^{N/2} > 0 \quad \forall j \in S'^{[N]} \tag{5.21}$$

(by definition of Q), which together imply that the product given on the right-hand side of (5.18) is strictly positive, $\forall i, j \in S'^{[N]}$. Since $\boldsymbol{P}$ is then irreducible and aperiodic, the result follows by use of Theorems 3.3 and 3.4.

5.3.2. Special case

In the following special case of GA/WO, we show that both algorithms described in this paper contain similar degenerate cases; thus, the following result shows that, in the case where GA/WI is parameterised as in Proposition 4.9, and GA/WO as below, that both algorithms have identical search properties with respect to the absence of bias towards cost; however, the choice of which operator to parameterise is different.

Theorem 5.17. *Let $\eta = [1/k]_{k \times k}$. Then the resultant stationary distribution for GA/WO is*

$$q_j = \frac{1}{k^N} \sum_{a \in S'^{[N]}} Q_{aj}, \quad j \in S'^{[N]}. \tag{5.22}$$

Proof. By substitution of η into (5.8), it follows that

$$M_{ij} = \frac{1}{k^N} \quad \forall i, j \in S'^{[N]}. \tag{5.23}$$

Thus, by substitution into (5.14),

$$P_{ij} = \frac{1}{k^N} \sum_{a \in S'^{[N]}} Q_{aj} \quad \forall i \in S'^{[N]}, \tag{5.24}$$

$\forall i, j \in S'^{[N]}$ since F is row-stochastic. Substitution of (5.22) into the invariant equation (3.8) with P given by (5.24) obtains the result, observing by definition of Q that

$$q_j \geqslant \frac{1}{k^N} Q_{jj} > 0 \quad \forall j \in S'^{[N]}. \tag{5.25}$$

The corresponding result for immediate convergence follows; essentially, this is because P in (5.24) has identical rows, as does P of Proposition 4.9. □

Within a relaxed theoretical framework of discussion, we see the surprising result that GA/WI degenerates immediately to "fitness-free" search for "com-

plete" *crossover* regardless of *mutation*, and GA/WO degenerates immediately to "fitness-free" search for "complete" *mutation* regardless of *crossover*. It is worthy of note that $\boldsymbol{q}$ above in (5.22) is discrete uniform ($= [1/k^N]$ over $S'^{[N]}$) in the case where no crossover actually takes place (i.e., crossover parameter $p_c = 0$ in [10]), since Q in this case is the $|S'^{[N]}| \times |S'^{[N]}|$ identity matrix I. Again, as for GA/WI, there may be other parameterizations for GA/WO which produce cost-independent search; as in the case for the former, we sought to prove an existence result here.

The extension of the approximation guarantee (which measures crossover bias) for GA/WI, given in Section 4.4.3 to GA/WO-type algorithms is currently being developed by the authors, and will be reported in a future publication; it should be pointed out that the distributions given in the rows of the matrix defined by (5.14) are non-standard probability distributions, and thus introduce into this problem an element of intractability.

6. Conclusions and future directions

In this paper we have described a number of models for GAs, which were then used to analyse the convergence properties of the given algorithms. We defined, in terms of stochastic matrices, genetic mutation and crossover operators which captured their essential stochastic properties in such a way that the resultant models were independent of the representation of solutions, yet yielded sufficient conditions for convergence. We made an important distinction between GAs which sample solutions with and without replacement from populations in terms of the size of the state-space of both algorithms. We presented, via definitions of genetic operators, stochastic models for the two classes of algorithm. We then analysed particular cases of genetic search, showing sufficient conditions for both classes for degeneracy to randomised search, and presented an approximation guarantee for optimisation via sampling with replacement GAs; this guarantee indicates the existence of bias within the GA crossover operator towards given solutions.

Future important theoretical work will concentrate on the presentation of the stationary distributions of GAs with parameters other than those used in our special case analyses, for both classes of algorithm. The extension of the estimation of the first moments of the stationary distributions of GA/WI-type GAs to higher moments represents important theoretical and practical work; the estimation of these moments is significant, due to the multivariate nature of the stationary distribution of the algorithms, and provides guidance with respect to algorithm parameterisation, as shown in Sections 4.4.2 and 4.4.3. It may also be possible to estimate these moments using particular cost functions, mutation and crossover probability matrices. Other important work concerns the analysis of the phenomenon known as "premature convergence of Genetic Algorithms" [3, 6, 10], together with the rate of convergence of Genetic Algorithms towards their stationary distributions [18], by use of these models; this is currently being investigated by the authors, and will be reported in a future publication. These

tasks represent important problems with respect to the theoretical and practical aspects of optimisation via Genetic Algorithms.

Acknowledgements

Both authors would like to thank Dr. C.N.B. Martin, Clean Energy Systems, National Engineering Laboratory, Glasgow for his interest in this work. David Reynolds is grateful to Glasgow Caledonian University for financial support during this work. He would also like to express thanks to members of staff of the Department of Mathematics, Glasgow Caledonian University for encouragement.

Appendix A. Example of crossover probability matrix, binary string encoding scheme of length 2

Example A.1 (Binary string uniform crossover [3, 16]). This can be implemented by simultaneously moving along both parents and selecting a bit from either parent with probability 1/2. Thus, in this case, for strings of length 2, the crossover probability matrix is given in Table A.1.

Table A.1
Uniform crossover probability matrix for binary string ($L = 2$) GA

$C_{ij,m}$	00	01	10	11
00 00	1	0	0	0
00 01	0.5	0.5	0	0
00 10	0.5	0	0.5	0
00 11	0.25	0.25	0.25	0.25
01 01	0	1	0	0
01 10	0.25	0.25	0.25	0.25
01 11	0	0.5	0	0.5
10 10	0	0	1	0
10 11	0	0	0.5	0.5
11 11	0	0	0	1

References

[1] I. Anderson, *A First Course in Combinatorial Mathematics* (OUP, Oxford, 1985).
[2] D.L. Battle and M.D. Vose, Isomorphisms of genetic algorithms, *Artif. Intell.* **60** (1993) 155–165.
[3] D. Beasley, D.R. Bull and R.R. Martin, An overview of Genetic Algorithms: Part 2, research topics, *Univ. Comput.* **15** (1993) 170–181.
[4] R.N. Bhattacharya and E.C. Waymire, *Stochastic Processes with Applications* (Wiley, New York, 1990).
[5] E. Cinlar, *Introduction to Stochastic Processes* (Prentice-Hall, Englewood Cliffs, NJ, 1975).

[6] L. Davis, What is a Genetic Algorithm, in: L. Davis, ed., *Handbook of Genetic Algorithms* (Van Nostrand Reinhold, New York, 1991) Chapter 1.
[7] T.E. Davis and J.C. Principe, A simulated annealing like convergence theory for the simple genetic algorithm, in: R.K. Belew and L.B. Booker, eds., *Proceedings of the Fourth Conference on Genetic Algorithms*, San Mateo, CA (Morgan Kaufmann, San Mateo, CA, 1991) 174–181.
[8] A.E. Eiben, E.H.L. Aarts and K.M. Van Hee, Global convergence of genetic algorithms: a Markov chain analysis, in: H.P. Schwefel and R. Manner, eds., *Parallel Problem Solving from Nature* (Springer, Berlin, 1991) 4–12.
[9] D.E. Goldberg, Finite Markov chain analysis of Genetic Algorithms, in: J.J. Grefenstette, ed., *Proceedings of the Second International Conference on Genetic Algorithms* (Lawrence Erlbaum, Hillsdale, NJ, 1987) 1–8.
[10] D.E. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine Learning* (Addison-Wesley, Reading, MA, 1989).
[11] A.F. Karr, *Probability* (Springer-Verlag, New York, 1993).
[12] W. Mendenhall, D.D. Wackerly and R.L. Scheaffer, *Mathematical Statistics with Applications* (Wiley, New York, 1993).
[13] H. Minc, *Nonnegative Matrices* (Wiley, New York, 1988).
[14] D.J. Montana, Automated parameter tuning for interpretation of synthetic images, in: L. Davis, ed., *Handbook of Genetic Algorithms* (Van Nostrand Reinhold, New York, 1991).
[15] U. Narayan Bhat, *Elements of Applied Stochastic Processes* (Wiley, New York, 2nd ed., 1984).
[16] J.C. Potts, T.D. Giddens and S.B. Yadav, The development and evaluation of an improved Genetic Algorithm based on migration and artificial selection, *IEEE Trans. Syst. Man Cybern.* **24** (1) (1993) 73–86.
[17] G. Rudolph, Convergence analysis of canonical Genetic Algorithms, *IEEE Trans. Neural Net.* **5** (1) (1994) 96–101.
[18] E. Seneta, *Non-negative Matrices and Markov Chains* (Springer-Verlag, New York, 2nd ed., 1981).
[19] J.H. van Lint and R.M. Wilson, *A Course in Combinatorics* (CUP, Cambridge, 1993).

ELSEVIER

Artificial Intelligence 82 (1996) 331-339

Artificial Intelligence

Research Note

A new theorem in particle physics enabled by machine discovery

Raúl E. Valdés-Pérez *

Computer Science Department and Center for Light Microscope Imaging and Biotechnology, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Received February 1995; revised December 1995

Abstract

A widespread objection to research on scientific discovery is that there has been a noticeable dearth of significant novel findings in domain sciences contributed by machine discovery programs. The implication is that the essential parts of the discovery process are not captured by these programs. The aim of this note is to document for the AI audience a novel finding in particle physics that was enabled by the machine discovery program PAULI reported previously. This finding consists of a theorem that expresses the minimum number of conservation laws that are needed, mathematically speaking, to account for any consistent experimental data on particle reactions. This note also reports how a puzzle raised by the theorem—its conflict with physics practice—is resolved.

1. Introduction

A widespread objection to research on computational scientific discovery is that there has been a noticeable dearth of significant novel findings in domain sciences contributed by discovery programs. The implication is that the essential parts of the discovery process are not captured by these programs. If this implication is true, then the whole automated scientific discovery enterprise is so far of doubtful soundness. Of even wider consequence is that the theory of heuristic search is seriously incomplete, since it fails to account for a salient aspect of human reasoning: discovery in science. Therefore, it becomes critical to record instances of novel machine discovery in order to falsify the premise that underlies this serious implication, i.e., the premise that discovery programs have enabled no significant new discoveries.

* E-mail: valdes@cs.cmu.edu.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0004-3702(95)00128-X

The aim of this note is to document for the AI audience a novel finding in particle physics that was enabled by the machine discovery program PAULI reported previously [4]. This finding consists of a theorem that expresses the minimum number of conservation laws that are needed, mathematically speaking, to account for any consistent experimental data on particle reactions.

I proceed by describing briefly the discovery task and summarizing prior work on it. Then I present the novel finding and the circumstances that led to it, and explore some of the implications of this finding within the particle physics domain. Since our research on scientific discovery emphasizes generality within science, I examine briefly the generic character of this task in science generally. This note closes by summarizing the lessons for machine discovery.

2. The particle physics task

The task of postulating conservation laws from observational data on particle reactions was apparently first mentioned in the scientific discovery literature by Langley, Simon, Bradshaw, and Zytkow [2]. I illustrate the task by a simple example. [1]

Let us suppose that experiments have shown that the following reaction among particles is observed to occur:

$$\overline{\pi} + p \rightarrow \pi^0 + n.$$

Furthermore, this second reaction

$$p \nrightarrow \pi + \pi^0$$

has *never* been observed (this is implied by the symbol $\nrightarrow$), despite much experimental effort, and despite the fact that the reaction is not ruled out by existing theory. These circumstances raise a quandary which calls out for resolution.

An adequate resolution lies in postulating a new conserved property that has the value of unity for the particles p and n and zero for the other particles. This property is conserved by the *observed* reaction and violated by the *unobserved* one, which explains why the unobserved reaction never occurs. Historically, particle physicists have faced similar puzzles involving more numerous reactions and have resolved them in such a partially data-driven manner by postulating conserved particle properties [3].

We can generalize the previous simple example into some general concepts of particle physics. First, a *conservation law* states that some aggregate quantity is conserved by a stated physical process. A *phenomenological conservation law* is one that is not discovered or justified on theoretical grounds, but instead serves as a rather ad-hoc explanation of observations; phenomenological reasoning in physics corresponds roughly to the data-driven reasoning of AI terminology. Conservation laws in particle physics are examples of *selection rules*: they select which hypothetical reactions cannot occur because they violate conservation; not all selection rules need be conservation laws.

[1] This example, and some other parts of this paper, are taken from previous publications in order to make the exposition self-contained.

Finally, the quantities that these conservation laws conserve are *quantum numbers*: simple, small numbers that characterize particles and which are not necessarily integers.

3. Prior work

Langley, Simon, Bradshaw, and Zytkow [2] were apparently the first in the AI literature to mention the discovery task of postulating phenomenological conservation laws. These authors pointed out the task's resemblance to the task carried out by their DALTON program in chemistry, implying that the former task was possibly amenable to the heuristic search methods that DALTON used, although no specific approach was given.

Kocabas [1] was the first to describe a program BR-3 capable of performing the task; his BR-3 used some algebraic manipulation to reduce the initial search space, followed by generating specific quantum numbers for particles, testing for contradictions, and backtracking. Given historical data on particle reactions, BR-3 was able to re-discover laws such as the conservation of baryon quantum number, lepton quantum number, and electron and muon numbers. However, Kocabas reported that BR-3 did not find the accepted strangeness quantum numbers when given the reactions that led historically to the discovery of strangeness.

Valdés-Pérez [4] then described the PAULI program which uses a combination of linear programming and backtrack search. PAULI's approach is based on re-representing the particle reactions as two sets of linear algebraic expressions,[2] making use of a well-known technique in chemistry that I employed in another discovery program MECHEM [5] for reasoning about multi-step reaction pathways. PAULI was able to re-discover the strangeness quantum numbers using the historical assumptions and data available to Murray Gell-Mann, the co-discoverer of strangeness [3].

3.1. Discrepancies between BR-3 and PAULI

Kocabas' paper showed how, given the reactions in our Table 1 (Kocabas' Table 3), BR-3 postulates the two accepted conservation laws of baryon and lepton number shown in Table 2. Every observed reaction in Table 1 conserves the sum of baryon numbers; that is, the summed baryon numbers of the reactants equals the corresponding products sum. The same conservation condition holds in the case of lepton numbers. On the other hand, each unobserved reaction in Table 1 violates at least one of the two laws. For example, the reaction $p \rightarrow \pi + \gamma$ violates baryon number conservation, but not lepton conservation.

On the same reactions data from Table 1, PAULI finds that one conservation law is enough to account for the observations; its quantum numbers also appear in Table 2. Only the three particles p, n, and K^0 receive unit quantum numbers, whereas the

[2] A reaction $A + B \rightarrow C + D$ that conserves some quantity Q implies the equation $a_Q + b_Q = c_Q + d_Q$ whose terms are variables rather than particles. Similarly, if the reaction fails to conserve Q, this implies the *inequation* $a_Q + b_Q \neq c_Q + d_Q$.

Table 1
Reactions giving rise to the baryon and lepton conservation laws

Observed reactions			Unobserved reactions		
$p+p$	$\rightarrow$	$p+p+\pi^0$	p	$\rightarrow$	$\bar{e}+\gamma$
$p+p$	$\rightarrow$	$p+\pi+n$	p	$\rightarrow$	$\pi+\pi^0$
$p+\pi$	$\rightarrow$	$\pi+p$	p	$\rightarrow$	$\pi+\gamma$
$\bar{\pi}+p$	$\rightarrow$	$\bar{\pi}+p$	p	$\rightarrow$	$\pi+\pi+\bar{\pi}+\pi^0+\pi^0$
$\bar{\pi}+p$	$\rightarrow$	π^0+n	$p+p$	$\rightarrow$	$\bar{\Lambda}+\bar{\Lambda}$
$\bar{\pi}+p$	$\rightarrow$	$p+\pi+\bar{\pi}+\bar{\pi}$			
$\gamma+e$	$\rightarrow$	$\gamma+e$			
$e+p$	$\rightarrow$	$e+p$			
π^0	$\rightarrow$	$\gamma+\gamma$			
$\bar{\pi}$	$\rightarrow$	$\mu+\bar{\nu}_\mu$			
π	$\rightarrow$	$\bar{\mu}+\nu_\mu$			
μ	$\rightarrow$	$e+\nu_\mu+\bar{\nu}_e$			
n	$\rightarrow$	$p+e+\bar{\nu}_e$			
$\bar{\pi}+p$	$\rightarrow$	$\Lambda+K^0$			

Table 2
Quantum numbers for the particles

Particle	Baryon number	Lepton number	PAULI's number	Particle	Baryon number	Lepton number	PAULI's number
p	1	0	1	$\bar{\nu}$	0	−1	0
n	1	0	1	μ	0	1	0
e	0	1	0	$\bar{\mu}$	0	−1	0
$\bar{e}$	0	−1	0	γ	0	0	0
Λ	1	0	0	π	0	0	0
$\bar{\Lambda}$	−1	0	0	$\bar{\pi}$	0	0	0
K^0	0	0	1	π_0	0	0	0
ν	0	1	0				

other particles are assigned zero. As is required of any solution, the observed reactions conserve PAULI's quantum number, whereas the unobserved ones violate it. Note that the program's quantum number is not a simple sum of the baryon and lepton numbers found by BR-3.

To explain its input data, PAULI prefers fewer conservation laws, ideally a single law (unless there are no unobserved reactions, in which case no "selection rules" are needed, since there is nothing to select *against*). This is the primary criterion of simplicity in the program. Given competing explanations of the same data, e.g., two alternative conservation laws, PAULI prefers the law that involves the smaller sum of the absolute values of quantum numbers; this preference expresses the program's secondary simplicity criterion.

4. Some puzzles, and an explanation

On the same input data, PAULI consistently found simpler (one-conservation-law) solutions than did BR-3, which was puzzling, since BR-3's achievements were rediscoveries of accepted results in particle physics. Our previous explanation for this

puzzle [4] consisted of three alternatives:

(1) Physicists erred by proposing unnecessarily complex assignments of quantum numbers.
(2) Physicists used further constraints to postulate the new quantum numbers.
(3) PAULI's simplicity criteria (inductive bias) are different from (and inferior to) the criteria used by physicists.

The second of these alternatives was correct in the case of PAULI's surprising conclusions about strangeness. That is, on the same strangeness input data as were used historically, consisting of (1) observed particle reactions, and (2) unobserved particle reactions, PAULI's solution was at first simpler than the solution accepted in physics. However, Ne'eman and Kirsh [3] mention a further constraint (the nucleon and pion families possess zero strangeness) that was not mentioned in the textbook (Omnes, 1971) cited by Kocabas, but was assumed by Murray Gell-Mann for his discovery of the strangeness conservation law in 1953. When this constraint was incorporated into PAULI for the strangeness case, the program did find the accepted values of strangeness for the particles involved in the input reactions. This solution to the puzzle *in the case of strangeness* involved the second alternative above (i.e., physicists had used additional constraints), and suggested that perhaps the entire puzzle of why PAULI persisted in finding simpler solutions could be solved by recourse to the same explanation, of an omission of analogous constraints. Surprisingly, this suggestion turned out to be wrong in the general case.

To gain further insight into this mystery, Kocabas and this author each ran his program on other inputs besides those in Kocabas' original paper; PAULI invariably found that the simplest solution involved only one conservation law, despite the fact that BR-3 would find multiple ones, and despite the fact that particle physicists had also postulated several conservation laws. From these observations of PAULI's invariant behavior, I conjectured that, on *any* consistent[3] reaction data *whatsoever*, one conservation law was provably sufficient to rule out the unobserved particle reactions and rule in the observed reactions. I turned to a colleague (Michael Erdmann) for help in proving this theorem, which he carried out by building on the matrix algebraic representations used to design PAULI.

Theorem. *For any set O of observed reactions and set U of unobserved reactions, at most one quantum number conservation law suffices to rule out U and rule in O. That is, there exists a numerical assignment to each particle appearing in the reactions such that every reaction in O conserves this number via summation, and every reaction in U fails to conserve it.*

We then reported these results to a physics audience [9] as three contributions: (1) an automation of the discovery task based on simple principles; (2) a systematic derivation

[3] Consistency means that the observed and unobserved reaction sets can in principle be distinguished using selection rules of conservation. If some unobserved reaction U is linearly dependent on the observed reactions O, then the dataset is inconsistent, because if every reaction in O conserves a quantum number, then U must also.

of the strangeness quantum numbers using historically accurate data and assumptions; and (3) the cited theorem on the parsimony of phenomenological conservation laws (i.e., one law is enough for all conceivable, consistent experimental data).

Section 5 resumes the discussion from the viewpoint of research on scientific discovery. I now proceed to explore some of the implications of this theorem, which is not without interest for those concerned with the philosophy and practice of induction in scientific inference.

4.1. *One law versus multiple laws*

For any given input data, the single all-encompassing conservation law found by PAULI is not in general logically equivalent to the alternative multiple conservation laws. That is, although both theories explain the reactions data, they make discrepant predictions about *unseen* data. In general, each theory prohibits some reactions that the other theory allows, although one expects that the multiple conservation laws will be more restrictive, since each law serves as an independent constraint on the possible reactions.

For example, Table 2 above showed two theories for the reactions of Table 1: the baryon/lepton numbers accepted by physicists and PAULI's numbers. As an experiment, I formed all 1575 possible reactions of the form $A \rightarrow B + C$ that involve the given particles. Of these reactions, the baryon/lepton theory prohibits 1325 reactions and accepts 250, while PAULI's theory prohibits 675 and allows 900 (as expected, the dual-conservation-law theory is more stringent). There are 168 reactions that are allowed by both theories, 593 reactions that are prohibited by both, and 814 reactions (more than half) on which the two theories disagree. Two examples of these 814 discordant reactions are: $p \rightarrow e + n$ is prohibited by the baryon/lepton theory, but not by PAULI's theory, and $K^0 \rightarrow \gamma + \pi$ is prohibited by PAULI, but not by baryon/lepton.

In brief, one law and multiple law theories can make conflicting predictions on unseen data; they are not generally equivalent.

4.2. *A second puzzle and its resolution*

The beginning of Section 4 mentioned one puzzle: PAULI consistently yielded single-conservation-law solutions where BR-3 and particle physics practice yielded solutions based on multiple conservation laws. This puzzle was resolved by discovering that single law solutions were to be expected *mathematically*, so that PAULI, which is based on a systematic, mathematical formulation of the search space, should not find anything other than single laws.

However, this mathematical resolution of the puzzle immediately raised a second puzzle which the closing statement in our physics article [9] expressed thus: "It might be worthwhile to reconcile this theorem with the multiplicity of phenomenological quantum properties". In other words, what criteria could lead, under a purely data-driven regime, to a *justification* of the multiple conservation laws found in particle physics phenomenology?

This latter question motivated a follow-up article [7] that examined and rejected two such criteria: *optimism* (all unseen reactions can occur) and *pessimism* (all unseen reactions are impossible) because they are easily shown to fail to provide the needed justification. A third, *minimax* criterion, according to which one seeks to *minimize* the *maximum* quantum number, does lead to a theoretical justification, since a single conservation law might involve, for example, a maximum quantum number of 3, whereas multiple laws might lead to a maximum quantum number of 2. However, even though minimax is a powerful concept in optimization, a justification of physics laws based on it seems somewhat esoteric and *ad hoc*.

The fourth and final justification examined in the cited follow-up article leads to the seemingly most satisfactory resolution, and is based on the following preliminary observation: If the number of observed, linearly independent reactions equals or exceeds the number of particles, then elementary matrix algebra indicates that *no* conservation law can exist, that is, no law can rule in the observed reactions and rule out the unobserved ones.[4] In such a case, one must either seek alternative selection rules not based on conservation of a quantum number, or perform *divide-and-conquer*.

The divide-and-conquer approach implies dividing all the observed reactions into two or more groups which need not be disjoint, such that within each group, the number of observed reactions is less than the number of particles. One then finds a conservation law for each group separately. Since no single law will cover all the reactions, the result is that every group's conservation law will find exceptions among some reactions outside that group. This is precisely the situation in particle physics practice, in which some reactions fail to conserve one quantum number while conserving all the others.

The resolution of this second puzzle, then, is that one can show, via our machine discovery aided theorem and some further analysis, that the multiple conservation laws of particle physics phenomenology are *mathematically necessary* whenever the observed reactions become numerous relative to the number of particles. Curiously, this constitutes a top-down justification of the state reached in a partly bottom-up manner by physics phenomenology.

5. Generic views of the discovery task

Most research on scientific discovery has been concerned with the relation of discovery processes to *general* problem solving [2]. Our recent research, in contrast, has been overtly preoccupied with the relation of a given discovery task to problem solving *in science*. That is, we seek generalization *not* within general problem solving, *but within science* [8]. We proceed thus not from lack of ambition, but in the belief that a body of computationally oriented theory about science will yield different, and perhaps crisper, results than will an analogous theory about much broader phenomena. These research goals are explicated elsewhere under the organizing concept of *generic task*

[4] Because a matrix R of observed reactions will be nonsingular and invertible, so that the only solution to the equation $RP = 0$, where P is the matrix of particle quantum numbers, will be $P = 0$, which means that no unobserved reaction can fail to conserve the quantum number.

of scientific discovery [6]. Previously, Valdés-Pérez, Zytkow, and Simon [10] showed that the current discovery task lies within a formally defined generic category that they called *scientific model building as search in matrix spaces*. Here I will analyze briefly some broader, but less formal, generic aspects of the discovery task.

PAULI addresses the task of providing a theoretical basis to distinguish the possible reactions that particles undergo from the impossible reactions. In an abstract sense, this task is common throughout science: to find or postulate a feature that distinguishes two classes of objects. For example, a developmental biologist searches for developmental characteristics that distinguish a mutant organism from the wild-type (normal) organism. A psychiatrist looks for eye-movement patterns that discriminate between psychotics and normal subjects. A physiologist seeks features to classify diseased and healthy cells. All of these tasks are somewhat analogous to the concept learning task in machine learning. However, all these scientific tasks emphasize *finding* or *postulating* discriminatory features rather than selecting from a known list.

As in concept learning, a correct but uninteresting theoretical explanation of the observed and unobserved reactions consists of a trivial disjunction of the observed reactions, meaning that all other reactions are prohibited. A more interesting and useful partial explanation was attempted by the physicist Abraham Pais, who pointed out that, for the reactions that led to the discovery of strangeness, particles are generally produced in pairs and never as single particles [3]. This observational pattern of *associated production* could serve as a partial discriminating feature between possible and impossible reactions. However, particle .physicists eventually followed a more theoretical and complete approach by postulating unseen, conserved properties (quantum numbers) such that any hypothetical reaction that violated conservation was deemed impossible. This approach is, of course, the one followed by BR-3 and PAULI.

So, although the task of inventing features to discriminate between two classes is broadly generic throughout science, the detailed solution strategies may be highly particular, as in the approach followed by particle physics in this case.

6. Lessons

The lesson of this paper for artificial intelligence is that machine discovery based on heuristic search does lead to new findings in science, even in sciences of such celebrity and theory density as particle physics. In this instance, the new finding is a theorem which *was enabled by machine discovery* in the sense that observations of the invariant behavior of a machine discovery program directly led to the theorem's conjecture; its proof used representational techniques borrowed from the design of the program itself. Follow-up work then addressed—and resolved—the question of why physics practice seemingly conflicts with the theorem.

It is important to document such results for AI audiences, in order to falsify the premise of an otherwise powerful argument: a dearth of machine discoveries implies that machine discovery research is not addressing the essential parts of the discovery process in science.

Acknowledgements

The author thanks Sakir Kocabas for discussions about BR-3 and about the comparative outputs of BR-3 and PAULI. Yuval Ne'eman guided us through the "particle zoo" and offered encouragement. Michael Erdmann carried out the formal proof of the parsimony theorem. Clark Glymour, Peter Jansen, Aurora Pérez, and Herbert Simon commented valuably on drafts of this note, as did the anonymous reviewers.

This work was supported by a grant #IRI-9421656 from the (USA) National Science Foundation, by the Center for Light Microscope Imaging and Biotechnology, and by the W.M. Keck Center for Advanced Training in Computational Biology.

References

[1] S. Kocabas, Conflict resolution as discovery in particle physics, *Mach. Learn.* **6** (1991) 277–309.
[2] P. Langley, H.A. Simon, G. Bradshaw and J.M. Zytkow, *Scientific Discovery: Computational Explorations of the Creative Processes* (MIT Press, Cambridge, MA, 1987).
[3] Y. Ne'eman and Y. Kirsh, *The Particle Hunters* (Cambridge University Press, New York, 1986).
[4] R.E. Valdés-Pérez, Algebraic reasoning about reactions: discovery of conserved properties in particle physics, *Mach. Learn.* **17** (1994) 47–68.
[5] R.E. Valdés-Pérez, Conjecturing hidden entities via simplicity and conservation laws: machine discovery in chemistry, *Artif. Intell.* **65** (1994) 247–280.
[6] R.E. Valdés-Pérez, Generic tasks of scientific discovery, in: *Working Notes of the Spring Symposium on Systematic Methods of Scientific Discovery*, AAAI Technical Reports (1995).
[7] R.E. Valdés-Pérez, On the justification of multiple selection rules of conservation in particle physics phenomenology, *Comput. Phys. Commun.* **94** (1996) 25–30.
[8] R.E. Valdés-Pérez, Computer science research on scientific discovery, *Knowledge Eng. Rev.* (to appear).
[9] R.E. Valdés-Pérez and M. Erdmann, Systematic induction and parsimony of phenomenological conservation laws, *Comput. Phys. Commun.* **83** (1994) 171–180.
[10] R.E. Valdés-Pérez, J.M. Zytkow and H.A. Simon, Scientific model-building as search in matrix spaces, in: *Proceedings AAAI-93*, Washington, DC (1993) 472–478.

ELSEVIER

Artificial Intelligence 82 (1996) 341–352

Artificial Intelligence

Research Note

A test battery for rational database updating

Sven O. Hansson *

Department of Philosophy, Uppsala University, Villavägen 5, S-752 36 Uppsala, Sweden

Received April 1995; revised October 1995

Abstract

A test battery is proposed that can be used to characterize the behaviour of updating systems. The battery is based on 20 rationality postulates of the types developed in studies of theory change and nonmonotonic reasoning. These postulates are arranged into seven sequences with increasing strength, each of which measures the degree of conformity with a basic desideratum for database updating. Some of these desiderata are partly in conflict, and they cannot all be satisfied to the highest degree. Mathematical characterizations are given of how far they can be combined.

The recent rapid development in the areas of belief revision and nonmonotonic reasoning has led to the introduction of a large number of rationality postulates for belief revision and database updating [1,5,6,10,12,13,15,16,18,21]. Some of these postulates can be used to characterize the behaviour of actual updating systems. For that purpose, the more operative of the postulates must be selected and systematized, and some of them need to be reformulated and adjusted. This is the task undertaken in the present paper.

In Section 1, a framework for the characterization of updating systems is proposed. In Sections 2 and 3, the test battery is outlined, and in Section 4 the interrelations between the various criteria are investigated. Some concluding remarks are offered in Section 5.

1. A conceptual framework

In an influential 1982 paper, Allen Newell postulated the existence in Artificial Intelligence of a *knowledge level* [20]. In the traditional hierarchy of system levels, beginning with the device level and the circuit level, the knowledge level is positioned immediately

* E-mail: filsoh@strix.udac.uu.se.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0004-3702(95)00127-1

above the symbol level (program level). The knowledge level is specified entirely in terms of the contents of the knowledge (beliefs). There is no distinction on this level between information that is explicitly available and information that is implied by available information [4]. It should be possible to predict and understand what takes place on the knowledge level without referring to the symbol level, in much the same way as the symbol level should allow for prediction and understanding without reference to the lower levels of the system.

The framework to be presented here is an application of knowledge level analysis. The contents of a database (before or after update) will be described on the knowledge level, i.e., in terms of what is implied by the information that the system can extract from it. Let us assume that we communicate with a database. Our communications are of two types: inputs (instructions) intended to change the state of the database and retrievals (question–answer dialogues) through which we find out whether or not a particular piece of information is supported by the database. The state of the system does not change between the inputs. By performing repeated retrievals before and after an update we can collect the information needed to characterize the effects of an update on a database.

In order to abstract from the particular subject-matter, all information will be assumed to be represented by sentences in sentential logic. The contents of the database can then be characterized by a set containing all those sentences that it supports. This set is assumed to be closed under logical consequence. Hence, if a sentence follows logically from sentences that are supported by the database, then it is itself supported by the database. This idealizing assumption makes the framework conformable to models used in studies of belief revision and nonmonotonic inference.

The two closely related formal frameworks of belief revision and nonmonotonic reasoning are both suitable for describing the behaviour of a database. In models of belief revision, the belief state of a person (or the state of a database) is represented by a set $\boldsymbol{K}$ of sentences, the "corpus" [17] or "belief set" [1]. $\boldsymbol{K}$ is closed under logical consequence. In other words, $\boldsymbol{K} = \mathrm{Cn}(\boldsymbol{K})$, where Cn is a consequence operator. (For every set A, $\mathrm{Cn}(A)$ is the set of logical consequences of A.) Revision, viz. belief-contravening incorporation of new information, is denoted $*$. Thus, $\boldsymbol{K} * \alpha$ is the outcome of revising $\boldsymbol{K}$ to include the sentence α. Contraction, i.e. the removal of a previous belief, is denoted $\div$. $\boldsymbol{K} \div \alpha$ is the outcome of removing α from $\boldsymbol{K}$.

In nonmonotonic reasoning (contrary to deduction), the receipt of new information may lead to the cancellation of old information. One way to study models of nonmonotonic reasoning in a systematic way is to focus on the inference relation that such a model gives rise to [6]. This relation is denoted by the symbol $\mid\!\sim$. $\alpha \mid\!\sim \beta$ means that the sentence β is inferred from the sentence α. Thus, in our framework, $\alpha \mid\!\sim \beta$ means that after receiving the input information α, the database supports β.

There is a close connection between nonmonotonic reasoning and belief revision [19]. Let $\boldsymbol{K}$ be a belief set and $*$ an operator of revision for $\boldsymbol{K}$. Then $\beta \in \boldsymbol{K} * \alpha$ means that from the new information α we infer β, given the set $\boldsymbol{K}$ of background beliefs. This is another way of saying that $\alpha \mid\!\sim \beta$, where $\mid\!\sim$ denotes inferences according to $\boldsymbol{K}$. If we want to be more precise, we can write $\alpha \mid\!\sim_{\boldsymbol{K}} \beta$. Given a fixed set $\boldsymbol{K}$ of background beliefs, $\beta \in \boldsymbol{K} * \alpha$ and $\alpha \mid\!\sim_{\boldsymbol{K}} \beta$ are just alternative notations. Using this relationship,

rationality postulates for nonmonotonic reasoning can be translated into postulates for belief change, and vice versa.

The belief change framework is somewhat more versatile than that of nonmonotonic inference relations. It allows for accumulated (repeated) updating, such as $\boldsymbol{K} * \alpha * \beta$, and for other types of change than revision, in particular contraction ($\boldsymbol{K} \div \alpha$). The framework of nonmonotonic reasoning can be seen as a restriction of the belief revision framework to single incorporations of new beliefs.

For the purposes of the present paper, the language of belief revision has been chosen. Thus, $\boldsymbol{K}$ denotes the state of the database before update, and $\boldsymbol{K}*\alpha$ its state after the update that results from an instruction to accept α. Postulates that originate in the framework of nonmonotonic reasoning will be translated into the belief revision framework in the way indicated above.

Criteria for rational incorporation of new information can be divided into three main categories. Those criteria that only concern update by one single sentence will be called *elementary*. Criteria that compare updates by different sentences will be called *comparative*. Criteria that refer to combined or repeated changes (such as $\boldsymbol{K} * \alpha * \beta$) are called *compositive* criteria.

Elementary criteria will be treated in Section 3 and comparative criteria in Section 4. Compositive criteria will not be treated here, since the few results on combined (repeated) change that are as yet available from the belief revision literature provide insufficient guidance. Much remains to be done before solid rationality criteria for combined operations of belief change or database updating are available.

2. Elementary criteria

After an update, we want the new state of the database to be a consistent state that supports the new information. Furthermore, we want the change to be conservative, i.e., unnecessary differences between the old and the new state should be avoided. No previous information should unnecessarily be lost, and no new information should unnecessarily be affixed. This adds up to four basic desiderata for database updating:

(1) The new information should be accepted (*Success*).
(2) The new state should be consistent (*Consistency*).
(3) Unwarranted losses of old information should be avoided (*Preservation*).
(4) Unwarranted addition of new information should be avoided (*Repellence*).

The strongest possible version of *Success* is $\alpha \in \boldsymbol{K} * \alpha$. However, this may be a too strong requirement since it forces us into inconsistency if α itself is inconsistent. The very least that we can require is that if α is already supported by a consistent database, then an update by α does not lead to the loss of this support. In the following series of four success postulates, each is stronger than the foregoing:

S1. If $\alpha \in \boldsymbol{K}$ and $\neg\alpha \notin \boldsymbol{K}$, then $\alpha \in \boldsymbol{K} * \alpha$.
S2. If $\neg\alpha \notin \boldsymbol{K}$, then $\alpha \in \boldsymbol{K} * \alpha$.
S3. If $\neg\alpha \notin \mathrm{Cn}(\emptyset)$, then $\alpha \in \boldsymbol{K} * \alpha$.
S4. $\alpha \in \boldsymbol{K} * \alpha$.

S3 has been called *weak success* by Rott [22]. S4 is the *success* postulate commonly referred to in the belief revision literature [8]. The corresponding principle in nonmonotonic reasoning ($\alpha \mid\!\sim \alpha$) has been called *reflexivity*. It is a basic requirement on inference, that "holds of all nonmonotonic inference relations currently in the literature and indeed ... should hold of any relation that deserves the name of 'inference relation'"([19], cf. [16]).

In accounts of non-prioritized belief change (in which the new information is given no special priority due to its novelty), S2 is satisfied, but S3 and S4 are not [14].

The strongest possible version of *Consistency* is "$\boldsymbol{K} * \alpha$ is consistent". It has the following three weaker versions:

C1. If $\alpha \in \boldsymbol{K}$ and $\neg\alpha \notin \boldsymbol{K}$, then $\boldsymbol{K} * \alpha$ is consistent.
C2. If $\neg\alpha \notin \boldsymbol{K}$, then $\boldsymbol{K} * \alpha$ is consistent.
C3. If $\neg\alpha \notin \mathrm{Cn}(\emptyset)$, then $\boldsymbol{K} * \alpha$ is consistent.
C4. $\boldsymbol{K} * \alpha$ is consistent.

In C1, $\boldsymbol{K} * \alpha$ is required to be consistent if $\boldsymbol{K} \cup \{\alpha\}$ is consistent and equal to $\boldsymbol{K}$, in C2, if $\boldsymbol{K} \cup \{\alpha\}$ is consistent, and in C3 if α is consistent. C3 is the conventional *consistency* postulate of the belief revision literature [8]. Its counterpart in nonmonotonic reasoning (if $\alpha \mid\!\sim \bot$ then $\alpha \vdash \bot$) has been called *consistency preservation* [11]. C4 has been called *strong consistency preservation* by Rott [22].

The strongest possible version of *Preservation* is that $\boldsymbol{K}$ is always preserved, $\boldsymbol{K} \subseteq \boldsymbol{K} * \alpha$. The very least that we can require is that if $\boldsymbol{K}$ is consistent, then it is preserved after a vacuous update (update by one of its elements). The following sequence contains two intermediate levels of preservation between these two:

P1. If $\alpha \in \boldsymbol{K}$ and $\neg\alpha \notin \boldsymbol{K}$, then $\boldsymbol{K} \subseteq \boldsymbol{K} * \alpha$.
P2. If $\neg\alpha \notin \boldsymbol{K}$, then $\boldsymbol{K} \subseteq \boldsymbol{K} * \alpha$.
P3. If $\beta \in \boldsymbol{K} \backslash (\boldsymbol{K} * \alpha)$, then $\neg\alpha \in \boldsymbol{K}$ and $\neg\beta \in \boldsymbol{K} * \alpha$.
P4. $\boldsymbol{K} \subseteq \boldsymbol{K} * \alpha$.

P2 is the *preservation* postulate referred to in studies of the relations between belief revision and the logic of conditional sentences [9]. Mostly, belief revisionists instead employ a related postulate:

If $\neg\alpha \notin \boldsymbol{K}$, then $\mathrm{Cn}(\boldsymbol{K} \cup \{\alpha\}) \subseteq \boldsymbol{K} * \alpha$ [7].

This postulate follows from our P2 and S2. In nonmonotonic reasoning, it corresponds to: If $\not\mid\!\sim \neg\alpha$ and $\mid\!\sim \alpha \rightarrow \beta$, then $\alpha \mid\!\sim \beta$, that has been called *weak rational monotony* [11]. P3 is a somewhat strengthened version of the following postulate that is discussed in [10]:

If $\beta \in \boldsymbol{K} \backslash (\boldsymbol{K} * \alpha)$, then $\neg\beta \in \boldsymbol{K} * \alpha$ (*tenacity*).

The choice of the stronger variant for the present purposes has mainly technical reasons. P3, but not tenacity, is intermediate in strength between P2 and P4. Obviously, P4 is too strong to be a desirable property. It represents "too much" of a desideratum.

The strongest possible version of *Repellence* is $\boldsymbol{K} * \alpha \subseteq \boldsymbol{K}$, that requires all new information to be rejected. This is a highly implausible property of an update operation.

Clearly, incorporation of the input sentence must be allowed for at least in some cases. The following are weaker and more plausible postulates of repellence:

R1. If $\alpha \in \boldsymbol{K}$ and $\neg\alpha \notin \boldsymbol{K}$, then $\boldsymbol{K} * \alpha \subseteq \boldsymbol{K}$.
R2. $\boldsymbol{K} * \alpha \subseteq \mathrm{Cn}(\boldsymbol{K} \cup \{\alpha\})$.

R2 is the inclusion postulate of belief revision [8]. Its counterpart in nonmonotonic reasoning (If $\alpha \mid\!\sim \gamma$, then $\mid\!\sim \alpha \rightarrow \gamma$) has been called *weak conditionalization* since it is a special case of *conditionalization* (If $\beta \& \alpha \mid\!\sim \gamma$, then $\beta \mid\!\sim \alpha \rightarrow \gamma$) [11].

3. Comparative criteria

Comparative criteria for updating are concerned with the relationships between updates by different sentences. Such relationships can only be expected if the sentences are somehow related. Through our choice of a formal framework we have abstracted from non-logical relationships between sentences. Therefore, it remains to compare updates by sentences that are logically related.

The strongest such relation is that of equivalence. We should expect logically equivalent sentences to be treated alike in updating:

E1. If $\alpha \leftrightarrow \beta \in \mathrm{Cn}(\emptyset)$, then $\boldsymbol{K} * \alpha = \boldsymbol{K} * \beta$.

This is the *extensionality* postulate of belief revision [8]. In nonmonotonic reasoning, the corresponding property (If $\vdash \alpha \leftrightarrow \beta$, then $\alpha \mid\!\sim \gamma$ if $\beta \mid\!\sim \gamma$) is called *left logical equivalence* [11].

It is less obvious how an updating system should treat two sentences, one of which implies the other. If the updates do not require deletion of old information, then the outcome of update by the stronger sentence should be a superset of the outcome of update by the weaker sentence:

If $\mathrm{Cn}(\{\alpha\}) \subseteq \mathrm{Cn}(\{\beta\})$, then $\boldsymbol{K} * \alpha \subseteq \boldsymbol{K} * \beta$ (*monotonicity*).

If β contradicts $\boldsymbol{K}$, then this is for most purposes a too strong requirement. It can be mitigated in various ways. The following series contains three variants with increasing strength (it is mostly more convenient to express monotonicity conditions in terms of the relation between a conjunction and one of its conjuncts):

M1. If $\beta \in \boldsymbol{K} * \alpha$ and $\neg\beta \notin \boldsymbol{K} * \alpha$, then $\boldsymbol{K} * \alpha \subseteq \boldsymbol{K} * (\alpha \& \beta)$.
M2. If $\neg\beta \notin \boldsymbol{K} * \alpha$, then $\boldsymbol{K} * \alpha \subseteq \boldsymbol{K} * (\alpha \& \beta)$.
M3. $\boldsymbol{K} * \alpha \subseteq \boldsymbol{K} * (\alpha \& \beta)$.

M1 is a weakened version of the following condition on nonmonotonic inference:

If $\beta \in \boldsymbol{K} * \alpha$, then $\boldsymbol{K} * \alpha \subseteq \boldsymbol{K} * (\alpha \& \beta)$ (*cautious monotony* [6,16]).

M2 is closely related to the postulate of *rational monotony* in nonmonotonic reasoning and to its counterpart in belief revision:

If $\neg\beta \notin \boldsymbol{K} * \alpha$, then $\mathrm{Cn}((\boldsymbol{K} * \alpha) \cup \{\beta\}) \subseteq \boldsymbol{K} * (\alpha \& \beta)$ (*Gärdenfors'* $*8$ [8]).

The adjusted versions have been chosen in order to obtain the desired logical relationships in the sequence M1, M2, M3. In most applications, M3 is too strong to be a desirable property.

It would not make much sense to require in general the reversal of monotony, $\boldsymbol{K} * (\alpha \& \beta) \subseteq \boldsymbol{K} * \alpha$. However, under the special condition that $\beta \in \boldsymbol{K} * \alpha$, this condition is sensible. It can also be required that what $\boldsymbol{K} * (\alpha \& \beta)$ contains in addition to $\boldsymbol{K} * \alpha$ should follow from $\boldsymbol{K} * \alpha$ and β. These are *antitonic* relationships, i.e., they are (partial) reversals of monotony:

A1. If $\beta \in \boldsymbol{K} * \alpha$, then $\boldsymbol{K} * (\alpha \& \beta) \subseteq \boldsymbol{K} * \alpha$.
A2. $\boldsymbol{K} * (\alpha \& \beta) \subseteq \mathrm{Cn}((\boldsymbol{K} * \alpha) \cup \{\beta\})$.

In nonmonotonic reasoning, A1 is called *cut* [16] and A2 is called *conditionalization* [11,18]. A2 is also known from the belief revision literature as $*7$, Gärdenfors' seventh revision postulate [8].

4. Maximal satisfaction of the criteria

The elementary properties of an updating system can be summarized in the form of a vector

$$\langle \mathrm{S}x_1, \mathrm{C}x_2, \mathrm{P}x_3, \mathrm{R}x_4 \rangle$$

where $\mathrm{S}x_1$ is the highest S-criterion satisfied by the system. (If none of them is satisfied, then we write S0.) Such a vector will be called an SCPR-vector. Similarly, the properties of a database system with respect to the full set of critera, including the comparative ones, can be summarized in the form of a vector

$$\langle \mathrm{S}x_1, \mathrm{C}x_2, \mathrm{P}x_3, \mathrm{R}x_4, \mathrm{E}x_5, \mathrm{M}x_6, \mathrm{A}x_7 \rangle.$$

This will be called a SCPREMA-vector.

It is not possible for all the criteria of Sections 2 and 3 to be satisfied at the same time, i.e., no updating system can satisfy:

$$\langle \mathrm{S4}, \mathrm{C4}, \mathrm{P4}, \mathrm{R2}, \mathrm{E1}, \mathrm{M3}, \mathrm{A2} \rangle.$$

The formal limits to satisfaction of the criteria can be expressed in terms of maximally satisfied vectors, as follows:

Definition. A SCPR-vector $\langle \mathrm{S}x_1, \mathrm{C}x_2, \mathrm{P}x_3, \mathrm{R}x_4 \rangle$ is *satisfaction-maximal* if and only if
(1) it is possible for an update system to satisfy $\langle \mathrm{S}x_1, \mathrm{C}x_2, \mathrm{P}x_3, \mathrm{R}x_4 \rangle$, and
(2) it is not possible for an update system to satisfy
- $\langle \mathrm{S}x'_1, \mathrm{C}x_2, \mathrm{P}x_3, \mathrm{R}x_4 \rangle$ if $x'_1 > x_1$,
- $\langle \mathrm{S}x_1, \mathrm{C}x'_2, \mathrm{P}x_3, \mathrm{R}x_4 \rangle$ if $x'_2 > x_2$,
- $\langle \mathrm{S}x_1, \mathrm{C}x_2, \mathrm{P}x'_3, \mathrm{R}x_4 \rangle$ if $x'_3 > x_3$, or
- $\langle \mathrm{S}x_1, \mathrm{C}x_2, \mathrm{P}x_3, \mathrm{R}x'_4 \rangle$ if $x'_4 > x_4$.

Satisfaction-maximality of SCPREMA-vectors is defined analogously.

There are only three satisfaction-maximal SCPR-vectors:

Observation 1. The following is a complete list of the satisfaction-maximal SCPR-vectors: $\langle S4, C3, P3, R2\rangle$, $\langle S3, C4, P3, R2\rangle$, and $\langle S4, C2, P4, R2\rangle$.

$\langle S4, C3, P3, R2\rangle$ is satisfied by an updating system that employs AGM-style maxichoice revision. $\langle S3, C4, P3, R2\rangle$ is satisfied by an updating system that employs AGM-style maxichoice revision except when the input (α in $\boldsymbol{K} * \alpha$) is inconsistent, in which case $\boldsymbol{K} * \alpha$ is a maximal consistent subset of $\boldsymbol{K}$. $\langle S4, C2, P4, R2\rangle$ is satisfied by AGM's operation of expansion. For detailed constructions, and a full proof of the observation, the reader is referred to Appendix A.

SCPREMA-vectors containing M3 represent the extreme case of database updating that satisfies monotonicity, i.e., never gives up old information. The operation $*$ such that $\boldsymbol{K} * \alpha = \mathrm{Cn}(\boldsymbol{K} \cup \{\alpha\})$ for all $\boldsymbol{K}$ and α is characterized by the SCPREMA-vector $\langle S4, C2, P4, R2, E1, M3, A2\rangle$, that can be shown to be satisfaction-maximal. The interesting satisfaction-maximal SCPREMA-vectors, though, are those that do not contain M3. It turns out that there are only two such vectors:

Observation 2. The following is a complete list of the satisfaction-maximal SCPREMA-vectors that do not contain M3: $\langle S4, C3, P3, R2, E1, M2, A2\rangle$ and $\langle S3, C4, P3, R2, E1, M2, A2\rangle$.

Hence, the only interesting conflict that remains after M3 has been given up is the obvious one between S4 and C4. $\langle S4, C3, P3, R2, E1, M2, A2\rangle$ is satisifed by AGM's transitively relational maxichoice revision. $\langle S3, C4, P3, R2, E1, M2, A2\rangle$ is satisfied by a modified version of the same, such that if α is inconsistent, then $\boldsymbol{K} * \alpha$ is the highest-ranked maximal consistent subset of $\boldsymbol{K}$. Detailed constructions and a proof of the observation are given in Appendix A.

More generally, it can be seen from Observations 1 and 2 that serious competition only occurs in the S-, C-, P-, and M-dimensions. In the remaining three dimensions, full satisfaction is always possible.

5. Conclusion

SCPREMA-vectors can be used to characterize the properties of updating systems and to compare in a precise manner how the conflicts between incompatible but desirable properties are settled in different such systems. We should not expect actual updating systems to always have satisfaction-maximal SCPREMA-vectors, but any failure of satisfaction-maximality should have an acceptable motivation.

Appendix A. Proofs

$\mathcal{L}$ denotes the language, i.e., the set of all sentences. $^{\perp}$ denotes any contradictory sentence, i.e., any sentence δ such that $\neg\delta \in \mathrm{Cn}(\emptyset)$. Furthermore, the following definition will be used in the proof:

Definition A.1 (*Alchourrón and Makinson* [2]). Let A be a set of sentences and α a sentence. The set $A \perp \alpha$ ("A remainder α") is the set of sets such that $B \in A \perp \alpha$ if and only if:

(i) $B \subseteq A$.
(ii) $\alpha \notin \mathrm{Cn}(B)$.
(iii) There is no set B' such that $B \subset B' \subseteq A$ and $\alpha \notin \mathrm{Cn}(B')$.

Hence, $\boldsymbol{K} \perp \alpha$ is the set of inclusion-maximal subsets of $\boldsymbol{K}$ that do not imply α.

The incompatibilities exhibited in the following lemma will be used several times in the proofs:

Lemma A.2.
(1) C4 *and* S4 *are incompatible.*
(2) P4 *and* C3 *are incompatible.*
(3) M3, S3, C3, *and* P2 *are incompatible.*

Proof. *Part* (1). They cannot both hold if α is inconsistent.

Part (2). They cannot both hold if $\boldsymbol{K}$ is inconsistent.

Part (3). Suppose that M3, S3, C3, and P2 are all satisfied. Let α and β be sentences, and $\boldsymbol{K}$ a set, such that $\neg(\alpha\&\beta) \notin \mathrm{Cn}(\emptyset)$, that $\neg(\alpha\&\beta) \in \boldsymbol{K}$ and $\neg\alpha \notin \boldsymbol{K}$. It follows from P2 that $\boldsymbol{K} \subseteq \boldsymbol{K} * \alpha$ and from M3 that $\boldsymbol{K} * \alpha \subseteq \boldsymbol{K} * (\alpha\&\beta)$. Hence $\boldsymbol{K} \subseteq \boldsymbol{K} * (\alpha\&\beta)$. It follows from S3 that $\alpha\&\beta \in \boldsymbol{K} * (\alpha\&\beta)$. Thus, $\boldsymbol{K} \cup \alpha\&\beta \subseteq \boldsymbol{K} * (\alpha\&\beta)$, so that $\boldsymbol{K} * (\alpha\&\beta)$ is inconsistent, contrary to C3. □

Proof of Observation 1. We need to show (1) that each of the three listed vectors is satisfaction-maximal, and (2) that they are all the satisfaction-maximal satisfiable vectors that there are.

Part (1A): Proof that $\langle \mathrm{S4}, \mathrm{C3}, \mathrm{P3}, \mathrm{R2} \rangle$ is a satisfaction-maximal SCPR-vector. We are first going to show that it is satisfiable. (What follows is essentially the construction of "choice" or "maxichoice" contraction according to [3].)

Let s be a function such that for all logically closed sets $\boldsymbol{K}$ of sentences and all non-tautological sentences β, $s(\boldsymbol{K}, \boldsymbol{K} \perp \beta)$ is an element of $\boldsymbol{K} \perp \beta$. Furthermore, let $*$ be an operation such that:

(I) If $\neg\alpha \notin \mathrm{Cn}(\emptyset)$, then $\boldsymbol{K} * \alpha = \mathrm{Cn}(s(\boldsymbol{K}, \boldsymbol{K} \perp \neg\alpha) \cup \{\alpha\})$.
(II) If $\neg\alpha \in \mathrm{Cn}(\emptyset)$, then $\boldsymbol{K} * \alpha = \mathcal{L}$.

S4: In both (I) and (II), $\alpha \in \boldsymbol{K} * \alpha$.

C3: If $\neg\alpha \notin \mathrm{Cn}(\emptyset)$, then we are in case (I). Since $s(\boldsymbol{K}, \boldsymbol{K} \perp \neg\alpha)$ does not, by definition, imply $\neg\alpha$, $\mathrm{Cn}(s(\boldsymbol{K}, \boldsymbol{K} \perp \neg\alpha) \cup \{\alpha\})$ is consistent.

P3: Let $\beta \in \boldsymbol{K} \backslash (\boldsymbol{K} * \alpha)$. In case (I), it follows from $\boldsymbol{K} \backslash (\boldsymbol{K} * \alpha) \neq \emptyset$ that $s(\boldsymbol{K}, \boldsymbol{K} \perp \neg\alpha) \neq \boldsymbol{K}$, and thus $\neg\alpha \in \mathrm{Cn}(\boldsymbol{K}) = \boldsymbol{K}$. Furthermore, it follows from $\beta \in \boldsymbol{K} \backslash (\boldsymbol{K} * \alpha)$ and $s(\boldsymbol{K}, \boldsymbol{K} \perp \neg\alpha) \in \boldsymbol{K} \perp \neg\alpha$ that $s(\boldsymbol{K}, \boldsymbol{K} \perp \neg\alpha) \cup \{\beta\}$ implies $\neg\alpha$, and thus $s(\boldsymbol{K}, \boldsymbol{K} \perp \neg\alpha) \cup \{\alpha\}$ implies $\neg\beta$, i.e., $\boldsymbol{K} * \alpha$ implies $\neg\beta$. Since $\boldsymbol{K} * \alpha$ is logically closed, this means that $\neg\beta \in \boldsymbol{K} * \alpha$. In case (II), we can conclude from $\neg\alpha \in \mathrm{Cn}(\emptyset)$ that $\neg\alpha \in \boldsymbol{K}$ and from $\boldsymbol{K} * \alpha = \mathcal{L}$ that $\neg\beta \in \boldsymbol{K} * \alpha$.

R2: In case (I) we have $s(\boldsymbol{K}, \boldsymbol{K} \perp \neg\alpha) \subseteq \boldsymbol{K}$, and thus

$$\boldsymbol{K} * \alpha = \mathrm{Cn}(s(\boldsymbol{K}, \boldsymbol{K} \perp \neg\alpha) \cup \{\alpha\}) \subseteq \mathrm{Cn}(\boldsymbol{K} \cup \{\alpha\}).$$

In case (II), it follows from $\neg\alpha \in \mathrm{Cn}(\emptyset)$ that $\mathrm{Cn}(\boldsymbol{K} \cup \{\alpha\}) = \mathcal{L}$, and hence $\boldsymbol{K} * \alpha \subseteq \mathrm{Cn}(\boldsymbol{K} \cup \{\alpha\})$.

Maximality: Since both the S- and the R-component have maximal values, it is sufficient to show that neither the C- nor the P-component can be increased without some other component being decreased. This follows quite straightforwardly from Lemma A.2, according to which (1) C4 and S4 are compatible, and (2) P4 and C3 are incompatible.

Part (1B): Proof that $\langle \mathrm{S3}, \mathrm{C4}, \mathrm{P3}, \mathrm{R2} \rangle$ is a satisfaction-maximal SCPR-vector. We are first going to show that it is satisfiable. Just as in Part (1A), let s be a function such that for all $\boldsymbol{K}$ and all non-tautological β, $s(\boldsymbol{K}, \boldsymbol{K} \perp \beta)$ is an element of $\boldsymbol{K} \perp \beta$. Furthermore, let $*$ be an operation such that:

(I) If $\neg\alpha \notin \mathrm{Cn}(\emptyset)$, then $\boldsymbol{K} * \alpha = \mathrm{Cn}(s(\boldsymbol{K}, \boldsymbol{K} \perp \neg\alpha) \cup \{\alpha\})$.
(II) If $\neg\alpha \in \mathrm{Cn}(\emptyset)$, then $\boldsymbol{K} * \alpha = s(\boldsymbol{K}, \boldsymbol{K} \perp^{\perp})$.

S3: Directly from the definition (case (I)).

C4: Directly from the definition. Note in case (I) that $s(\boldsymbol{K}, \boldsymbol{K} \perp \neg\alpha)$ is an element of $\boldsymbol{K} \perp \neg\alpha$, and thus does not imply $\neg\alpha$.

P3: In case (I), it follows in the same way as in Part (1A) of the present proof that $\neg\beta \in \boldsymbol{K} * \alpha$. In case (II), if $\boldsymbol{K} \backslash (\boldsymbol{K} * \alpha)$ is non-empty, then $s(\boldsymbol{K}, \boldsymbol{K} \perp^{\perp}) \neq \boldsymbol{K}$, and thus $\boldsymbol{K}$ is inconsistent, i.e., $\boldsymbol{K} = \mathcal{L}$. It follows directly from this that $\neg\alpha \in \boldsymbol{K}$. Furthermore, it follows from $\boldsymbol{K} * \alpha \in \boldsymbol{K} \perp^{\perp} = \mathcal{L} \perp^{\perp}$ that $\boldsymbol{K} * \alpha$ is a maximal consistent subset of the language; hence it follows from $\beta \notin \boldsymbol{K} * \alpha$ that $\neg\beta \in \boldsymbol{K} * \alpha$.

R2: As in Part (1A) of the present proof.

Maximality: Since both the C-component and the R-component have maximal values, it is sufficient to show that neither the S- nor the P-component can be increased without a decrease in some other component. This follows directly from the two observations, given in Lemma A.2, that (1) S4 and C4 are incompatible and (2) P4 and C4 are incompatible.

Part (1C): Proof that $\langle \mathrm{S4}, \mathrm{C2}, \mathrm{P4}, \mathrm{R2} \rangle$ is a satisfaction-maximal SCPR-vector. We are first going to show that it is satisfiable. Let $\boldsymbol{K} * \alpha = \mathrm{Cn}(\boldsymbol{K} \cup \{\alpha\})$ for all $\boldsymbol{K}$ and α. It follows directly from the definition that S4, C2, P4, and R2 are all satisfied.

Maximality: Since the S-, P-, and R- components all have maximal values, it is sufficient to show that the value of the C-component cannot be increased without a decrease in the value of some other component. This follows directly from the incompatibility of C3 and P4 that was shown in Lemma A.2.

Part (2). It remains to prove that the three listed vectors are all the satisfaction-maximal SCPR-vectors that there are.

Since $\langle \mathrm{S4}, \mathrm{C2}, \mathrm{P4}, \mathrm{R2} \rangle$ has maximal values in all components except the C-component, no satisfaction-maximal vector except this one can have a lower C-value than 3. With this we can restrict our search to vectors whose C-value is 3 or 4.

Since C3 and P4 are incompatible (Lemma A.2, Part (2)), no satisfiable vector with the C-value 3 can contain P4. Since S4 and R2 represent maximal values in their

respective dimensions, it follows that any satisfiable vector with the C-value 3 must be either identical to or beaten by $\langle$S4, C3, P3, R2$\rangle$.

Similarly, since C4 is not compatible with either S4 or P4 (Lemma A.2, Parts (1) and (2)), no satisfiable vector with the C-value 4 can have an S- or a P-value that exceeds 3. Since R2 represents a maximal value, we can conclude that all satisfiable vectors containing C4 are either identical to $\langle$S3, C4, P3, R2$\rangle$ or beaten by it. This concludes the proof. □

The following lemmas will be used in the proof of Observation 2:

Lemma A.3 (Alchourrón et al. [1]). *Let A be a logically closed set. If $X \in A \perp \alpha$, then $X \in A \perp \beta$ for all $\beta \in A \backslash X$.*

For a proof, the reader is referred to [1].

Lemma A.4. *Let A be a logically closed set. If $\alpha \in A$ and $\alpha \rightarrow \beta \in \mathrm{Cn}(\emptyset)$, then $A \perp \beta \subseteq A \perp \alpha$.*

Proof. Let $X \in A \perp \beta$. Then $\alpha \notin X$, and since $\alpha \in A$ it follows from Lemma A.3 that $X \in A \perp \alpha$. □

Proof of Observation 2. We need to show (1) that the two listed vectors are satisfaction-maximal SCPREMA-vectors, and (2) that all other satisfaction-maximal SCPREMA-vectors have the M-value 3, i.e., they represent satisfaction of full monotonicity.

Part (1A): Proof that $\langle$S4, C3, P3, R2, E1, M2, A2$\rangle$ is satisfaction-maximal. We first need to show that it is satisfiable. (What follows is essentially the construction of "orderly choice contraction" in [3], or transitively relational maxichoice contraction in the terminology of [1].) Let $<$ be a strict ordering on $\mathcal{U}_K = \{X \mid X \in K \perp \alpha$ for some $\alpha\}$. For each non-empty subset X of $\mathcal{U}_K$, let $s(X)$ be the element of X that is ranked highest according to $<$. Let $*$ be the operation such that:

(I) If $\neg\alpha \notin \mathrm{Cn}(\emptyset)$, then $K * \alpha = \mathrm{Cn}(s(K \perp \neg\alpha) \cup \{\alpha\})$.
(II) If $\neg\alpha \in \mathrm{Cn}(\emptyset)$, then $K * \alpha = \mathcal{L}$.

It follows from Part (1A) of the proof of Observation 1 that S4, C3, P3, and R2 are satisfied. That E1 is satisfied follows directly from the definition.

M2: Let $\neg\beta \notin K * \alpha$. Then, according to (II), $\neg\alpha \notin \mathrm{Cn}(\emptyset)$, and we are in case (I). Then $\neg\beta \notin K * \alpha = \mathrm{Cn}(s(K \perp \neg\alpha) \cup \{\alpha\})$ yields $\alpha \rightarrow \neg\beta \notin s(K \perp \neg\alpha)$, i.e., $\neg(\alpha\&\beta) \notin s(K \perp \neg\alpha)$. It follows from Lemma A.3 that $s(K \perp \neg\alpha) \in K \perp \neg(\alpha\&\beta)$ and from Lemma A.4 that $K \perp \neg(\alpha\&\beta) \subseteq K \perp \neg\alpha$. Since s is based on $<$, we can conclude that $s(K \perp \neg(\alpha\&\beta)) = s(K \perp \neg\alpha)$. It follows directly from (I) that $K * \alpha \subseteq K * (\alpha\&\beta)$.

A2: This is obvious if $\neg\beta \in K * \alpha$. If $\neg\beta \notin K * \alpha$, then (II) yields $\neg\alpha \notin \mathrm{Cn}(\emptyset)$. We then have (as was shown in the proof for M2) $s(K \perp \neg\alpha) = s(K \perp \neg(\alpha\&\beta))$, $K * \alpha = \mathrm{Cn}(s(K \perp \neg\alpha) \cup \{\alpha\})$ and $K * (\alpha\&\beta) = \mathrm{Cn}(s(K \perp \neg(\alpha\&\beta)) \cup \{\alpha\&\beta\})$. From this it follows that A2 holds.

Maximality: Since all components except the C-, P- and M-components have their maximal values, it is sufficient to show that neither the C-, the P-, nor the M-component can be increased without some other component being decreased. That this holds for the C- and P-components can be proved exactly as in Part (1A) of the proof of Observation 1. To show that this holds for the M-component, it is sufficient to note that according to Lemma A.2, M3, S3, C3, and P2 are incompatible, and hence so are M3, S4, C3, and P3.

Part (1B): Proof that $\langle S3, C4, P3, R2, E1, M2, A2\rangle$ is satisfaction-maximal. We first need to show that it is satisfiable. Let $<$ be a strict ordering on $\mathcal{U}_K$. For each non-empty subset X of $\mathcal{U}_K$, let $s(X)$ be the element of X that is ranked highest according to $<$. Let $*$ be the operation such that:

(I) If $\neg\alpha \notin \mathrm{Cn}(\emptyset)$, then $K * \alpha = \mathrm{Cn}(s(K \perp \neg\alpha) \cup \{\alpha\})$.
(II) If $\neg\alpha \in \mathrm{Cn}(\emptyset)$, then $K * \alpha = s(K \perp^{\perp})$.

It follows from Part (1B) of the proof of Observation 1 that S3, C4, P3, and R2 are satisfied. That E1 is satisfied follows directly from the definition.

The proofs for M2 and A2 coincide with the corresponding parts of Part (1A) of the present proof, except in the case when $\neg\alpha \in \mathrm{Cn}(\emptyset)$. In that case, we also have $\neg(\alpha \& \beta) \in \mathrm{Cn}(\emptyset)$. It follows that $K * \alpha = K * (\alpha \& \beta) = s(K \perp^{\perp})$, from which M2 and A2 follow directly.

Maximality: Since only the S-, P-, and M-components are non-maximal, it is sufficient to show that none of these can be increased without an increase in some other component. We know from Lemma A.2 that each of S4 and P4 is incompatible with C4. Furthermore, we know from the same lemma that M3, S3, C3, and P2 are incompatible, and hence so are M3, S3, C4, and P3.

Part (2). Our search for satisfaction-maximal vectors with lower M-values than 3 can be divided into cases according to the value of the C-component.

First case, the C-component is C4: according to Lemma A.2, S4 and C4 are incompatible, and so are P4 and C4. Thus, any vector with C4 has at most S3 and at most P3. Hence, if it does not have M3 it must be either identical to or beaten by $\langle S3, C4, P3, R2, E1, M2, A2\rangle$.

Second case, the C-component is C3: according to Lemma A.2, P4 and C3 are incompatible. Thus, any vector with C3 has at most P3. Hence if it does not contain M3 it must be either identical to or beaten by $\langle S4, C3, P3, R2, E1, M2, A2\rangle$.

Third case, the C-component is at most C2: the operation referred to in Part (1C) of the proof of Observation 1 has the SCPREMA-vector $\langle S4, C2, P4, R2, E1, M3, A2\rangle$. Since this vector is maximal in all components except the C-component, no satisfaction-maximal vector can have a C-component lower than C3 without containing M3. With this the proof is completed. □

References

[1] C.E. Alchourrón, P. Gärdenfors and D. Makinson, On the logic of theory change: partial meet contraction and revision functions, *J. Symbolic Logic* **50** (1985) 510–530.

[2] C.E. Alchourrón and D. Makinson, Hierarchies of regulation and their logic, in: Hilpinen, ed., *New Studies in Deontic Logic* (Reidel, Dordrecht, Netherlands, 1981) 125–148.
[3] C.E. Alchourrón and D. Makinson, On the logic of theory change: contraction functions and their associated revision functions, *Theoria* **48** (1982) 14–37.
[4] R.J. Brachman and H.J. Levesque, What makes a knowledge base knowledgeable? A view of databases from the knowledge level, in: Kerschberg, ed., *Expert Database Systems, Proceedings First International Workshop* (Benjamin/Cummings, Menlo Park, CA, 1986) 69–78.
[5] A. Fuhrmann and S.O. Hansson, A survey of multiple contraction, *J. Logic Lang. Inf.* **3** (1994) 39–76.
[6] D. Gabbay, Theoretical foundations for nonmonotonic reasoning in expert systems, in: Apts, ed., *Logics and Models of Concurrent Systems* (Springer, Berlin, 1985) 439–457.
[7] P. Gärdenfors, An epistemic approach to conditionals, *Amer. Philos. Quart.* **18** (1981) 203–211.
[8] P. Gärdenfors, Rules for rational changes of belief, in: *Philosophical Essays Dedicated to L. Åqvist* (Uppsala, 1982) 88–101.
[9] P. Gärdenfors, Belief revision and the Ramsey test for conditionals, *Philos. Rev.* **95** (1986) 81–93.
[10] P. Gärdenfors, *Knowledge in Flux Modeling the Dynamics of Epistemic States* (MIT Press, Cambridge, MA, 1988).
[11] P. Gärdenfors and H. Rott, Belief revision, in: *The Handbook of Logic in Artificial Intelligence and Logic Programming* (Oxford University Press, Oxford, to appear).
[12] S.O. Hansson, Reversing the Levi identity, *J. Philos. Logic* **22** (1993) 637–669.
[13] S.O. Hansson, Theory contraction and base contraction unified, *J. Symbolic Logic* **58** (1993) 602–625.
[14] S.O. Hansson, Taking belief bases seriously, in: D. Prawitz and D. Westerståhl, eds., *Logic and Philosophy of Science in Uppsala* (Kluwer, Dordrecht, Netherlands, 1994) 13–28.
[15] S.O. Hansson, Knowledge-level analysis of belief base operations, *Artif. Intell.* **82** (1996) 215–235 (this volume).
[16] S. Kraus, D. Lehmann and M. Magidor, Nonmonotonic reasoning, preferential models and cumulative logics, *Artif. Intell.* **44** (1990) 167–207.
[17] I. Levi, Subjunctives, dispositions and chances, *Synthese* **34** (1977) 423–455.
[18] D. Makinson, General patterns in nonmonotonic reasoning, in: *The Handbook of Logic in Artificial Intelligence and Logic Programming* (Oxford University Press, Oxford, to appear).
[19] D. Makinson and P. Gärdenfors, Relations between the logic of theory change and nonmonotonic logic, in: A. Fuhrmann, ed., *The Logic of Theory Change* (Springer, Berlin, 1989) 185–205.
[20] A. Newell, The knowledge level, *Artif. Intell.* **18** (1982) 87–127.
[21] H. Rott, Belief contraction in the context of the general theory of rational choice, *J. Symbolic Logic* **58** (1993) 1426–1450.
[22] H. Rott, Two perspectives on belief change, Manuscript (1995).

ELSEVIER

Artificial Intelligence 82 (1996) 353–367

Artificial Intelligence

Research Note

On the relative expressiveness of description logics and predicate logics ★

Alex Borgida *

Department of Computer Science, Rutgers University, New Brunswick, NJ 08903, USA

Received June 1995; revised January 1996

Abstract

It is natural to view concept and role definitions in description logics as expressing monadic and dyadic predicates in predicate calculus. We show that the descriptions built using the constructors usually considered in the DL literature are characterized exactly as the predicates definable by formulas in $\ddot{\mathcal{L}}^3$, the subset of first-order predicate calculus with monadic and dyadic predicates which allows only three variable symbols. In order to handle "number bounds", we allow numeric quantifiers, and for transitive closure of roles we use infinitary disjunction. Using previous results in the literature concerning languages with limited numbers of variables, we get as corollaries the existence of formulas of FOPC which cannot be expressed as descriptions. We also show that by omitting role composition, descriptions express exactly the formulas in $\ddot{\mathcal{L}}^2$, which is known to be decidable.

1. Overview

Description languages (DLs) are descendants of the KL-ONE [6] knowledge representation system, and have been the object of intensive theoretical study in the past decade, as well as forming the basis of several widely used implemented systems. A significant contribution of the original KL-ONE proposal of Brachman was the idea that in addition to primitive notions, such as "Person", one can also *define* concepts such as "Persons with at least three friends"—the definition providing both necessary and sufficient conditions for membership in this new concept.

It seems natural to wonder whether there are limits to the concepts that one could define in KL-ONE, or one of its descendants; for example, in a knowledge base dealing

★ Preliminary versions of some results were presented, for a database audience, at the CIKM'94 Conference.

* Supported in part by NSF Grant IRI 91-19310. E-mail: borgida@cs.rutgers.edu.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
PII S0004-3702(96)00004-5

with persons and their relationships, is it possible to define the notion of "persons who have a clique of at least four friends" (i.e., persons who have four friends, who in turn are all friends of each other). Surprisingly, such a question has not been generally addressed in the knowledge representation literature (but see [2]), although in databases there has been considerable work on comparing the "expressive power" of various query languages.

After establishing a formal framework that allows us to compare the "meaning" expressed by two different formalisms, we provide results that show, among others, that the DLs considered so far, even ones for which subsumption is undecidable, can express only (and all) those notions that can be expressed in (variants of) FOPC with three or fewer variables. Previous results in the literature then show that there are indeed things that one can say with $k+1$ variables that cannot be expressed in any way with just k variables. These limitations are all the more significant since, recently, there have been a number of proposals for using DLs as query languages for accessing data in databases (e.g., [5,7]). It is in fact this practical question that motivated the present research.

The equivalences established in this paper also offer, as incidental corollaries, alternate proofs of decidability and undecidability for several subsets of description constructors, using previously known results about predicate calculus.

2. Descriptions

Descriptions are used to specify concepts (which group individuals) and roles (which relate pairs of individuals). For example, consider the description in Fig. 1. It is constructed from identifiers denoting binary relations (e.g., `venue`, `players`), individuals (e.g., `Toronto`, `1`) and other concepts (e.g., `GAME`, `STADIUM`) using description constructors **and**, **all**, **at-most**, **fills** and **one-of**. The description in Fig. 1 has as intended denotation "*Games which are held in a stadium* (their `venue` role's value must be an instance of concept `STADIUM`), *involving at most ten players* (the `players` role has at most ten values/fillers), *all of whom are from Toronto* (all `players` fillers are restricted to have value `Toronto` for attribute `hometown`)". Roles need not be atomic identifiers—there are composite descriptions denoting roles. For example, the term **compose**[`children`, **restrict**[`children`, `MALE`]] denotes "*grandsons*" in English, since **restrict** takes a binary relation such as `children` and derives the one where all elements in its range are instances of concept `MALE` (hence `sons`), and **compose** corresponds to binary relation composition.

```
and[
    all[venue, STADIUM]
    at-most[10, players]
    all[players, fills[hometown, Toronto]]
```

Fig. 1. Composite description for a concept.

2.1. The languages and logics of descriptions

The language of descriptions is obtained recursively by starting from a *schema* $S = (\mathcal{CN}, \mathcal{RN}, \mathcal{IN})$ of names for concepts, roles, and individuals, and building from them more complex terms using description constructors.

Over the years, a considerable variety of DLs have been proposed, studied and implemented. Table 1 presents the language $\mathcal{DL}$, which contains a comprehensive list of the constructors based on recent survey papers [3, 19, 24]. (Clearly, the set of constructors in $\mathcal{DL}$ is not minimal; this is intentional, since considerable work in the field is devoted to finding subsets of constructors for which various decision problems have good computational properties.)

In the table, and elsewhere, we use the symbols A, B, C to range over concept descriptions, $p, q, \ldots$ to range over role descriptions, $a, b, \ldots$ for individual names, and D, E, F to denote descriptions in general.

The semantics of description terms is given denotationally, using the notion of an *interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, (\cdot)^{\mathcal{I}} \rangle$, which starts with a domain (non-empty universe) of

Table 1
$\mathcal{DL}$: a compendium of concept and role constructors

Term	Interpretation
TOP-CONCEPT	$\Delta^{\mathcal{I}}$
NOTHING	$\emptyset$
and[C, D]	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
or[C, D]	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
not[C]	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
all[p, C]	$\{\delta \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(\delta) \subseteq C^{\mathcal{I}}\}$
some[p, C]	$\{\delta \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(\delta) \cap C^{\mathcal{I}} \neq \emptyset\}$
at-least[n, p]	$\{\delta \in \Delta^{\mathcal{I}} \mid \lvert p^{\mathcal{I}}(\delta)\rvert \geqslant n\}$
at-most[n, p]	$\{\delta \in \Delta^{\mathcal{I}} \mid \lvert p^{\mathcal{I}}(\delta)\rvert \leqslant n\}$
at-least-c[n, p, C]	$\{\delta \in \Delta^{\mathcal{I}} \mid \lvert p^{\mathcal{I}}(\delta) \cap C^{\mathcal{I}}\rvert \geqslant n\}$
at-most-c[n, p, C]	$\{\delta \in \Delta^{\mathcal{I}} \mid \lvert p^{\mathcal{I}}(\delta) \cap C^{\mathcal{I}}\rvert \leqslant n\}$
same-as[p, q]	$\{\delta \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(\delta) = q^{\mathcal{I}}(\delta)\}$
subset[p, q]	$\{\delta \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(\delta) \subseteq q^{\mathcal{I}}(\delta)\}$
not-same-as[p, q]	$\{\delta \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(\delta) \neq q^{\mathcal{I}}(\delta)\}$
fills[p, b]	$\{\delta \in \mathrm{dom}^{\mathcal{I}} \mid b^{\mathcal{I}} \in p^{\mathcal{I}}(\delta)\}$
one-of[$b_1, \ldots, b_m$]	$\{b_1^{\mathcal{I}}, \ldots, b_m^{\mathcal{I}}\}$
TOP-ROLE	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
IDENTITY	$\{(\delta, \delta) \mid \delta \in \Delta^{\mathcal{I}}\}$
role-and[p, q]	$p^{\mathcal{I}} \cap q^{\mathcal{I}}$
role-or[p, q]	$p^{\mathcal{I}} \cup q^{\mathcal{I}}$
role-not[p]	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}$
inverse[p]	$\{(\delta, \delta') \mid (\delta', \delta) \in R^{\mathcal{I}}\}$
restrict[p, C]	$\{(\delta, \delta') \in p^{\mathcal{I}} \mid \delta' \in C^{\mathcal{I}}\}$
compose[p, q]	$p^{\mathcal{I}} \circ q^{\mathcal{I}}$
product[C, D]	$C^{\mathcal{I}} \times D^{\mathcal{I}}$
trans[p]	$\bigcup_{n \geqslant 0} (p^{\mathcal{I}})^n$

values $\Delta^{\mathcal{I}}$, and a mapping $(\cdot)^{\mathcal{I}}$ from concept descriptions to subsets of the domain, and role descriptions to sets of 2-tuples over the domain; the mapping also associates with every individual name $\mathcal{IN}$ some distinct value in $\Delta^{\mathcal{I}}$. (The reason for distinctness is the Unique Name Assumption, which is normally made in knowledge representation.) The interpretation function $(\cdot)^{\mathcal{I}}$ is extended recursively to composite descriptions in Table 1, where the interpretation of roles is viewed, in the obvious ways, as a function from an individual to the set of individuals related to it (i.e., as $\mathcal{RN} \rightarrow (\Delta^{\mathcal{I}} \rightarrow 2^{\Delta^{\mathcal{I}}})$).

The *meaning* of a description D can then be thought of as a mapping from interpretations $\mathcal{I}$ to extents $D^{\mathcal{I}}$, and a variety of logical judgments are defined on its basis:

- description E *subsumes* D, written $D \Longrightarrow E$, iff for every interpretation $\mathcal{I}$, $D^{\mathcal{I}} \subseteq E^{\mathcal{I}}$;
- description D is coherent/satisfiable, if there is at least one $\mathcal{I}$ such that $D^{\mathcal{I}} \neq \emptyset$;
- descriptions E and D are disjoint iff for every interpretation $\mathcal{I}$, $D^{\mathcal{I}} \cap E^{\mathcal{I}} = \emptyset$.

3. Relating descriptions to predicate calculus

There is an obvious similarity between concepts (respectively roles) in DLs, and monadic (respectively dyadic) predicates in predicate calculus. Such a similarity was already exploited by Schmolze and Israel [22] to give a semantics for the original KL-ONE language using the λ-calculus. Our aim is to compare the "expressive power" of various sublanguages of descriptions and predicate calculus. To do so we need to set up a common framework for the two formalisms.

3.1. Predicate calculus

We start, as usual, from a set of names for predicates and individual constants. Since almost all the work on DLs has been carried out in the framework of binary (as opposed to n-ary) roles/relationships, we will side-step the problem of dealing with n-ary predicates by restricting the arity of predicates to 1 or 2. We therefore start again from a schema $S = (\mathcal{CN}, \mathcal{RN}, \mathcal{IN})$ of monadic and dyadic predicates, and constant symbols. From these, as well as the equality predicate and variable symbols in the set $\mathcal{VN}$ (which is assumed to have a lexicographic ordering on it), atomic and composite formulas are built, as usual, with connectives $\neg$, $\wedge$, $\exists$, the other connectives being defined as macros. The notion of "free variable" is defined as usual, and we use the notation $\Psi(x, y)$ to refer to a formula that has as free variables, x and y, and only them.

The semantics of $\mathcal{FOPC}$ is also based on an interpretation $\mathcal{I}$ of $\mathcal{CN}$, $\mathcal{RN}$ and $\mathcal{IN}$, but in this case one traditionally also has to deal with variable symbols; this can be done by means of a partial function $\mu : \mathcal{VN} \rightarrow \Delta^{\mathcal{I}}$ which provides a substitution for some of the variables. An interpretation $\mathcal{I}$ and a substitution μ define a partial function $[\![\cdot]\!]^{\mathcal{I},\mu}$ from formulas to truth values $\{True, False\}$ in the usual way; for example, for the formula $P(x, a)$, we have $[\![P(x, a)]\!]^{\mathcal{I},\mu} = True$ iff $(\mu(x), a^{\mathcal{I}}) \in P^{\mathcal{I}}$.

In order to be able to compare formulas with free variables on equal footing with descriptions, we extend the interpretation function $(\cdot)^{\mathcal{I}}$ to formulas of the predicate calculus as done in the field of databases: the meaning $\Psi^{\mathcal{I}}$ of a formula Ψ is a set of k-tuples, where k is the number of distinct free variables in Ψ:

$$\Psi(x_1,\ldots,x_k)^{\mathcal{I}} = \{(\alpha_1,\ldots,\alpha_k) \mid \alpha_i \in \Delta^{\mathcal{I}}, [\![\Psi(x_1,\ldots,x_k)]\!]^{\mathcal{I},\mu} = True \text{ for the substitution } \mu \text{ mapping } x_i \text{ to } \alpha_i\}.$$

Since the tuples are ordered, in order to obtain a unique meaning we require the free variables $x_1, x_2, \ldots$ to appear in lexicographic order.

The *meaning of a formula* Ψ is then defined as the function $\lambda\mathcal{I}.\Psi^{\mathcal{I}}$, where $\Psi^{\mathcal{I}}$ is defined as above if Ψ has at least one free variable, and is otherwise a truth value. Note that by the semantic definition of $\forall$ and $\Leftrightarrow$, for any formulas $\Psi(x_1,\ldots)$ and $\Phi(x_1,\ldots)$

Ψ and Φ have the same meaning
iff $\forall x_1,\ldots.\Psi(x_1,\ldots) \Leftrightarrow \Phi(x_1,\ldots)$ is a theorem.

3.2. Comparing the meaning of formulas in different languages

We now have "meanings" for sentences in both languages defined uniformly as mappings from interpretations to sets of tuples over $\Delta^{\mathcal{I}}$. We can therefore say that description D has the same meaning as formula $\Psi(x)$ iff $\lambda\mathcal{I}.D^{\mathcal{I}} = \lambda\mathcal{I}.\Psi(x)^{\mathcal{I}}$, where $\mathcal{I}$ are interpretations over the same schema S.[1] We can also create hybrid sentences, which mix the two kinds of formulas: On the one hand, we can allow descriptions to appear as monadic and dyadic predicates in FOPC formulas, interpreting $[\![D(a)]\!]^{\mathcal{I},\mu}$, where D is a concept description, as $a^{\mathcal{I}} \in D^{\mathcal{I}}$; in this formulation, $\Psi(x)$ expresses the meaning of D iff $\forall x.D(x) \Leftrightarrow \Psi(x)$ is a theorem. On the other hand, we can treat $\Psi(x)$ as a description by interpreting it using $(\cdot)^{\mathcal{I}}$ whenever we encounter it, in which case $\Psi(x)$ expresses the meaning of D iff $D \Longrightarrow \Psi(x)$ and $\Psi(x) \Longrightarrow D$. Similar hybrid formulas can be set up for the other kinds of judgments one is normally interested in for DLs.

We will then say that some language $\mathcal{L}_2$ *is as expressive as* language $\mathcal{L}_1$, if there is a total function *transl* from $\mathcal{L}_1$ to $\mathcal{L}_2$ such that for every sentence L in $\mathcal{L}_1$, *transl*(L) expresses the meaning of L. Two languages are *equally expressive* if each is as expressive as the other.

For example, Schmolze and Israel [22] show that FOPC is as expressive as $\mathcal{DL} - \{\textbf{trans}\}$ by essentially defining a translation function $\tau\langle\cdot\rangle$, which maps concepts to formulas with free variable x, and roles to formulas with free variables x and y. For example, $\tau\langle\textbf{all}[\mathrm{p},\mathrm{C}]\rangle$ is "$\lambda x.\forall w.p(x,w) \Rightarrow C(w)$", while the translation of $\textbf{compose}[\mathrm{p},\mathrm{q}]$ is $\lambda x,y.\exists z.p(x,z) \wedge q(z,y)$.

4. DLs and FOPC with limited variables

Note that the translation from descriptions to FOPC mentioned in the previous section must introduce new variables whenever a new quantifier appears, in order to avoid spurious capture of variables. For example, without this precaution, the translation of $\textbf{compose}(\mathrm{p},\textbf{compose}(\mathrm{p},\mathrm{p}))$ would yield $\exists z.p(x,z) \wedge \exists z.(p(z,z) \wedge p(z,y))$, which is clearly wrong—one wants $\exists z_2.p(x,z_2) \wedge \exists z_1.(p(z_2,z_1) \wedge p(z_1,y))$.

[1] For alternate techniques for comparing the "expressive power" of languages, see [2,23].

Table 2

Term C	$\mathcal{T}^x\langle \mathtt{C}\rangle$	$\mathcal{T}^y\langle \mathtt{C}\rangle$
TOP-CONCEPT	$x = x$	$y = y$
NOTHING	$\neg(x = x)$	$\neg(y = y)$
and[C, D]	$\mathcal{T}^x\langle \mathtt{C}\rangle \wedge \mathcal{T}^x\langle \mathtt{D}\rangle$	$\mathcal{T}^y\langle \mathtt{C}\rangle \wedge \mathcal{T}^y\langle \mathtt{D}\rangle$
or[C, D]	$\mathcal{T}^x\langle \mathtt{C}\rangle \vee \mathcal{T}^x\langle \mathtt{D}\rangle$	$\mathcal{T}^y\langle \mathtt{C}\rangle \vee \mathcal{T}^y\langle \mathtt{D}\rangle$
not[C]	$\neg\mathcal{T}^x\langle \mathtt{C}\rangle$	$\neg\mathcal{T}^y\langle \mathtt{C}\rangle$
all[p, C]	$\forall y.\mathcal{T}^{x,y}\langle \mathtt{p}\rangle \Rightarrow \mathcal{T}^y\langle \mathtt{C}\rangle$	$\forall x.\mathcal{T}^{y,x}\langle \mathtt{p}\rangle \Rightarrow \mathcal{T}^x\langle \mathtt{C}\rangle$
some[p, C]	$\exists y.\mathcal{T}^{x,y}\langle \mathtt{p}\rangle \wedge \mathcal{T}^y\langle \mathtt{C}\rangle$	$\exists x.\mathcal{T}^{y,x}\langle \mathtt{p}\rangle \wedge \mathcal{T}^x\langle \mathtt{C}\rangle$
subset[p, q]	$\forall y.\mathcal{T}^{x,y}\langle \mathtt{p}\rangle \Rightarrow \mathcal{T}^{x,y}\langle \mathtt{q}\rangle$	$\forall x.\mathcal{T}^{y,x}\langle \mathtt{p}\rangle \Rightarrow \mathcal{T}^{y,x}\langle \mathtt{q}\rangle$
same-as[p, q]	$\forall y.\mathcal{T}^{x,y}\langle \mathtt{p}\rangle \Leftrightarrow \mathcal{T}^{x,y}\langle \mathtt{q}\rangle$	$\forall x.\mathcal{T}^{y,x}\langle \mathtt{p}\rangle \Leftrightarrow \mathcal{T}^{y,x}\langle \mathtt{q}\rangle$
not-same-as[p, q]	$\exists y.\neg(\mathcal{T}^{x,y}\langle \mathtt{p}\rangle \Leftrightarrow \mathcal{T}^{x,y}\langle \mathtt{q}\rangle)$	$\exists x.\neg(\mathcal{T}^{y,x}\langle \mathtt{p}\rangle \Leftrightarrow \mathcal{T}^{y,x}\langle \mathtt{q}\rangle)$
fills[p, b]	$\exists y.(y = b) \wedge \mathcal{T}^{x,y}\langle \mathtt{p}\rangle$	$\exists x.(x = b) \wedge \mathcal{T}^{y,x}\langle \mathtt{p}\rangle$
one-of[$\mathtt{b}_1, \ldots, \mathtt{b}_m$]	$x = b_1 \vee \cdots \vee x = b_m$	$y = b_1 \vee \cdots \vee y = b_m$

We now show that in fact one does not need to introduce new variables, thus relating DLs to FOPC with limited number of variables. So let $\mathcal{L}^k$ be the set of all FOPC formulas with equality which can be expressed using at most k variables. Note that $\mathcal{L}^k$ does not limit the number of nested quantifiers in a formula since the same variable may be reused in nested subformulas, as in $\forall x, y.P(x, y) \Rightarrow \exists x.Q(y, x)$. Properties of such language families have been studied, among others, in [8, 11, 16]. In our case, since we are dealing with roles and concepts, we will be interested only in those formulas that (i) have one or two free variables (though they may have closed subformulas), and (ii) have only monadic and dyadic predicates. Henceforth, we will use $\ddot{\mathcal{L}}^k$ to refer to this sublanguage.

Our first result shows that almost everything that can be said with DLs, can be said with just a few variables.

Theorem 1. *The language $\ddot{\mathcal{L}}^3$ is as expressive as $\mathcal{DL}$ – {**trans**, **at-least**, **at-most**}. The language $\ddot{\mathcal{L}}^2$ is as expressive as $\mathcal{DL}$ – {**compose**, **trans**, **at-least**, **at-most**}.*

Proof. The proof relies on a more careful encoding of the constructors into predicate calculus, where the same variable is reused as much as possible. We will present the translation function in several variants that behaves as follows: $\mathcal{T}^x\langle\rangle$ makes x be the free variable of the monadic predicate it will produce for its argument concept, while $\mathcal{T}^y\langle\rangle$ makes the free variable be y. So, for a concept C in $\mathcal{CN}$, $\mathcal{T}^x\langle \mathtt{C}\rangle = C(x)$, while $\mathcal{T}^y\langle \mathtt{C}\rangle = C(y)$. In the case of roles R, $\mathcal{T}^{x,y}\langle \mathtt{R}\rangle$ produces a predicate $R(x, y)$, while $\mathcal{T}^{y,x}\langle \mathtt{R}\rangle$ produces predicate $R(y, x)$. The translation functions $\mathcal{T}^x\langle\rangle$, $\mathcal{T}^y\langle\rangle$, and $\mathcal{T}^{x,y}\langle\rangle$ are presented in Tables 2 and 3. $\mathcal{T}^{y,x}\langle\rangle$ is obtained from $\mathcal{T}^{x,y}\langle\rangle$ by simultaneously exchanging *all* occurrences of x and y (whether free or bound).

The key ideas in the above translation are the alternating use of $\mathcal{T}^x\langle\rangle$ and $\mathcal{T}^y\langle\rangle$ in nested concept descriptions (such as **all** or **some**), and the use of the equalities $x = z$ and $y = z$ in the translation of **compose**, which make it unnecessary to introduce new variables during the translation process.

The translation function $\mathcal{T}\langle\rangle$ can now be defined simply as $\mathcal{T}\langle \mathtt{C}\rangle = \mathcal{T}^x\langle \mathtt{C}\rangle$ for concept descriptions C, and $\mathcal{T}\langle \mathtt{r}\rangle = \mathcal{T}^{x,y}\langle \mathtt{r}\rangle$ for role descriptions r. Once again, a straightfor-

Table 3

Term R	Translation $\mathcal{T}^{x,y}\langle \mathtt{R} \rangle$
TOP-ROLE	$x = x \wedge y = y$
IDENTITY	$x = y$
role-and[p, q]	$\mathcal{T}^{x,y}\langle \mathtt{p} \rangle \wedge \mathcal{T}^{x,y}\langle \mathtt{q} \rangle$
role-or[p, q]	$\mathcal{T}^{x,y}\langle \mathtt{p} \rangle \vee \mathcal{T}^{x,y}\langle \mathtt{q} \rangle$
role-not[p]	$\neg\mathcal{T}^{x,y}\langle \mathtt{p} \rangle$
inverse[p]	$\mathcal{T}^{y,x}\langle \mathtt{p} \rangle$
restrict[p, C]	$\mathcal{T}^{x,y}\langle \mathtt{p} \rangle \wedge \mathcal{T}^{y}\langle \mathtt{C} \rangle$
compose[p, q]	$\exists z.(\exists y.y = z \wedge \mathcal{T}^{x,y}\langle \mathtt{p} \rangle) \wedge (\exists x.x = z \wedge \mathcal{T}^{x,y}\langle \mathtt{q} \rangle)$
product[C, D]	$\mathcal{T}^{x}\langle \mathtt{C} \rangle \wedge \mathcal{T}^{y}\langle \mathtt{D} \rangle$

Table 4

$\Upsilon(x)$	D_Υ
$C(x)$, $C \in \mathcal{CN}$	C
$P(x,b)$, $P \in \mathcal{RN}$	**fills**[P, b]
$P(b,x)$	**fills**[**inverse**[P], b]
$P(x,x)$	**some**[**role-and**[P, IDENTITY], TOP-CONCEPT]
$x = b$	**one-of**[b]
$x = x$	TOP-CONCEPT
$\neg\Psi_1(x)$	**not**[D_{Ψ_1}]
$\Psi() \wedge \Phi(x)$	**and**[D_Ψ, D_Φ]
$\Psi(x) \wedge \Phi(x)$	**and**[D_Ψ, D_Φ]
$\exists y.\Psi(x,y)$	**some**[R_Ψ, TOP-CONCEPT]
$\exists y.\Psi(x)$	D_Ψ

ward proof by induction shows that for every description D and interpretation $\mathcal{I}$, $\mathtt{D}^{\mathcal{I}} = \mathcal{T}\langle \mathtt{D} \rangle^{\mathcal{I}}$. □

It turns out that the converses of the above results also hold. To begin with, we have:

Theorem 2. *The description language with concept constructors* {TOP-CONCEPT, NOTHING, **and, not, some, fills, one-of**} *and role constructors* {**role-and, role-not, product, inverse**} *is as expressive as* $\ddot{\mathcal{L}}^2$.

Proof. Suppose the two variables we can use in $\ddot{\mathcal{L}}^2$ are x and y. We shall proceed by structural recursion on the syntax of formulas with up to two free variables.

Table 4 lists the various kinds of formulas $\Upsilon(x)$ that have a single free variable x, and shows how each kind is translated into a concept description D_Υ:

The translation of formulas with a single free variable y is identical, except for the case when $\Upsilon(y)$ is of the form $\exists x.\Psi(x,y)$, when we need to invert the relationship represented by Ψ, so it is translated as **some**[**inverse**[D_Ψ], TOP-CONCEPT].

Formulas of the form $\Psi(x,y)$ are translated to roles relating x to y according to Table 5.

A formula $\Upsilon()$ without free variables occurs only as a conjunct, and the number of free variables (one or two) in its context determines its translation: a concept or a role.

Table 5

$\Upsilon(x,y)$	R_Υ
$P(x,y)$, $P \in \mathcal{RN}$	P
$P(y,x)$	**inverse**[P]
$x = y$	IDENTITY
$\neg\Psi(x,y)$	**role-not**[$R_{\Psi(x,y)}$]
$\Psi(x) \wedge \Phi(y)$	**product**[D_Ψ, D_Φ]
$\Psi(x,y) \wedge \Phi()$	**role-and**[R_Ψ, R_Φ]
$\Psi(x,y) \wedge \Phi(x)$	**role-and**[R_Ψ, **product**[D_Φ, TOP-CONCEPT]]
$\Psi(x,y) \wedge \Phi(y)$	**role-and**[R_Ψ, **product**[TOP-CONCEPT, D_Φ]]
$\Psi(x,y) \wedge \Phi(x,y)$	**role-and**[R_Φ, R_Ψ]

Table 6

$\Upsilon()$	D_Υ
$C(b)$	**all**[**product**[TOP-CONCEPT, **oneof**[b]], C]
$P(b,b)$	**all**[**product**[TOP-CONCEPT, **oneof**[b]], **fills**[P, b]]
$P(a,b)$	**all**[**product**[TOP-CONCEPT, **oneof**[a]], **fills**[P, b]]
$b = b$	TOP-CONCEPT
$a = b$	NOTHING
$\neg\Psi()$	**not**[$D_{\Psi()}$]
$\Psi() \wedge \Phi()$	**and**[$D_{\Psi()}$, $D_{\Phi()}$]
$\exists x.\Psi(x)$	**some**[TOP-ROLE, D_Ψ]
$\exists y.\Psi(y)$	**some**[TOP-ROLE, D_Ψ]
$\exists x.\Psi()$	D_Ψ

For the case when a concept is desired, we need a description D_Υ with the property that for any interpretation $\mathcal{I}$, if $\Upsilon()^\mathcal{I} = True$ then $D_\Upsilon^\mathcal{I} = \Delta^\mathcal{I}$, and if $\Upsilon()^\mathcal{I} = False$ then $D_\Upsilon^\mathcal{I} = \emptyset$. This essentially "gates" the meaning of the other conjunct. Table 6 provides such translations.

In contexts where we require roles, the translation is just

$$R_{\Upsilon()} = \textbf{product}[D_{\Upsilon()}, D_{\Upsilon()}]. \qquad \square$$

More generally, we have:

Theorem 3. *The description language* $\mathcal{DL} - \{\textbf{trans}, \textbf{at-least}, \textbf{at-most}\}$ *is as expressive as* $\ddot{\mathcal{L}}^3$.

Proof. In this case we deal with formulas having possibly three variables: x, y, and z. Once again we define a recursive procedure for translating formulas into descriptions. Formulas Ψ with one or two free variables are translated, as before, into concept descriptions D_Ψ or role descriptions R_Ψ. The translation of formulas of the form $\Psi(x,z)$ or $\Psi(y,z)$ is carried out by the same procedure as for formulas $\Psi(x,y)$. Similarly, the translation of monadic formulas of the form $\Psi(y)$ and $\Psi(z)$ is handled by the same procedure as for formulas $\Psi(x)$.

The main novelty therefore lies in the translation of subformulas $\Psi(x,y,z)$ with three free variables. Since the final formula can have no more than two free variables, every subformula $\Psi(x,y,z)$ must in fact be part of a larger subformula of the form $\exists\gamma.\Phi(x,y,z)$, where γ is either x, y or z. Without loss of generality, suppose it is $\exists z.\Phi(x,y,z)$. Because all atomic formulas involve only monadic and dyadic predicates, we will be able to prove that it is sufficient to consider the case when $\exists z.\Phi(x,y,z)$ is of the form $\exists z.\Phi_1(x,z) \wedge \Phi_2(y,z)$, which can be translated as the role description **compose**$[R_{\Phi_1}$, **inverse**$[R_{\Phi_2}]]$.

It therefore remains to show that every formula of the form $\exists z.\Phi(x,y,z)$ can be reduced to the form $\exists z.\Phi_1(x,z) \wedge \Phi_2(y,z)$. This is accomplished by massaging $\Phi(x,y,z)$ into a normal form that allows the quantifier to be moved in. Specifically, let Φ_i be the *maximal subformulas* of $\Phi(x,y,z)$ that have at most two free variables. Since there are only three variables and all predicates are of arity at most 2, $\Phi(x,y,z)$ must be the boolean combination of these Φ_i—any intervening quantifier would reduce the number of variables to 2, and hence would be part of some Φ_j. Using de Morgan's laws, it is therefore possible to rewrite $\Phi(x,y,z)$ into disjunctive normal form $\bigvee_j(\bigwedge_k \Phi_{j,k})$, where each $\Phi_{j,k}$ is either some Φ_i or its negation. Since an existential quantifier can be moved in past disjunctions ($\exists z.(\alpha \vee \beta) \equiv (\exists z.\alpha) \vee (\exists z.\beta)$), $\exists z.\Phi(x,y,z)$ is therefore logically equivalent to $\bigvee_j \Theta_j$ where $\Theta_j = \exists z.(\bigwedge_k \Phi_{j,k})$, and note that each Θ_j has at most two free variables (z is being quantified over).

Therefore we need only consider formulas $\exists z.\Phi(x,y,z)$ where $\Phi(x,y,z)$ is the conjunction of subformulas $\Phi_{j,k}$, each with at most two free variables. By associativity of conjunction, group together the subformulas that have the same free variables, thereby obtaining that $\Phi(x,y,z)$ is in the most general case of the form $\Psi_0() \wedge \Psi_1(x) \wedge \Psi_2(y) \wedge \Psi_3(z) \wedge \Psi_4(x,y) \wedge \Psi_5(x,z) \wedge \Psi_6(y,z)$. But then we can move the subformulas not containing z outside the quantifier, rewriting $\exists z.\Phi(x,y,z)$ in the form $\beta(x,y) \wedge \exists z.((\Psi_3(z) \wedge \Psi_5(x,z)) \wedge \Psi_6(y,z))$. Therefore, in the end the formula in the scope of $\exists z$ does have the desired restricted form $\exists z.\Psi_7(x,z) \wedge \Psi_5(y,z)$, establishing our claim. (Note that if subformulas Ψ_5 or Ψ_6 are empty then we are left inside the scope of $\exists z$ with a formula with at most two free variables, whose translation had already been provided earlier.) □

Combining the preceding theorems we get

Corollary 4.

(i) *The description language* $\mathcal{DL}$ – {**trans**, **compose**, **at-least**, **at-most**} *and* $\ddot{\mathcal{L}}^2$ *are equally expressive.*

(ii) *The description language* $\mathcal{DL}$ – {**trans**, **at-least**, **at-most**} *and* $\ddot{\mathcal{L}}^3$ *are equally expressive.*

4.1. Consequences concerning DLs

Several potentially interesting corollaries follow from the above theorems and existing results in the logic and computer science literature. In each case, we associate the corollary with the reference where the key result comes from.

Corollary 5 (Immerman). *There exists a "conjunctive" FOPC formula $\Psi(x)$ (one involving only logical connectives $\wedge$ and $\exists$) which cannot be expressed by any description in $\mathcal{DL} - \{$**trans, at-least, at-most**$\}$.*

This is true for any conjunctive formula that is expressible in FOL but not in $\mathcal{L}^3$. For example, Immerman [11] shows, among others, that for the schema $\langle\{NODE\},\{Edge\},\emptyset\rangle$, which can be used to represent a graph, $\ddot{\mathcal{L}}^{k-1}$ does not allow the expression of such graph-theoretic properties as the existence of a k-subclique. In other words, there are pairs of graphs that differ in having the property but that cannot be distinguished by any formula of $\ddot{\mathcal{L}}^{k-1}$. Therefore, for the following simple formula $\Psi(y)$ from $\ddot{\mathcal{L}}^4$

$$(\exists x_1, x_2, x_3)\left(NODE(y) \bigwedge_i Edge(y, x_i) \bigwedge_{i \neq j} Edge(x_i, x_j)\right)$$

there is no concept in $\mathcal{DL} - \{$**trans, at-least, at-most**$\}$ which has the same meaning.

Immerman's results also allow us to prove, among others, that **compose** provides an increase in expressive power, which is not entirely obvious since in some situations **compose** can be eliminated; for example **all**[**compose**[p, q], C] is equivalent to **all**[p, **all**[q, C]].

Corollary 6. *The constructor* **compose** *is independent of the other ones in $\mathcal{DL} - \{$**trans, at-least, at-most**$\}$ in the sense that it cannot be eliminated by finding an equivalent description without it.*

This is obtained simply from the fact that the formula

$$(\exists x_1, x_2)(NODE(y) \bigwedge_i Edge(y, x_i) \bigwedge_{i \neq j} Edge(x_i, x_j)),$$

which is in $\ddot{\mathcal{L}}^3$, is not in $\ddot{\mathcal{L}}^2$ for the schema above. (An alternate proof is provided by the decidability results below.)

As pointed out by Franz Baader, the previous results also have consequences concerning the decidability of subsumption for various classes of connectives.

Corollary 7.

(Mortimer) *Subsumption is decidable for $\mathcal{DL} - \{$**compose, trans, at-least, at-most**$\}$.*

(Lewis) *Subsumption is undecidable for $\mathcal{DL} - \{$**trans, at-least, at-most**$\}$.*

The first result follows from the decidability of validity in the logic $\mathcal{L}^2$ with equality, proven in [16]: the subsumption problem $D \Longrightarrow E$ can be posed as the validity of the formula $\forall x.\mathcal{T}^x\langle \mathrm{D}\rangle \Rightarrow \mathcal{T}^x\langle \mathrm{E}\rangle$, which by Theorem 1, is in $\ddot{\mathcal{L}}^2$.

The second result follows from the undecidability of validity for the class of formulas with quantifier prefix $\forall\exists\forall$, shown in [14]: the formula exhibited in Lewis' proof only has monadic and dyadic predicates, and given a closed formula $\Psi()$ in $\ddot{\mathcal{L}}^3$, its validity can be determined by verifying that the formula $\Psi'(x)$, defined as

$(x = x) \wedge \Psi$ is true for every substitution for x; by Theorem 3, $\Psi'(x)$ can then be expressed by a description $D_{\Psi'}$, and Ψ is valid if and only if the subsumption TOP-CONCEPT$\Longrightarrow D_{\Psi'}$ holds. (We note that the second result is not novel—various subsets of $\mathcal{DL}$ are known to be undecidable (e.g., [18])—but the proof is novel.)

4.2. Dealing with terminological axioms

Most description logics provide the ability to specify a knowledge base of additional assertions, which restrict the set of interpretations that are considered in making judgments.

The simplest such axioms provide definitions for some concept and role names, in the form $\mathtt{C} \doteq \mathtt{D}$ where $\mathtt{C} \in \mathcal{CN}$ and D is a concept description. If such definitions do not produce cycles (i.e., there is no recursion), then they only provide abbreviations and can be easily expanded out.

More generally, axioms of the form $\mathtt{D} \sqsubseteq \mathtt{E}$, for arbitrary descriptions D and E, constrain the meaning of descriptions to consider only those interpretations $\mathcal{I}$ which are "models" of the axioms, in the sense that $D^{\mathcal{I}} \subseteq E^{\mathcal{I}}$. (In this case definitions $\mathtt{C} \doteq \mathtt{D}$, even ones involving cycles, can be replaced by axioms $\mathtt{C} \sqsubseteq \mathtt{E}$ and $\mathtt{E} \sqsubseteq \mathtt{C}$, if we are content to capture the so-called "descriptive semantics" for recursion [17].) Therefore, given a knowledge base KB of axioms $\{E_i \sqsubseteq F_i\}$, the meaning of a description is now redefined as $\mathcal{M}[D, \mathrm{KB}](\mathcal{I}) = D^{\mathcal{I}}$ for interpretation $\mathcal{I}$ if for every i, $E_i^{\mathcal{I}} \subseteq F_i^{\mathcal{I}}$, and is $\emptyset$ otherwise.

In this case, the translation of descriptions into predicate calculus offered in Theorem 1 must be extended to also take into account the context of the KB. The following translation function

$$\mathcal{T}\langle \mathtt{D}, \mathtt{KB} \rangle = \mathcal{T}\langle \mathtt{D} \rangle \wedge \bigwedge_i (\forall x. \mathcal{T}^x \langle E_i \rangle \Rightarrow \mathcal{T}^x \langle F_i \rangle)$$

provides a formula which has the same meaning as D in the context of the KB. Significantly, $\mathcal{T}\langle \mathtt{D}, \mathtt{KB} \rangle$ is in $\ddot{\mathcal{L}}^3$ (or $\ddot{\mathcal{L}}^2$ respectively) just in case the translation of the descriptions D, E_i and F_i are themselves in this category. Therefore judgments such as subsumption and disjointness continue to translate to logical questions about $\ddot{\mathcal{L}}^2$ or $\ddot{\mathcal{L}}^3$.

Parallel arguments apply in case the knowledge base contains additional kinds of axioms, dealing for example with the disjointness of certain descriptions.

5. Numeric quantifiers and transitive closure

The translation of descriptions involving counting, such as **at-least**[7, `players`] into standard FOPC would seem to require seven distinct variables, which would put us outside the bounds of languages with limited variables. We proceed however by extending the syntax of FOPC to allow numeric/counting quantifiers, as in [12]; for example, $\exists_7 y.\mathtt{players}(x, y)$ predicates the existence of seven distinct values for which the formula is satisfied. Note that this is not treated as an abbreviation because we will wish to say that the above formula has only two variables, x and y!

Table 7

$\mathcal{T}$	$D_{\mathcal{T}}$
$\exists_n y.\Psi(x,y)$	**at-least**[n, R_Ψ]
$\exists_n y.\Psi(x)$	D_Ψ and **at-least**[n, TOP-ROLE]
$\exists_n x.\Psi(x)$	**at-least-c**[n, TOP-ROLE, D_Ψ]
$\exists_n y.\Psi(y)$	**at-least-c**[n, TOP-ROLE, D_Ψ]

Let us extend the languages $\ddot{\mathcal{L}}^k$ with counting quantifiers $\exists_n$, for every positive integer n, obtaining the languages $\ddot{\mathcal{C}}^k$—FOPC with monadic and dyadic predicates with counting quantifiers. In fact, we will take a restricted subset of such languages—ones where in any subformula $\exists_n.\Psi$, Ψ has no more than two free variables. For the case of two and three variables, let us call these languages $\ddot{\mathcal{L}}^2_{CNT}$ and $\ddot{\mathcal{L}}^3_{CNT}$.

Corresponding to Theorems 1–3, we then have:

Theorem 8.

(i) $\ddot{\mathcal{L}}^2_{CNT}$ *and the description language* $\mathcal{DL} - \{$**trans**, **compose**$\}$ *are equally expressive.*

(ii) $\ddot{\mathcal{L}}^3_{CNT}$ *and the description language* $\mathcal{DL} - \{$**trans**$\}$ *are equally expressive.*

Proof. For (i), in one direction $\mathcal{T}^x\langle$**at-least-c**[n, p, C]$\rangle$ is translated as

$$\exists_n y.(\mathcal{T}^{x,y}\langle \mathrm{p}\rangle \wedge \mathcal{T}^y\langle \mathrm{C}\rangle),$$

while **at-most-c**[n, p, C] is translated as

not[**at-least-c**[n+1, p, C]].

In the other direction, we need to give the translations given in Table 7 for formulas with zero, one or two free variables.

The proof of part (ii) follows immediately from part (i) and earlier proofs because we have restricted counting quantifiers (as opposed to the ordinary existential quantifier) so they can have in their scope only subformulas with two or fewer free variables. □

We leave it as an open problem whether the constructors in $\mathcal{DL}$ can be used to "simulate" formulas of the form $\exists_n z.\Psi(x,y,z)$, particularly $\exists_n z.\Phi_1(x,z) \wedge \Phi_2(y,z)$, or whether one would need to introduce a new version of **compose** for it. In the former case, $\ddot{\mathcal{L}}^3_{CNT}$ could be replaced by the more general $\ddot{\mathcal{C}}^3$ in the statement of the preceding theorem.

The importance of this characterization lies in the fact that in a more recent paper [8], Cai et al. present examples of the expressive limitations of $\ddot{\mathcal{C}}^k$. In particular, they describe pairs of graphs that cannot be distinguished using formulas of $\ddot{\mathcal{C}}^k$. These graphs can again be easily distinguished using only existential quantifiers and conjunction in FOPC. We therefore get once again corollaries about the expressive limitations of DLs, in this case even with number restrictions.

There are many ways in which one can attempt to deal with transitive closure. In order to obtain the similar kinds of results about the expressive limitation of DLs, we follow Kolaitis et al. in using infinitary disjunction, thus translating **trans** as

$$\mathcal{T}^{x,y}\langle \mathbf{trans}[\mathrm{p}]\rangle = \bigvee_{n=0}^{\infty} \mathcal{T}^{x,y}\langle \mathbf{compose}^n[\mathrm{p},\mathrm{p}]\rangle.$$

Infinitary disjunction leads to the FOPC variant $\mathcal{L}^k_{\infty,\omega}(\mathbf{COUNT})$. In this case we get the equivalent of Theorem 1, but not the converse, since there are of course many formulas involving infinite disjunction that cannot be expressed as transitive closure.

The inexpressibility results in [8, 11] are obtained using variants of Ehrenfeucht–Fraïssé pebbling games, which are shown to characterize $\ddot{\mathcal{L}}^k$ and $\ddot{\mathcal{C}}^k$. Kolaitis and Vardi [13] present a modified pebbling game that characterizes $\ddot{\mathcal{C}}^k$ extended with infinitary disjunction, and the proofs in [8] go through for this case, establishing that transitive closure will not be helpful in expressing the relevant formulas in [8]. As a result, we continue to have formulas in the FOPC with only existential quantification and conjunction that cannot be expressed as descriptions in the full $\mathcal{DL}$.

6. Conclusions

We have compared the "expressive power" of two different kinds of knowledge representation languages: description logics and predicate calculus. For this we have used a formal framework in which the meaning of a formula is defined to be a mapping from interpretations (over a fixed schema of identifiers) to sets of tuples over the domain of interpretation—a framework that preserves the logics of the two approaches. Our results characterize various subsets of the "universal" description language $\mathcal{DL}$ as having exactly the expressive power of certain subsets of FOPC with two or three variables, possibly augmented by counting quantifiers.

While we have obtained our results essentially by "direct simulation", it has been pointed out that several other results in the logical literature are relevant, and can be used to obtain alternative proofs in some cases. First, Tarski and Givant [23] have investigated a variable-free algebra for binary relations, for which they prove that it is "equipolent" with $\mathcal{L}^3$. Since the equations of this algebra can be expressed as axioms for descriptions in $\mathcal{DL}$, this provides an alternative path to proving that all of $\mathcal{L}^3$ can be expressed by $\mathcal{DL}$ (see [21]). The same referee points to a connection between description logics and $\mathcal{L}^2$ via results concerning modal logics: the description logic $\mathcal{ALC}$ is a notational variant of modal logic [20]; modal logic can be expressed in $\mathcal{L}^2$ [1]; $\mathcal{L}^2$ can be expressed by an appropriate modal logic [10].[2]

In addition to their intrinsic interest, the results presented here have several relevant consequences for research on DLs and their application, because, as corollaries of previous results from the logical literature on $\ddot{\mathcal{L}}^k$, we have pointed to the existence of certain FOPC formulas, built only with conjunction and existential quantification, which cannot be expressed as descriptions in $\mathcal{DL}$.

Traditionally, work in the DL area has attempted to identify *subsets* of description constructors from $\mathcal{DL}$, or limited forms thereof, for which subsumption is at least decidable, maybe even tractable, and which are expressively adequate for some particular

[2] The modal logics involved in these three results are not identical.

family of applications. Our results imply that this strategy will not work for hybrid systems in which the terminological component needs to express the above kinds of formulas. In such cases, one must look for entirely new kinds of concept constructors.

Second, in applications where DLs are used as query languages for existing data or knowledge bases (see [4] for a survey), one is very likely going to need an extended query language, since even the full $\mathcal{DL}$ cannot express the so-called "conjunctive queries"—the least powerful query languages considered in the relational database literature, and for which, incidentally, subsumption is definitely decidable. The obvious route to follow in this direction is to add formulas with variables, or their equivalent (e.g., relational algebra expressions), into description languages. For example, in Loom [15] FOPC formulas may be given as arguments to the **:satisfies** concept constructor. Another approach is to integrate descriptions and Horn formulas, as in [9]. Alternatively, one can consider ways of presenting queries that can create new objects or relationships (like relational algebra in databases).

Finally, an open question remaining in this work is to examine the expressive power of DLs with *recursive* concept definitions, where recursion is defined by some fixed point semantics. It is known that such forms of recursion enhance the power of database query languages, so we may expect similar kinds of results for DLs.

Acknowledgment

This work was prompted by a remark of Moshe Vardi on the possible connection to $\mathcal{L}^2$. I am deeply indebted to Franz Baader for extensive comments on this work and its presentation, and I have benefited from the insightful comments of the referees, as well as discussions with W. Nutt, R. van der Meyden, M. Yannakakis, and others. M. Vardi, A. Mendelzon, P. Kolaitis, and an anonymous referee provided very useful references, for which I am grateful.

References

[1] H. Andreka, J. van Benthem and I. Nemeti, Back and forth between modal logic and classical logic, *J. Int. Gp. Pure Appl. Logic* **3** (5) (1995) 685–720.

[2] F. Baader, A formal definition for expressive power of Knowledge Representation Languages, in: *Proceedings ECAI-90*, Stockholm (1990) 53–58.

[3] F. Baader, H-J. Bürckert, J. Heinsohn, B. Hollunder, J. Müller, B. Nebel, W. Nutt and H. Profitlich, Terminological knowledge representation: a proposal for a terminological logic, DFKI Report, DFKI, Saarbrücken (1992).

[4] A. Borgida, Description logics in data management, *IEEE Trans. Knowledge Data Eng.* **7** (1995) 671–682.

[5] A. Borgida and R.J. Brachman, Loading data into description reasoners, in: *Proceedings ACM SIGMOD Conference on Data Management*, Washington, DC (1993) 217–226.

[6] R.J. Brachman, A structural paradigm for representing knowledge, Ph.D. Thesis, Division of Engineering and Applied Physics, Harvard University, Cambridge, MA (1977).

[7] M. Buchheit, M. Jeusfeld, W. Nutt and M. Staudt, Subsumption between queries in object-oriented databases, *Inf. Syst.* **19** (1) (1994) 33–54.

[8] J. Cai, M. Fürer and N. Immerman, An optimal lower bound on the number of variables for graph identification, in: *Proceedings 30th IEEE Symposium on Foundations of Computer Science* (1989) 612–617.

[9] F.M. Donini, M. Lenzerini, D. Nardi and A. Schaerf, A hybrid system with Datalog and concept languages, in: *Trends in Artificial Intelligence*, Lecture Notes in Artificial Intelligence **549** (Springer, Berlin, 1991) 88–97.

[10] D. Gabbay, Expressive functional completeness in tense logic, in: U. Mönnich, ed., *Aspects of Philosophical Logic* (Reidel, Dordrecht, Netherlands, 1981) 91–117.

[11] N. Immerman, Upper and lower bounds for first-order expressibility, *J. Comput. Syst. Sci.* **25** (1982) 76–98.

[12] N. Immerman, Relational queries computable in polynomial time, *Inf. Control* **68** (1986) 86–104.

[13] P. Kolaitis and M. Vardi, On the expressive power of Datalog: tools and a case study, in: *Proceedings 9th ACM Symposium on Principles of Database Systems*, Nashville, TN (1990) 61–71.

[14] H.R. Lewis, *Unsolvable Classes of Quantificational Formulas* (Addison-Wesley, Reading, MA, 1979).

[15] R.M. MacGregor, A deductive pattern matcher, in: *Proceedings AAAI-87*, Seattle, WA (1987) 403–408.

[16] M. Mortimer, On languages with two variables, *Z. Math. Logik Grundl. Math.* **21** (1975) 135–140.

[17] B. Nebel, Terminological cycles: semantics and computational properties, in: J. Sowa, ed., *Principles of Semantic Networks* (Morgan Kaufmann, Los Altos, CA, 1991) 331–362.

[18] P.F. Patel-Schneider, Undecidability of subsumption in NIKL, *Artif. Intell.* **39** (1989) 263–272.

[19] P.F. Patel-Schneider and B. Swartout, Description logic knowledge representation system specification from the KRSS group of the ARPA knowledge sharing effort, AT&T Bell Laboratories Report, Murray Hill, NJ (1994).

[20] K. Schild, A correspondence theory for terminologic logics, in: *Proceedings IJCAI-91*, Sydney, Australia (1991).

[21] K. Schild, The role of domain-closure axioms and complete knowledge in enabling decidable and tractable reasoning in description logics, in: K.R. Apt and E. Marchiori, eds., *Deliverables for Year 3 of ESPRIT BRA Compulog 2* (1995).

[22] J.G. Schmolze and D. Israel, KL-ONE: semantics and classification, in: *Research in Knowledge Representation for NL Understanding*, Tech. Rept. 5421, BBN Laboratories (1983).

[23] A. Tarski and S. Givant, *A Formalization of Set Theory without Variables*, AMS Colloquium Publications (American Mathematical Society, Providence, RI, 1987).

[24] W.A. Woods and J.G. Schmolze, The KL-ONE family, *Comput. Math. Appl.* **23** (2–5) Special Issue on Semantic Networks in Artificial Intelligence.

ELSEVIER

Artificial Intelligence 82 (1996) 369–380

Artificial Intelligence

Book Review

Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach* ★

Nils J. Nilsson

Robotics Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305, USA

1. Introductory remarks

I am obliged to begin this review by confessing a conflict of interest: I am a founding director and a stockholder of a publishing company that competes with the publisher of this book, and I am in the process of writing another textbook on AI. What if Russell and Norvig's book turns out to be outstanding? Well, it did! Its descriptions are extremely clear and readable; its organization is excellent; its examples are motivating; and its coverage is scholarly and thorough! End of review? No; we will go on for some pages—although not for as many as did Russell and Norvig.

In their Preface (p. vii), the authors mention five distinguishing features of their book: Unified presentation of the field, Intelligent agent design, Comprehensive and up-to-date coverage, Equal emphasis on theory and practice, and Understanding through implementation. These features do indeed distinguish the book. I begin by making a few brief, summary comments using the authors' own criteria as a guide.

- Unified presentation of the field and Intelligent agent design. I have previously observed that just as Los Angeles has been called "twelve suburbs in search of a city", AI might be called "twelve topics in search of a subject". We textbook authors have tried stuffing the hodge-podge that is AI into alternative Procrustean beds: ones based on problem solving, production rules, search, or logic, to name just a few of the possible organizing themes. Each accommodated only a part of its over-sized occupant. Russell and Norvig (henceforth abbreviated to R&N), use what I think is the best theme of all, namely *Intelligent agent design*. Their treatment begins by describing desired abilities of an intelligent agent, and the rest of the book shows how these abilities might be realized. Along the way, the reader is taught the technologies that are most relevant to agent design: search, logic, knowledge representation and reasoning, planning and acting, reasoning and acting under

★ (Prentice Hall, Englewood Cliffs, NJ, 1995); xxviii + 932 pages.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
PII S0004-3702(96)00007-0

uncertainty, learning, generating and understanding communicative acts, machine vision, and robotics. Although the design of intelligent agents provides a unifying motivation for these technologies, the reader who is interested in other applications of AI can find plenty of material and examples that stray from the theme of robots and softbots.

- Comprehensive and up-to-date coverage. Any book of over 900 pages better be comprehensive, and this one does not disappoint. It includes some topics that arguably could have been left out of an introductory text and fails to cover some that perhaps should have been covered. And, the level of coverage is a bit uneven. For example, the subject of truth maintenance (pp. 325ff) is covered at about the level of detail that might be expected in an introductory text—a nice summary but nothing too deep. On the other hand, R&N go into probably more-than-needed detail on such subjects as dynamic decision networks and chart parsing, to name just two examples.

  The student who completes the two-semester course required to assimilate all of this material probably will know more than do many practitioners. (R&N painlessly filled in gaps in my own knowledge resulting from my inattention to some of the recent literature.) Adding any additional material would have risked Knuthian fission of the book into several volumes. The purposes of some readers might be better served by a shorter introductory book accompanied by one of a series of more detailed volumes on the many topics covered. In recognition that the entire book covers the union of many needs, R&N prescribe (on p. ix) seven different subsets of chapters appropriate for one-semester or one-quarter courses. Don't be frightened by the length—some material can be omitted.
- Equal emphasis on theory and practice. Underlying both *theory* (i.e., definitions, theorems, analysis) and *practice* (i.e., programs, implementational tricks, engineering lore) is the vastly more important body of knowledge that comprises the key *ideas* of a field. For example, the concept of a problem space and its representation as a graph structure that can be searched by heuristic methods are important ideas in artificial intelligence. Built on these ideas is all the theoretical apparatus of admissibility, complexity analysis, and so on. Also stemming from these ideas are the efficient programming methods for search. I could give other examples; in fact the landscape of artificial intelligence is richly populated with ideas developed over the last forty years: declarative representations of knowledge, production rules, Bayes networks, speech acts, neural networks, STRIPS-like rules, partial-order planning, configuration spaces, and so on. Each idea brings forth important theory and implementations. R&N actually concentrate (appropriately, in my opinion) more on the ideas than they do on either the deep theory or on detailed practical matters, and they explain the ideas especially well.

  The book is not heavily theoretical; most of the theory involves statements about the computational complexity of various algorithms. They do provide a proof of the completeness of resolution (pp. 286ff) and a nice outline of a proof of Gödel's Incompleteness Theorem (p. 288). The reader who is not overly intimidated by discrete mathematics, logic, probability theory, and a wee bit of calculus will have no difficulty.

The "practice" part of the book involves discussion of some 1990s AI programs and many useful observations about practical implementations. Including discussions of actual systems is risky in a textbook because such material will soon be dated. But overall, I think R&N have achieved the balance they sought.

- Understanding through implementation. Many of the *ideas* of artificial intelligence involve algorithmic procedures. One cannot understand these algorithms adequately unless they are defined precisely. R&N provide easy-to-understand "pseudo-code" for over 100 algorithms. (Some of the pseudo-code uses English and mathematics to avoid more complex formal locutions.) Occasionally (pp. 152 and 663, are examples), the use of pseudo-code to describe "agents" seems a bit pedantic. R&N have also provided a repository of written Lisp code implementing these algorithms. To receive instructions on how to get this code, they advise (p. 857): " ... send an electronic mail message to `aima-request@cs.berkeley.edu` with the word "help" in the subject line or in the body".

  In addition, R&N exhort readers to write their own programs to search, to unify, to prove, to parse, and so on.

2. Summary of the major parts

The book is divided into eight parts. These are (along with excerpts from R&N's own summary of each):

(1) Artificial Intelligence. "The two chapters in this part introduce the subject of artificial intelligence or AI and our approach to the subject: that AI is the study of *agents* that exist in an environment and perceive and act."

(2) Problem Solving. "In this part we show how an agent can act by establishing *goals* and considering sequences of actions that might achieve those goals. A goal and a set of means for achieving the goal is called a *problem*, and the process of exploring what the means can do is called *search*. We show what search can do, how it must be modified to account for adversaries, and what its limitations are."

(3) Knowledge and Reasoning. "In this part, we extend the capabilities of our agents by endowing them with the capacity for general logical reasoning. A logical, knowledge-based agent begins with some knowledge of the world and of its own actions. It uses logical reasoning to maintain a description of the world as new percepts arrive, and to deduce a course of action that will achieve its goals."

(4) Acting Logically. "... [here, we] build *planning agents*. At the most abstract level, the task of planning is the same as problem solving. Planning can be viewed as a type of problem solving in which the agent uses beliefs about actions and their consequences to search for a solution over the more abstract space of plans, rather than over the space of situations. Planning algorithms can also be viewed as special-purpose theorem provers that reason efficiently with axioms describing actions."

(5) Uncertain Knowledge and Reasoning. "... we reexamine the very foundation of the logical approach, describing how it must be changed to deal with the often

unavoidable problem of uncertain information. *Probability theory* provides the basis for our treatment of systems that reason under uncertainty. Also, because actions are no longer certain to achieve goals, agents will need ways of weighing up the desirability of goals and the likelihood of achieving them. For this, we use *utility theory*. Probability theory and utility theory together constitute *decision theory*, which allows us to build rational agents for uncertain worlds."

(6) Learning. "Whenever the designer has incomplete knowledge of the environment that the agent will live in, learning is the only way that the agent can acquire what it needs to know. Learning thus provides *autonomy* in the sense defined in Chapter 1. It also provides a good way to build high-performance systems—by giving a learning system experience in the application domain."

(7) Communicating, Perceiving, and Acting. "In this part, we concentrate on the interface between the agent and the environment. On one end, we have *perception*: vision, hearing, touch, and possibly other senses. On the other end, we have *action*: the movement of a robot arm, for example.

"Also covered in this part is *communication*. A group of agents can be more successful—individually and collectively—if they communicate their beliefs and goals to each other. We look more closely at human language and how it can be used as a communication tool."

(8) Conclusions. "In this part we summarize what has come before, and give our view of what the future of AI is likely to hold. We also delve into the philosophical foundations of AI, which have been quietly assumed in the previous parts."

In addition to introductions to each part, R&N begin each of the twenty-seven chapters with a summary paragraph or two, preceded by "pooh-like" sentences. The first of these goes, "*In which we try to explain why we consider artificial intelligence to be a subject most worthy of study and in which we try to decide what exactly this is, this being a good thing to decide before embarking.*" The last goes, "*In which we take stock of where we are and where we are going, this being a good thing to do before continuing.*"

3. Special features and highlights

At the end of each chapter is a collection of bullets summarizing the main topics, ideas, and results. The bold-faced terms in some of these summaries are a useful checklist for the student.

Following the chapter summaries are bibliographic and historical notes. These notes are scholarly and exceedingly well researched (by Douglas Edwards). I learned a great deal from them and noted only an occasional minor error. The reader will be treated to several interesting (if not always compellingly relevant) pointers, such as: "*Dung Beetle Ecology* (Hanski and Cambefort, 1991) provides a wealth of interesting information on the behavior of dung beetles". Many historical and bibliographic comments occur sprinkled throughout the body of the book as well.

The margins of the book are judiciously populated with terms and phrases occurring in the adjacent text; these might be helpful for locating topics looked up in the extensive index. Especially important passages are noted with a finger-pointing icon.

The index is very thorough (28 pages) and helpful. Of my many uses of it, I was only foiled once: when I attempted to find something about language generation, I soon tried "generation (of language)". There, I was told to "*see* natural language, generation", where I was told to "*see* language, natural", where there was no mention of "generation".

Exercises at the end of each chapter vary from technically sharp to tedious. Here are examples in each category:

"Show that convolution with a given function f commutes with differentiation, that is, $(f * g)' = f * g'$." (p. 771).

"Read this chapter from the beginning until you find ten examples of homophones." (p. 772).

Many of the exercises expand on examples presented in the text. Probably the list will evolve as more experience is gained with the book in courses.

The book is full of interesting insights, and fresh and compelling illustrations. To give the reader a feeling for proprioceptive feedback, for example, R&N suggest an interesting eyes-closed experiment which illustrates that human finger-positioning repeatability (based on proprioception alone) is better than finger-positioning accuracy.

Helpful footnotes abound; on p. 314, for example, we learn that "Rete is Latin for net. It rhymes with treaty." There are occasional frivolous ones as well (see p. 476).

Many of their explanations connect to topics already known to the reader. For example (p. 266), based on the statement "there is a number that ... satisfies the equation $d(x^y)/dy = x^y$...", R&N point out that the name we use for this number is e—establishing the fact that the mysterious Skolem constants soon to be introduced are really already quite familiar. Several helpful analogies are based on computer science ideas. For example, they compare (on p. 379) a "critic" in a hierarchical planner to a "peephole optimizer" in a compiler.

And here (p. 190) is a technical point I hadn't thought about very much: In $(\forall x)[Cat(x) \supset Mammal(x)]$, quantification ranges over *all* objects in the domain—not just over cats!

There are very few typographical errors. The book's www page (see below) lists those in the first and second printings that are known to the authors. Here are two treasure-hunt challenges, not on the www page, for the careful reader: find the places where R&N use "vary" when they meant "very" and "Finish" when they meant "Finnish".

The book also has its own world-wide web home page. It's URL is: http://www.cs.berkeley.edu/ russell/aima.html. Here is what is available through this page (words in italics are hot links):

- *Ordering*: how to get a copy of the book. Or check your *local bookstore*. If you are a professor teaching AI you can *order a sample copy*.
- *Brief* and *complete* tables of contents.
- *Preface*: Excerpts from the preface.
- *Comments*: Comments on the book by various instructors, students, and other reviewers.
- *Code*: How to get Lisp programs that support the material in the book.
- *Index to code*: Pages and files where all functions, types and variables appear.
- *Errata*: typos and other errors that have been spotted by alert readers.
- *Clarifications*: Clarifying remarks and additional material.

- *Cover*: A large (590K) version of the cover image.
- *Surveys*: Results of student questionnaires from several schools.
- *Other books*: The Prentice Hall Series in Artificial Intelligence.
- *Other courses*: Home pages for introductory AI courses at various universities.
- *Other sites*: Other servers on the web with information on AI.

4. More detailed comments on content and organization

In a book of this length and depth, an experienced reader of even the most carefully crafted book will find a number of places where he or she feels compelled to make some minor marginal notes. Sometimes this interruption by the "trees" interferes with sizing up the "forest". In the following, I'll give a chapter-by-chapter discussion of my view of the forest interspersed with comments about some of the trees that arrested me.

Part I: Artificial Intelligence

In Chapter 1, R&N begin with a discussion of the foundations and history of AI. This chapter is very well researched—going all the way back to Plato. I would have mentioned the subsumption architecture, situated automata, and the so-called "animat approach" to AI in the section dealing with recent history. These topics have influenced the field at least as much as have neural networks, which were cited.

For additional interesting historical material about the development of AI, the reader might want to see the histories written about the early days of each of four major AI labs [1,6–8].

In Chapter 2, AI is defined as the study of "intelligent agents". Agent complexity varies along a spectrum; some agents merely react to stimuli, some react to a combination of stimuli and internal state, and some take explicitly provided goals into account. Those capable of "high-quality behavior" are "utility-based agents", which compute what to do based on knowledge of the environment and its dynamics, knowledge of the effects of actions, and the utility to the agent of various environmental states. This spectrum provides a rough guide for the order of presentation of topics in the book.

Part II: Problem-Solving

Postponing treatment of simple reactive agents until later in the book, R&N proceed directly in Chapter 3 to consider what they call *problem-solving agents*. Such agents search a *problem space* using state descriptions, operators, and goal descriptions. The chapters in this part present up-to-date material on uninformed and heuristic search, including game-tree search. Although R&N adhere to their agent-based approach by using some robot-like illustrative examples, they also employ standard ones like the 8-puzzle.

I think that R&N were overly impatient to get into "real AI" by beginning with problem-solving agents instead of with simple reactive ones. Much of the material that

occurs later in the book (some visual processing, neural net learning, production-rule and subsumption-based action computation, and situated automata) could have been dealt with first; that order of treatment would have correlated nicely with the agent complexity spectrum. Situated automata, for example, constitute a fundamental topic in AI, yet they are mentioned only in Chapter 25—a chapter that occurs in only one of R&N's recommended seven alternative one-quarter or one-semester syllabi.

Part III: Knowledge and Reasoning

After dealing with AI search methods, R&N segue brilliantly, in Chapter 6, into logic and its methods for knowledge representation and reasoning. Using the example of an agent in the "wumpus world", they show how manipulating knowledge about the world can be used to augment an agent's immediate sensations to help it compute appropriate actions. They introduce many of the needed concepts using propositional logic, and then, in Chapter 7, elegantly show how first-order logic greatly reduces the representational burden.

In my opinion, their use of *semantics* in Chapter 6 to help introduce and motivate *syntax* risks conflating these two quite separate notions—which shouldn't really be joined until the propositional truth table is described. The syntax-semantics distinction is just one of many that must be observed in the study of logic. Another, which isn't adequately stressed, is the difference between meta-symbols used in statements *about* logic, such as "$\vdash$" and "$\models$" and linguistic symbols, such as "$\Rightarrow$", used in logical sentences.

R&N (unfortunately, I think) introduce the situation calculus and the frame problem in Chapter 7 rather than waiting until their chapters on planning. Although there is still debate about whether *reasoning* about actions using the situation calculus can be made efficient enough for an agent to *compute* which actions to perform, the approach taken later in Part IV suggests that reasoning be limited to establishing what is true in each world state and that extra-logical methods be used for computing a plan of action to traverse the states.

In the section of Chapter 7 dealing with extensions to logic (pp. 194ff), R&N might have mentioned modal logics.

One of the high points of the book is Chapter 8, *Building a Knowledge Base*. R&N use the domain of digital circuits to illustrate how knowledge of specialized subject matter can be represented in first-order logic. This example permits them to describe the process of deciding "what to talk about", what should be the "vocabulary", how to "encode general rules", how to encode the "specific instance" of the problem at hand (a full adder with carry), and how to "pose queries to the inference procedure" (which latter, by the way, the reader hasn't heard about yet). This chapter also deals with the following important notions: categories, measures, composite objects, time, space, change, events, processes, physical objects, substances, mental objects, and beliefs. Applications of these ideas are illustrated in a "grocery shopping world".

One of the advantages claimed for sentences in knowledge bases is that they are "self-contained" or context independent (p. 222). The *meanings* of sentences, though, are context dependent in that the set of models of a knowledge base depends on all of the sentences in it; models can be eliminated by the addition of new sentences.

I applaud R&N's bravery in attempting this chapter—a lot of material rolled into less than 50 pages. Personally, I would have advised them to get on with the fundamentals of inference, planning, and uncertain reasoning and then cover the more advanced topics (such as events, time, processes, and substances) later in the book. The grocery shopping example could still have been introduced in simplified form at this stage and then reconsidered later to illustrate the use of additional technical apparatus.

The reader is likely to be confused by the paragraph at the bottom of p. 237. There, the expression $T((At(Agent, Loc_1) \wedge At(Tomato_1, Loc_1)), I_3)$ is pointed out as flawed because a sentence appears as the first argument of the predicate T. Yet later, after introducing the function *And*, the expression $T(And(At(Agent, Loc_1), At(Tomato_1, Loc_1)), E)$, is held up as OK, where $At(Agent, Loc_1)$ has suddenly now changed from a sentence into an "event category". That went by a bit fast.

Next, in Chapter 9, R&N deal with inference, unification, and resolution. They make what I think is the unfortunate choice of using "implicative normal form" rather than conjunctive normal form for the clauses used in resolution refutations. Although implications are perhaps more understandable than clauses, the alleged pedagogic value of this appealing and familiar form is sullied by the necessity of a more complex resolution rule and the need to remember that antecedents are conjunctions and consequents are disjunctions. (Matt Ginsberg succumbed to the same temptation in his text *Essentials of Artificial Intelligence*).

R&N state (on p. 281) that each step in the conversion of a sentence to normal form does not change the meaning of a sentence. That's not strictly true of Skolemization because $(\exists x)P(x)$ doesn't logically entail its Skolem form, $P(Sk)$. What is true is that a set of formulas, Δ, is satisfiable if and only if the Skolem form of Δ is. Or, more usefully for purposes of resolution refutations, Δ is unsatisfiable if and only if the Skolem form of Δ is unsatisfiable [4].

As regards collecting substitutions during a resolution refutation in order to provide "who" or "what" answers, R&N make no mention of the "answer literal" construction [5], which provides a nice way of maintaining a "... composed unifer so that a solution is available as soon as a contradiction is found" (p. 283).

In Chapter 10, R&N lump together theorem provers, logic programming languages, production systems and the rete algorithm, semantic networks, terminological logics, and truth maintenance systems. It's unclear what thread links these disparate topics. I probably would have parcelled them out to other chapters.

There is little or no mention of deductive databases and the connections between research in that area (magic sets, negation as failure, stratification) and logic programming languages discussed in this chapter. [Although, R&N do cite [9] in the historical section (p. 328).]

It ought to be pointed out that "metareasoning" (p. 309) is important beyond the realm of logic programming.

Part IV: Acting Logically

Without spending much time on the situation calculus (already prematurely mentioned in Chapter 7) or STRIPS-style planners, R&N move directly to "plan space" representa-

tions in Chapter 11. Their discussion of partial-order planning (POP) systems is well motivated and clear, illustrated by a nice example loosely based on the grocery shopping world.

The footnote on p. 361 that correctly points out that "Shakey was never dextrous enough to climb on a box or toggle a switch", should have been used to qualify their sentence on p. 360, which claimed "Shakey can ... climb on and off rigid objects (such as boxes), and turn light switches on and off".

In Chapter 12, they extend their discussion of planning to include several important "scaling-up" topics: hierarchical planners, operators with conditional effects, and the use of "resources". I would have included hierarchical planning in Chapter 11 devoted to basic ideas rather than in one concentrating especially on "practical planning".

The statement (p. 371) that "SIPE ... was the first planner to deal with the problem of replanning, ..." is inaccurate; both STRIPS/PLANEX and NOAH did that.

Chapter 13, "*In which planning systems must face up to the awful prospect of actually having to take their own advice*", deals with matters involved in executing plans: the use of run-time conditionals in plans, execution monitoring, and replanning.

Part V: Uncertain Knowledge and Reasoning

In Chapter 14, R&N confront the difficulty that most knowledge is uncertain. They give a good account of the basic ideas of probability theory including a discussion of its axiomatic basis, conditional and joint probabilities, and Bayes' rule. With this foundation, they then move on in Chapter 15 to present Bayes belief networks—structures that graphically stipulate conditional independencies which greatly simplify representations of joint probabilities. Some of the more complex mathematical material in the book, that dealing with inference in Bayes nets, is presented here (although no mention was made of techniques based on "arc reversal"). The reader will no doubt need to read these pages two or more times (as did I). More examples of the use of these methods would have been welcome. Dempster–Shafer theory and fuzzy logic are briefly treated. R&N mention default reasoning (nonmonotonicity, etc.) under "Other Approaches to Uncertain Reasoning". Perhaps that topic should have been taken up back in the logic chapters. They do not discuss the closed-world, unique names, nor domain-closure assumptions.

This part concludes with two chapters on the use of probabilities and utilities to make decisions. Material covered in these two chapters includes: rational decision making, utility theory, decision networks, the value of information, dynamic programming, value and policy iteration, and dynamic belief and decision networks. Although these topics seem fashionable in AI these days, I think they could have been safely omitted from an introductory text.

Part VI: Learning

As is often done, R&N treat "Learning" as a separate AI topic. To emphasize the point that learning is or could be used almost everywhere in AI systems, I would have spread discussion of learning methods throughout the book—treating neural nets, decision trees,

decision lists, reinforcement learning, and genetic algorithms in chapters dealing with reactive agents. Whatever needs to be said in an introductory book about PAC learning could be mentioned back in those chapters also. Inductive logic programming could be moved into a chapter dealing with representing knowledge by sentences in first-order predicate calculus, and explanation-based learning could occur either in those chapters or in ones on planning. Notwithstanding that personal opinion, R&N give excellent descriptions of the major techniques in machine learning—including much post-1990 material on reinforcement learning and on learning Bayes networks.

Part VII: Communicating, Perceiving, and Acting

This part is the grand finale of the book. Agents that do none of these things can hardly be called agents. After briefly explaining varieties of "speech acts", R&N do an outstanding two-chapter job of describing natural language processing. They readily acknowledge the difficulty of capturing human language in grammars—even so, grammars are "useful tools for dealing with some aspects of natural language". R&N concentrate on definite clause grammars and their augmentations—first to enforce number and case agreements and second to implement the construction of the meaning of a sentence compositionally from the meaning of its components. Their strategy is to rely on compositional semantics for the context-free aspects of language and then to use pragmatics to resolve ambiguities. One chapter introduces the important ideas, and a second concentrates on the practical matters inherent in real-world applications. Among the topics covered in the "scaling-up" chapter are: database and information retrieval applications, chart parsers, more complex grammars, dealing with various forms of ambiguity, and discourse understanding.

In the historical section of Chapter 22, they might have mentioned LIFER [3] as an early example of a semantic grammar.

Even though the phrase "natural language processing", usually includes both "understanding" and "generation", there is no mention of generation methods in the book, outside of a brief discussion of generating speech acts.

Chapter 24 on perception concentrates mainly on machine vision but also includes material on speech recognition. (R&N credit Jitendra Malik with writing most of this chapter.) Basic material on image formation, image processing operations (averaging and edge detection), and methods for extracting 3-D information is clearly presented. I found the description of convolution and of the Canny edge detector particularly lucid. As is appropriate for a text based on intelligent agents, emphasis is placed on the *uses* of vision rather than on abstract "scene analysis". These uses are manipulation, navigation, and object recognition. There is a nice example on pp. 749 and 750 that clearly shows that "*for a specific task, one does not need to recover all the information that in principle can be recoverd from an image*".

The pages on speech recognition use Bayes' rule to break the problem into two components: one using a language model and the other an acoustic model. R&N approximate the language model by bigram and trigram statistics. Hidden Markov models are used for the acoustic model (of word pronunciation). As I mentioned earlier, except perhaps for speech recognition, this chapter on perception could have appeared earlier in the

book. And, one could question why machine vision and speech recognition are in the same chapter.

The final chapter of this part is entitled "Robotics" (written mainly by John Canny). Fundamental ideas on sensing and effecting are described including the concepts of degrees of freedom, holonomism, rotary and prismatic motion, proprioception, force, tactile, and sonar sensing, and laser range finders. Robot architectures, "classical" and behavior-based, are briefly described. These architectural matters might have been more appropriately discussed in Chapter 13, Planning and Acting. The chapter concludes with a presentation of configuration spaces and their uses in navigation and motion planning.

Part VIII: Conclusions

Chapters 26 and 27 treat the reader to some philosophical and strategic big questions: Can machines think? What is consciousness? Have we succeeded yet? What exactly are we trying to do? What if we do succeed? In their discussion of consciousness, R&N concentrate on various things John Searle has had to say about it. Surprisingly, they don't refer to [2].

I was glad to see that these questions were addressed even though there are no real answers presently available.

Appendices A and B describe complexity analysis, O( ) notation, Backus–Naur form, the pseudo-code conventions, and the repository of Lisp code.

5. Final remarks

The study of AI requires near-renaissance universality. The components needed to mechanize intelligence fill an entire engineering curriculum; small wonder this book is so large. I am awed by the amount of work and study that must have gone into its writing. The authors (and their helpers) have done a remarkable job, and the field owes them a hearty thanks and "well done!". There are some major topics left out (how can that be in a book so thick?)—among them qualitative physics and a full discussion of "tractable" reasoning. There are some minor flaws; rather, there are things you or I might have done somewhat differently. On the www pages for the book, R&N state: "The second printing fixes typos on 68 pages, and leaves the other pages unchanged. ... (Look for a second edition in 1997 or 1998, and see the list of *clarifications* for some things that will be addressed in the second edition but were too costly to fix in a reprinting.)"

This book, and its promised second edition, will deservedly dominate the field for some time.

References

[1] B. Buchanan, Introduction to the COMTEX microfiche edition of memos from the Stanford Artificial Intelligence Laboratory, *AI Magazine* **4** (4) (1983) 37–42.

[2] D. Dennett, *Consciousness Explained* (Little, Brown and Company, Boston, MA, 1991).

[3] G. Hendrix, Human engineering for applied natural language processing, in: *Proceedings IJCAI-77*, Cambridge, MA (1977).
[4] D.W. Loveland, *Automated Theorem Proving: A Logical Basis* (North-Holland, New York, 1978) 41ff.
[5] D.C. Luckham and N.J. Nilsson, Extracting information from resolution proof trees, *Artif. Intell.* **2** (1971) 27–54.
[6] M. Minsky, Introduction to the COMTEX microfiche edition of the early MIT Artificial Intelligence Memos, *AI Magazine* **4** (1) (1983) 19–22.
[7] A. Newell, Introduction to the COMTEX microfiche edition of reports on artificial intelligence from Carnegie-Mellon University, *AI Magazine* **5** (3) (1984) 35–39.
[8] N.J. Nilsson, Introduction to the COMTEX microfiche edition of the SRI Artificial Intelligence Center Technical Notes, *AI Magazine* **5** (1) (1984) 41–52.
[9] J. Ullman, *Principles of Database and Knowledge-Base Systems* (Computer Science Press, Rockville, MD, 1988/1989) Volumes I/II.

ELSEVIER

Artificial Intelligence 82 (1996) 381–391

Artificial Intelligence

Book Review

Patricia S. Churchland and Terrence J. Sejnowski, *The Computational Brain* ★

George N. Reeke Jr *

The Rockefeller University, 1230 York Avenue, New York, NY 10021, USA

1. Overview

The Computational Brain represents a notable attempt to summarize the current status of research on computational models of brain function at an introductory level. As we might expect from the authors' backgrounds, the work integrates topics in philosophy, neuroscience, and connectionist modelling, and it will therefore be reviewed in all three areas after an outline of the contents.

The book begins with a background chapter on neuroscience. Following Shepherd [31], the authors lay out seven levels of description (behavior, systems and pathways, local circuits, neurons, microcircuits, synapses, membranes and molecules) that must be considered to understand the brain. The question, "What are the most basic structural features relevant to neural computation?" is asked, and answered with a list of likely "must know" items that includes such things as specialization of function in regions of the nervous system, numbers and connectivity of neurons and synapses, spiking, timing of neural signals, receptive fields, and parallel architecture. This is followed by a description of some computational organizations that have been proposed as models for neural systems. These include linear associators, Hopfield networks, Boltzmann machines, competitive learning and back-propagation, feedforward and recurrent nets. The authors close this section with a strong statement that the method of parameter adjustment in a model network is not important so long as the model network and the real network "end up at locations on the error surface that are rather close" (p. 131). This assertion permeates the authors' entire approach to the subject in a way that I find most troubling, and I shall return to it at some length below.

Churchland and Sejnowski go on to explore the problem of representations, taking examples mainly from vision to describe concepts such as vector coding and state

★ (MIT Press, Cambridge, MA, 1992); xi, 544 pages, $39.95.

* E-mail: reeke@lobimo.rockefeller.edu.

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
PII S0004-3702(96)00008-2

spaces. Shape from shading, stereopsis, and hyperacuity are discussed as paradigm neural computations. Additional examples designed to illustrate neural plasticity, development, and sensorimotor integration constitute the remaining body of the text. Approximately a dozen models are treated in some detail, with plasticity in the hippocampus, the vestibulo-ocular reflex, and the generation of segmental motions in the leech and lamprey being picked out for special attention. The book ends with a good discussion of some of the ethical issues that will arise if we ever succeed in understanding the neural basis of the mind. This is followed by a brief appendix on neurobiological lab techniques. Throughout the book, mathematical and programming details are mostly omitted. The reader interested in the question of how symbolic processing of the kind studied in Artificial Intelligence might be implemented in connectionist models (see, e.g. Smolensky [32]), and more generally in the relationship between AI and computational brain models, will find almost no discussion of these topics.

The summary of neurobiology is generally understandable and accurate, although the electrophysiology of the neural membrane is deferred to a few pages near the end of the book. This material should be better integrated with the rest of the text, because we need to understand it in order to formulate criteria for evaluating more abstract models [19, pp. 31-34]. Synaptic plasticity and, in particular, long term potentiation (LTP) are emphasized as possible mechanisms of memory, with a clearer differentiation than one usually sees between synaptic change, which is a local phenomenon, and learning, which requires global coordination. The computational overview is also very germane. The authors explain why no brain model can be totally realistic and advise as a working rule that models operating at a particular level of organization would do well to incorporate constraints from one level above and one level below the one under consideration, but probably not more. Other workers will prefer to encompass as many levels as possible in more comprehensive models.

The book is most successful at portraying some of the current excitement in this relatively new field. The models are often multidisciplinary and cover a wide variety of interesting phenomena. They are discussed with due attention to their limitations as well as their successes, and the need for real-world testing to avoid the snares of oversimplified toy worlds is well emphasized. The failure of most current models to perform more than one behavioral task (a key capability of real nervous systems that modelers usually ignore) is rightly pointed out. It should, however, be noted that the selection of material reflects the authors' frankly admitted predilection for models developed in California, and especially in the San Diego area (Preface p. x). Some important contributions from other places are omitted or mentioned only in passing, for example the ART theory of Grossberg [12].

In as much as the major thrust of the book is to present a computational analysis of brain function, I will deal now with certain fundamental issues that I believe any theory of brain function must address, but which Churchland and Sejnowski mostly ignore. This discussion, which is somewhat philosophical in content, is followed by comments on the treatments of neuroscience and of computational models in the book.

2. Treatment of mind as computation

As the title suggests, *The Computational Brain* is primarily a manifesto for the currently popular proposition that the mind can be explained as a form of computation, expanding on Sejnowski, Koch, and Churchland's earlier treatment of the same topic [29]. I use the term "manifesto" because the main proposition is argued by assertion, then accepted implicitly as the basis for a program of research. In my view and that of many other authors (Edelman [9], Searle [28], Bickhard [4], Lakoff [18], Johnson [14], Winograd [38] to name a few), this proposition is quite untenable. Yet, the highly salient criticisms of these authors are for the most part not rebutted or even acknowledged. For example, Edelman is mentioned in connection with an obscure proposal about synaptic weights, but his major work on neural Darwinism is never discussed. The authors evidently believe that if computationalism is presented as straightforward, uncontroversial science, it will become just that. If readers fail independently to notice the deep questions about the explanatory power of the approach that have been raised by others, then these questions effectively do not exist. Toward this end, the philosophical sections of the book tend to deal with relatively uncontroversial issues such as materialism versus dualism, and do not ask questions about the nature of scientific explanation that are particularly critical as science begins to grapple with the subjective experiences of conscious beings. Perhaps the authors consider criticisms of computationalism to be beneath reply, but the reader deserves better. At the risk of being tedious (for I have discussed this topic previously in [24] and elsewhere with Edelman [25]) I will give a brief analysis of why computation, at least as usually defined, cannot explain the mind. Then I will consider whether this analysis can be vitiated, as Churchland and Sejnowski appear to believe it can, by defining the term "computation" sufficiently broadly.

The fundamental question is, "How does a computation come to have a meaning?" The concept of computation generally comprises the purposeful manipulation of encoded representations of information to yield some desired result. The purpose of the manipulations (algorithms) and the meaning of the representations (symbols) are not contained within current computational systems, but in the mind of an external observer (typically, a programmer).[1] The critical question is whether this observer's mind can itself be a computer that has somehow escaped from the infinite regress of observers implied by the above definition, perhaps by some principle of "self-organization". For this to have happened, this computer would somehow have to have acquired the ability to interpret representations in terms of things that are not themselves representations [4]. Such a device would no longer be a formal symbol processing system. It would be unlike any computer ever built, analog or digital, in as much as all such devices operate upon representations that have no meaning in themselves, but only as interpreted by outside observers. Brains, on the other hand, construct representations which have meaning that is directly apprehended as subjective experience. They construct causal relationships between states of affairs in the world and appropriate, responsive behaviors—relationships that are not impressed from the outside and that do not require the interpretation of any observer to be efficacious.

[1] This fact was emphasized by J.R. Searle in his recent H.W. Smith lectures at New York University.

A particularly critical concept for an understanding of this point is the *mapping*. Churchland and Sejnowski discuss mappings as "lookup" functions that can quickly compute any arbitrary function, which may be defined explicitly or implicitly as a relationship between a sufficient set of given inputs and the corresponding correct outputs. But what, really, does this description tell us about the essential business of the brain? If the functional relationship is known a priori, or can be detected from statistical correlations between the inputs and outputs, then a conventional computer or "neural network" can indeed encode the mapping to any desired degree of accuracy. But what such a device does is relate one set of representations to another set of representations. It does not give meaning to its representations or have insights about the things they represent. What we most need to know about the brain is what kind of mappings it has, how they come to refer to particular objects and events in the world, how they support conscious awareness of those objects and events, and how they give rise to adaptive behavior and the ability to respond to novelty. In order to escape the need for an observer to define representations, we must put the observer *inside*, not *outside* the system. To even begin to make progress, we must consider the total system of a biological organism and its environment, and investigate the development of its behavior, not just its mature performance. We must test our models with tasks that reflect real-world variability, not with a priori fixed input/output pairs.

These problems also exist for rule-based AI. Connectionism has been presented as an alternative that avoids the need for explicit programming, but it will be well known to readers of this journal that rule-based systems can also be designed to acquire mappings through training. More fundamentally, both approaches share the belief that representational systems can have intrinsic semantics. In both approaches, the subjective observer is not only unnecessary, but in fact can only be understood as an epiphenomenon of an underlying computation (see, e.g. [23]). This theory of semantics is part and parcel of the computationalist *Weltanschauung*, and it is in this fundamental sense that Churchland and Sejnowski's approach is indeed computational. However, this theory of semantics fails to explain subjective awareness. For those to whom the conscious observer is an entity whose real existence must be acknowledged in scientific theories of mind, a different theory of semantics is needed. Computation, even if framed in terms of "self-organizing systems" and the like, is not enough.

Churchland and Sejnowski, however, ignore these important issues and choose the simpler course of defining away the problem by broadening the definition of computation to include whatever operations are required to support the functioning of the mind. To permit the exact nature of these operations to be specified later, they are careful to leave their definition of "computation" open to future modification as more is learned about the brain: "Notice in particular that once we understand more about what sort of computers *nervous systems* [authors' emphasis] are, and how they do whatever it is they do, we shall have an enlarged and deeper understanding of what it is to compute and represent." (p. 61) This circular definition, which is very similar to one made by Minsky [23], allows us to call the brain a computer if that is what we want to do, but it does not automatically make the results of either axiomatic or empirical computer science applicable to brains.

In fact, adoption of this approach would force us to maintain two theories under

the same name: let us distinguish them as "Computation Theory 1" (the theory of ordinary computers) and "Computation Theory 2" (the theory of brain-computers). Once Churchland and Sejnowski have equated brains with computers, they observe that the computers we are familiar with are quite unlike brains, and thus quite unlike the real computers that are yet to come (and that we don't now know enough about to describe). Thus, on p. 7, "what is being modelled by a computer [referring to the brain] is itself a kind of computer, albeit one quite unlike the serial,[2] digital machines on which computer science cut its teeth. That is, nervous systems and probably parts of nervous systems are themselves naturally evolved computers—organically constituted, analog in representation, and parallel in their processing architecture." In other words, the shortcomings of existing computers in exemplifying mind exist just because they are still so primitive, not because they are fundamentally different. This is a *Through the Looking Glass*[3] approach because it uses the word "computation" to mean something different from its well established popular and technical meanings. It creates confusion by encouraging inappropriate application of results from Computation Theory 1 to entities (brains) that really should be studied with Computation Theory 2, and vice versa. For example, Computation Theory 1 has only begun to deal with nondeterministic automata, whereas nondeterminism is an essential characteristic of brain responses, with critical implications for the philosophy of mind (see e.g. [3]). While some authors (e.g. [23]) have claimed that volition is a sham, the result of low-level deterministic processes, most would find this solution unacceptable. Ultimately, the equation of brains with computers leaves us with an empty description of both that has no more than metaphorical value.

Of course, it must be acknowledged that a conscious observer may often be able, ex post facto, to describe mental processes as computations, and these descriptions can often be useful in helping us formulate necessary conditions for a particular cognitive process to succeed, as pointed out by Marr [21]. The book at hand contains many such descriptions. Problems arise in conceding causal status to these descriptions, for at least three reasons: First, the description of the brain as a symbol manipulation system cannot account for its most important salient characteristic, the existence of subjective experience. (Dennett [5] and Minsky [23] have tackled this issue, explaining such experiences as essentially epiphenomena emerging from the operations of a complex computational system, but the attempt is not very convincing.) Second, the assignment of meaning to symbols cannot occur inside a formal computational system, as explained in detail above. Finally, because computational descriptions are purely functional, that is, not concerned with the physical instantiation or history of an information processing system, they cannot help us understand either the biological or the experiential development of the brain.

[2] The authors here seem to imply that non-serial (i.e., parallel) computers are somehow different in kind from serial ones, although it is well known that there is only a difference in speed, provided the individual processors have no more capability than Universal Turing Machines.

[3] "'When *I* use a word,' Humpty Dumpty said, in rather a scornful tone, 'it means just what I choose it to mean—neither more nor less.'" [L. Carroll, *Through the Looking Glass, and What Alice Found There* (Macmillan, 1871)].

Churchland and Sejnowski mostly avoid this last, ontogenetic, objection to computationalism by simply not dealing with the origins of mind—they just invoke evolution every time a question of origins arises. This allows them to avoid linking development in models to development in the brain and further allows them to accept models that are known to be ontogenetically incorrect as long as their mature behavior is correct. For them, the experiential history of the individual organism may as well not exist, since it is not considered important to incorporate it accurately into neural models. (For an opposite point of view, see [8, 13]). Similarly, Churchland and Sejnowski almost totally ignore the effect that behavior has, through changes it causes in the environment, upon the categorizations that determine future behavior in what Edelman has called "global mappings" [8]. Their approach is incoherent because it asserts the unimportance of developmental mechanisms at the same time as it declares the primacy of self-organizing computational systems as models for the brain. It does, however, permit the authors to get on with the business at hand without immersing the reader in philosophical thickets. Following their lead, I will also pass over these matters and get on with the business at hand, the neuroscience that is the main subject matter of the book.

3. Treatment of neuroscience

The coverage of neuroscience is generally accurate and focussed on issues of current interest. However, the treatments are not always well balanced or fully up to date. For example, in the area of vision, 26 pages are devoted to the vestibulo-ocular reflex, but only a single paragraph to the cortical oscillations discovered by Gray and Singer [10] and by Eckhorn et al. [6] that have been the inspiration for over twenty published computational models. The visual figure-ground segregation task is exemplified by the 1986 model of Kienker et al. [16]. More realistic models available at the time of writing (e.g. [7, 15, 17, 33, 34, 37]) are not mentioned.

The treatment of learning is an area of special importance for brain theory. The examples, particularly the computational models, generally are based on "supervised learning", in which a "teacher" with knowledge of the correct responses to a given set of stimuli is available. The unrealism of this form of learning in a biological context is acknowledged. To correct this problem, the supervisor is replaced with an internal "monitor" that provides similar feedback information. The authors then point out that there is really no essential difference between supervised and monitored learning and go on to postulate some mechanisms by which biologically realistic monitored learning could occur. These begin with mechanisms based on detecting high-frequency features in the environment (p. 102) but don't get much farther. The notion that a monitoring signal is not needed for pattern completion (more generally, perceptual categorization), but is necessary to link behavior with its adaptive consequences (in their words, an "input-output mapping"), is made clear. In as much as the source of that monitoring signal is again suggested to be supervision, however, the "supervised"/"unsupervised" distinction reappears.

I would like to suggest here that a better distinction to make would be whether the signal giving rise to modified behavior requires as its source an entity with a similar

or greater level of cognitive ability than that of the system being taught ("homuncular learning") or does not ("nonhomuncular learning"). Homuncular learning requires the existence of one mind to train another, and therefore is inherently circular. It can generate copies of an existing cognitive system, provided that system is computationally based so it operates on well-defined rules, but cannot generate new cognitive systems. Nonhomuncular learning involves no third-party intervention—it is based on the ex post facto sensing of the adaptive value of behavior by a system interacting with an environment. In short, nonhomunculuar learning is biological learning. To avoid circularity, systems capable of nonhomuncular learning must require no learning to become functional in newborns and must be simple enough structurally to be encoded in the DNA inherited by an organism [27]. Accordingly, it is hard to see how such systems could be computational in nature, since no mechanisms are known by which DNA can encode either the symbolic representations of information or the rules for "bootstrap" computation that would be required for the system to start up.

4. Treatment of computational models

As already mentioned, the authors warn us in the preface that they will give us a California-oriented view of the state of neural modelling, and this is exactly what they do. That this geographical orientation happens mostly to coincide with the computational orientation criticized above is perhaps no accident. The selection of some of their own work (e.g. NETTALK, [30]) is certainly understandable, but the reasoning behind other decisions is obscure. For example, a chapter on plasticity that includes 68 pages on learning in the hippocampus culminates in the unrelated Miller, Keller, and Stryker model [22] for ocular dominance column development in visual cortex. Yet the intracortical connections in this model aren't even plastic! Detailed, compartmental models of neuron electrophysiology, which represent a major thrust of current research in neural systems, are mentioned only in a figure legend.

Most of the models presented are based on the back-propagation learning rule in spite of the authors' admission (p. 362) that this approach is not biologically realistic. A few quotations from their explanation encapsulate in a very clear way the central philosophical alignment of the book and allow us to see just how the computational approach encourages research that ignores the rich possibilities of learning more about the brain through study of its development as well as its mature functioning:

> The beauty of using a learning algorithm such as backpropagation rather than setting the weights by hand was that it allowed the network to discover an architecture that was efficient, accurate, and suited to neuronlike components. Comparisons could then be made between the properties of hidden units and properties of actual neurons to determine whether the model was approximating the real thing. (p. 359)

Comment: Whoever suggested setting weights by hand in a neural network model? This is a "straw-man" comparison. There are many training procedures available that avoid the unrealistic assumptions of back-propagation and yield performance that is as good or better (models using some of these have recently been reviewed by Reeke and Sporns

[26]). Is it not at least as important to understand how the nervous system acquires its abilities as it is to understand what properties the neurons have when the system is fully trained?

> But, of course, this is Nature, not Texas Instruments, and multitudinous advantages of vector-coding (distributed representing) accrue to an organization found by backpropagation—and apparently also by evolution. (p. 359)

Comment: Actually, back-propagation *is* Texas Instruments—an automated method of carrying out gradient descent optimization that is useful in engineering applications but, as far as we know, is not found in Nature. Once again, the sole organizing principle from Nature that is considered important in this book is evolution.

> Notice that "backprop" is used not because anyone thinks a real nervous system gets wired up in that way. Of course, it does not. Rather, backprop is used to get the network into a weight space where it functions accurately and with a behavioral profile that is comparable to its real counterpart .... Backprop enables the network to find an error minimum, and it is not unreasonable to assume that evolution, over many trials and with countless errors, stumbled into much the same error minimum. Having a trained-up network in hand, the researcher can then study it, and more particularly, can study properties of the network that would be awesomely difficult or downright impossible to study in a real network. By [sic] why use backprop rather than just an exhaustive search of all possible weight settings? In a network with 100 variable weights, and a supercomputer to try out all the possibilities, it would take approximately 10^{30} centuries to examine all possible combinations of weights. Backprop is a lot faster. (pp. 362–363)

Comment: The straw man of sequential search is again put down. Evolution is again the only mechanism mentioned for the development of organization in *real* neural systems, although the authors surely recognize that individual experience is also important. Other mechanisms are permitted in *model* neural systems. Among these, back-propagation is preferred. Back-propagation is considered to be so marvelous that the possibility of learning something by trying to model how neural systems actually become organized is not considered. The attempt of Miller, Keller, and Stryker to do just this, presented with approbation a little earlier, is here forgotten. In my opinion, computationalism is directly responsible for this sorry oversight, because it encourages its followers to describe brain function from the point of view of an outside observer, in functional terms, accepting the semantics of neural activity as given, rather than to seek the developmental and experiential basis by which neural activity acquires meaning.

5. The writing

This book will either delight or drive the reader to distraction with its wildly enthusiastic, informal language. For example, the difference between local and distributed representations is said to fall "rather short of spinetingling, barn-burning stuff". Long-term potentiation comes in various flavors named for ice creams, such as vanilla and

strawberry ripple. After LTP has been described, "it is time to come up for air". There is a plentiful stock of "withals" and "anons" for lovers of anachronism. Transitions tend to be sudden and not always well prepared, perhaps vestiges of boundaries where one author left off and the other began. Nonetheless, it is worth noting, because it is rare today, that the English grammar is almost always careful and correct; the editing is generally superb, with very few infinitives split or participles left dangling. Only a few errors marred the text: the confusion of a capacitance with a time constant on p. 123, that of a chemical (NMDA) with a mechanism on p. 263, that of efflux of sodium ions with influx on p. 400. Figure 6.40(A) is described in a legend but is nowhere to be found.

6. Summary

In summary, *The Computational Brain* is by no means a comprehensive textbook of neuroscience, but rather a good general introduction followed by more detailed tours through a few areas of special interest to the authors. The book gives the impression that the computational approach to the brain consists in exploring a variety of individual problems involving different brain regions that employ different architectures, different data encodings, and different learning algorithms to accomplish their different information processing tasks. This diversity is probably inevitable given the present immature state of the field, and the reader seeking a common theme to organize the disparate data of neuroscience will find little more help here than in other recent works on computational models. Churchland and Sejnowski do quote Marr [21], who suggested breaking the analysis of mental tasks down into computational, algorithmic, and implementational levels, but there is little effort to apply these principles to the modelling studies. The unity that could come from basing the models on a common conception of the physiology of the neuron and on experimentally observed mechanisms of neural plasticity is lacking. What appears to be important, in the view of the authors, is to obtain a model that reproduces some critical aspect of the behavior of the nervous system, and then to compare the properties of the model neurons with those of the real ones; how the real system comes to behave as it does is assigned little importance. This explication of the field is not even fair to computational modelers who have tried to consider issues of development (e.g. [20,22,36], etc.) and is certainly not representative of the view among large segments of the neuroscience community, who consider that models might be of some use for relating behavior to neuronal function, if only more of the *data* could be included. Churchland and Sejnowski present one or two realistic models [11,35], but do little to bridge the gulf between these and the simplified larger-scale models that they emphasize; a service would have been performed for the community if the authors had done more to show how realistic modelling can be extended to the behavioral level. As it is, the book will be of most interest to those who would like to become familiar with the approaches and results of one prominent group of researchers in this field. For those seeking a presentation less colored by the computationalist philosophy of mind, one of the several excellent compendia of original research papers that are available [1,2] would form a better introduction to the subject.

References

[1] J.A. Anderson, A. Pellionisz and E. Rosenfeld, eds., *Neurocomputing 2. Directions for Research* (MIT Press, Cambridge, MA, 1990).

[2] J.A. Anderson and E. Rosenfeld, eds., *Neurocomputing. Foundations of Research* (MIT Press, Cambridge, MA, 1988).

[3] W. Bechtel, *Philosophy of Mind: An Overview for Cognitive Science* (Erlbaum, Hillsdale, NJ, 1988).

[4] M.H. Bickhard, Cognitive representation in the brain, *Encycl. Human Biol.* **2** (1991) 547–558.

[5] D.C. Dennett, *Consciousness Explained* (Little Brown, Boston, 1991).

[6] R. Eckhorn et al., Coherent oscillations: a mechanism of feature linking in the visual cortex? Multiple electrode and correlation analyses in the cat, *Biol. Cybern.* **60** (1988) 121–130.

[7] R. Eckhorn et al., A neural network for feature linking via synchronous activity: results from cat visual cortex and from simulations, in: R.M.J. Cotterill, ed., *Models of Brain Function* (Cambridge University Press, Cambridge, 1989).

[8] G.M. Edelman, *Neural Darwinism: The Theory of Neuronal Group Selection* (Basic Books, New York, 1987).

[9] G.M. Edelman, *Bright Air, Brilliant Fire. On the Matter of the Mind* (Basic Books, New York, 1992).

[10] C.M. Gray and W. Singer, Stimulus-specific neuronal oscillations in orientation columns of cat visual cortex, *Proc. Nat. Acad. Sci. USA* **86** (1989) 1698–1702.

[11] S. Grillner, J.T. Buchanan and A. Lansner, Simulation of the segmental burst generating network for locomotion in lamprey, *Neurosci. Lett.* **89** (1988) 31–35.

[12] S. Grossberg, Competitive learning: from interactive activation to adaptive response, *Cogn. Sci.* **11** (1987) 23–63.

[13] J.W. Gyr et al., Computer simulation and psychological theories of perception, *Psychol. Bull.* **65** (1966) 174–192.

[14] M. Johnson, *The Body in the Mind. The Bodily Basis of Meaning, Imagination, and Reason* (University of Chicago Press, Chicago, IL, 1987).

[15] D.M. Kammen, P.J. Holmes and C. Koch, Cortical architecture and oscillations in neuronal networks: feedback versus local coupling, in: R.M.J. Cotterill, ed., *Models of Brain Function* (Cambridge University Press, Cambridge, 1989).

[16] P.K. Kienker et al., Separating figure from ground with a parallel network, *Perception* **15** (1986) 197–216.

[17] P. König and T.B. Schillen, Segregation of oscillatory responses by conflicting stimuli—desynchronizing connections in neural oscillator layers, in: R. Eckmiller, G. Hartmann and G. Hauske, eds., *Parallel Processing in Neural Systems and Computers* (Elsevier, Amsterdam, 1990).

[18] G. Lakoff, *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind* (University of Chicago Press, Chicago, IL, 1987).

[19] S.A. Levin et al., *National Science Foundation Workshop: Mathematics and Biology: The Interface. Challenges and Opportunities* (Lawrence Berkeley Laboratory, University of California, Berkeley, CA, 1992).

[20] R. Linsker, From basic network principles to neural architecture: emergence of orientation-selective cells, *Proc. Nat. Acad. Sci. USA* **83** (1986) 8390–8394.

[21] D. Marr, *Vision* (Freeman, San Francisco, CA, 1982).

[22] K.D. Miller, J.B. Keller and M.P. Stryker, Ocular dominance column development: analysis and simulation, *Science* **245** (1989) 605–615.

[23] M. Minsky, *The Society of Mind* (Simon & Schuster, New York, 1985).

[24] G.N. Reeke Jr, Book Review of *The Society of Mind* (Marvin Minsky), *Artif. Intell.* **48** (1991) 341–348.

[25] G.N. Reeke Jr and G.M. Edelman, Real brains and artificial intelligence, Daedalus, *Proc. Am. Acad. Arts Sci.* **117** (1988) 143–173.

[26] G.N. Reeke Jr and O. Sporns, Behaviorally based modeling and computational approaches to neuroscience, *Ann. Rev. Neurosci.* **16** (1993) 597–623.

[27] G.N. Reeke Jr et al., Synthetic neural modeling: a multilevel approach to the analysis of brain complexity, in: G.M. Edelman, W.E. Gall and W.M. Cowan, eds., *Signal and Sense: Local and Global Order in Perceptual Maps* (Wiley, New York, 1990).

[28] J.R. Searle, *The Rediscovery of the Mind* (MIT Press, Cambridge, MA, 1992).
[29] T.J. Sejnowski, C. Koch and P.S. Churchland, Computational neuroscience, *Science* **241** (1988) 1299–1306.
[30] T.J. Sejnowski and C.R. Rosenberg, Parallel networks that learn to pronounce English text, *Complex Syst.* **1** (1987) 145–168.
[31] G.M. Shepherd, The significance of real neuron architectures for neural network simulations, in: E.L. Schwartz, ed., *Computational Neuroscience* (MIT Press, Cambridge, MA, 1990).
[32] P. Smolensky, On the proper treatment of connectionism, *Behav. Brain Sci.* **11** (1988) 1–74.
[33] H. Sompolinsky, D. Golomb and D. Kleinfeld, Global processing of visual stimuli in a neural network of coupled oscillators, *Proc. Nat. Acad. Sci. USA* **87** (1990) 7200–7204.
[34] O. Sporns, G. Tononi and G.M. Edelman, Modeling perceptual grouping and figure-ground segregation by means of active reentrant conditioning, *Proc. Nat. Acad. Sci. USA* **88** (1991) 129–133.
[35] R.D. Traub and R. Miles, *Neural Networks of the Hippocampus* (Cambridge University Press, Cambridge, 1991).
[36] C. von der Malsburg and J.D. Cowan, Outline of a theory for the ontogenesis of iso-orientation domains in visual cortex, *Biol. Cybern.* **45** (1982) 49–56.
[37] D. Wang, J. Buhmann and C. von der Malsburg, Pattern segmentation in associative memory, *Neural Comp.* **2** (1990) 94–106.
[38] T. Winograd and F. Flores, *Understanding Computers and Cognition. A New Foundation for Design* (Addison-Wesley, Reading, MA, 1986).

ELSEVIER

Artificial Intelligence 82 (1996) 393–394

Artificial Intelligence

Forthcoming Papers

A. Becker and D. Geiger, Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem

We show how to find a small *loop cutset* in a Bayesian network. Finding such a loop cutset is the first step in the method of conditioning for inference. Our algorithm for finding a loop cutset, called MGA, finds a loop cutset which is guaranteed in the worst case to contain less than twice the number of variables contained in a minimum loop cutset. The algorithm is based on a reduction to the weighted vertex feedback set problem and a new approximation of the latter problem. The complexity of MGA is $O(m + n \log n)$ where m and n are the number of edges and vertices respectively. A greedy algorithm, called GA, for the weighted vertex feedback is also analyzed and bounds on its performance are given. We test MGA on randomly generated graphs and find that the average ratio between the number of instances associated with the algorithm's output and the number of instances associated with an optimum solution is 1.22 for the graphs tested.

P. Walley, Measures of uncertainty in expert systems

This paper compares four measures that have been advocated as models for uncertainty in expert systems. The measures are additive probabilities (used in the Bayesian theory), coherent lower (or upper) previsions, belief functions (used in the Dempster–Shafer theory) and possibility measures (fuzzy logic). Special emphasis is given to the theory of coherent lower previsions, in which upper and lower probabilities, expectations and conditional probabilities are constructed from initial assessments through a technique of natural extension. Mathematically, all the measures can be regarded as types of coherent lower or upper previsions, and this perspective gives some insight into the properties of belief functions and possibility measures. The measures are evaluated according to six criteria: clarity of interpretation; ability to model partial information and imprecise assessments, especially judgements expressed in natural language; rules for combining and updating uncertainty, and their justification; consistency of models and inferences; feasibility of assessment; and feasibility of computations. Each of the four measures seems to be useful in special kinds of problems, but only lower and upper previsions appear to be sufficiently general to model the most common types of uncertainty.

Y. Moses and M. Tennenholtz, Off-line reasoning for on-line efficiency: knowledge bases

The complexity of reasoning is a fundamental issue in AI. In many cases, the fact that an intelligent system needs to perform reasoning on-line contributes to the difficulty of this reasoning. This paper considers the case in which an intelligent system computes whether a query is entailed by the system's knowledge base. It investigates how an initial phase of off-line preprocessing and design can improve the on-line complexity

Elsevier Science B.V.

considerably. The notion of an *efficient basis* for a query language is presented, and it is shown that off-line preprocessing can be very effective for query languages that have an efficient basis. The usefulness of this notion is illustrated by showing that a fairly expressive language has an efficient basis. A dual notion of an *efficient disjunctive basis* for a knowledge base is introduced, and it is shown that off-line preprocessing is worthwhile for knowledge bases that have an efficient disjunctive basis.

A.C.C. Say and S. Kuru, Qualitative system identification: deriving structure from behavior

Qualitative reasoning programs (which perform simulation, comparative analysis, data interpretation, etc.) either take the model of the physical system to be considered as input, or compose it using a library of model fragments and input information about how to combine them. System identification is the task of creating models of systems, using data about their behaviors. We present the qualitative system identification algorithm QSI, which takes as input a set of qualitative behaviors of a physical system, and produces as output a constraint model of the system. QSI's output is guaranteed to produce its input when simulated. Furthermore, the QSI-made models usually contain meaningful "deep" parameters of system which do not appear in the input behaviors. Various aspects of QSI and its applicability to diagnosis, as well as the model fragment formulation problem, are discussed.

J. Lin, Integration of weighted knowledge bases

P.P. Nayak and L. Joskowicz, Efficient compositional modeling for generating causal explanations

S. Das and N. Ahuja, Active surface estimation: integrating coarse-to-fine image acquisition and estimation from multiple cues

G. Zlotkin and J.S. Rosenschein, Compromise in negotiation: exploiting worth functions over states

S. Kraus, An overview of incentive contracting

H. Almuallim, An efficient algorithm for optimal pruning of decision trees

Artificial Intelligence 82 (1996) 395–396

Author Index—Volume 82 (1996)

(The issue number is given in front of the page numbers)

Borgida, A.
On the relative expressiveness of description logics and predicate logics (Research Note) 353–367
Chakrabarti, P.P., see Dasgupta, P. 237–257
Dasgupta, P., P.P. Chakrabarti and S.C. DeSarkar
Searching game trees under a partial order 237–257
Dechter, A., see Dechter, R. 1– 20
Dechter, R. and A. Dechter
Structure-driven algorithms for truth maintenance 1– 20
DeSarkar, S.C., see Dasgupta, P. 237–257
Geiger, D. and D. Heckerman
Knowledge representation and inference in similarity networks and Bayesian multinets 45– 74
Gomatam, J., see Reynolds, D. 303–330
Greiner, R. and P. Orponen
Probably approximately optimal satisficing strategies 21– 44
Hansson, S.O.
Knowledge-level analysis of belief base operations 215–235
Hansson, S.O.
A test battery for rational database updating (Research Note) 341–352
Heckerman, D., see Geiger, D. 45– 74
Hickey, R.J.
Noise modelling and evaluating learning from examples 157–179
Huang, Z., M. Masuch and L. Pólos
ALX, an action logic for agents with bounded rationality 75–127
Kambhampati, S. and D.S. Nau
On the nature and role of modal truth criteria in planning 129–155
Levesque, H.J., see Selman, B. 259–272
Masuch, M., see Huang, Z. 75–127
Nau, D.S., see Kambhampati, S. 129–155

Elsevier Science B.V.

Nilsson, N.J.
Book Review of *Artificial Intelligence: A Modern Approach* (Stuart Russell and Peter Norvig) 369–380
Orponen, P., see Greiner, P. 21– 44
Pólos, L., see Huang, Z. 75–127
Reeke Jr, G.N.
Book Review of *The Computational Brain* (Patricia S. Churchland and Terrence J. Sejnowski) 381–391
Reynolds, D. and J. Gomatam
Stochastic modelling of Genetic Algorithms 303–330
Roth, D.
On the hardness of approximate reasoning 273–302
Russell, S., see Zilberstein, S. 181–213
Selman, B. and H.J. Levesque
Support set selection for abductive and default reasoning 259–272
Valdés-Pérez, R.E.
A new theorem in particle physics enabled by machine discovery (Research Note) 331–339
Zilberstein, S. and S. Russell
Optimal composition of real-time systems 181–213

Artificial Intelligence

Editor-in-Chief
Daniel G. BOBROW
Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304, USA

Associate Editor
Michael BRADY
Department of Engineering Science
University of Oxford
Parks Road
Oxford OX1 3PJ, UK

Research Notes Editor
Wolfgang BIBEL
Informatik/Intellektik
Technische Hochschule
Alexanderstr. 10
D-64283 Darmstadt, Germany

Review Editor
Mark J. STEFIK
Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304, USA

Associate Review Editor
Stephen W. SMOLIAR
FX Palo Alto Laboratory, Inc.
3400 Hillview Avenue, Bldg. 4
Palo Alto, CA 94304, USA

Editorial Board

L.C. AIELLO
Università di Roma "La Sapienza"
Rome, Italy

S. AMAREL
Rutgers University
New Brunswick, NJ, USA

Y. ANZAI
Keio University
Hiyoshi, Yokohama, Japan

R.K. BAJCSY
University of Pennsylvania
Pittsburgh, PA, USA

H.G. BARROW
University of Sussex
Brighton, England, UK

H. BERLINER
Carnegie-Mellon University
Pittsburgh, PA, USA

R.S.B. BOYER
University of Texas
Austin, TX, USA

R. BRACHMAN
AT&T Bell Laboratories
Murray Hill, NJ, USA

B.G. BUCHANAN
University of Pittsburgh
Pittsburgh, PA, USA

A. BUNDY
University of Edinburgh
Edinburgh, Scotland, UK

J.G. CARBONELL
Carnegie-Mellon University
Pittsburgh, PA, USA

W.J. CLANCEY
Institute for Research on Learning
Palo Alto, CA, USA

R. DAVIS
AI Lab, MIT
Cambridge, MA, USA

R. DECHTER
University of California
Irvine, CA, USA

J. DE KLEER
Xerox PARC
Palo Alto, CA, USA

L. ERMAN
Cimflex-Teknowledge
Palto Alto, CA, USA

K.D. FORBUS
Northwestern University
Evanston, IL, USA

M.R. GENESERETH
Stanford University
Stanford, CA, USA

M.L. GINSBERG
University of Oregon
Eugene, OR, USA

B. GROSZ
Harvard University
Cambridge, MA, USA

P.J. HAYES
Beckman Institute
Urbana, IL, USA

G. HINTON
University of Toronto
Toronto, Ont., Canada

K. KONOLIGE
SRI International
Menlo Park, CA, USA

W.G. LEHNERT
University of Massachusetts
Amherst, MA, USA

H.J. LEVESQUE
University of Toronto
Toronto, Ont., Canada

V. LIFSCHITZ
University of Texas at Austin
Austin, TX, USA

A.K. MACKWORTH
University of British Columbia
Vancouver, BC, Canada

J. McCARTHY
Stanford University
Stanford, CA, USA

D. McDERMOTT
Yale University
New Haven, CT, USA

T.M. MITCHELL
Carnegie-Mellon University
Pittsburgh, PA, USA

F. MIZOGUCHI
Science University of Tokyo
Tokyo, Japan

R. MOORE
SRI International
Menlo Park, CA, USA

H.-H. NAGEL
Fraunhofer-Institut für Informations- und Datenverarbeitung
Karlsruhe, Germany

J. PEARL
University of California
Los Angeles, CA, USA

C.R. PERRAULT
SRI International
Menlo Park, CA, USA

M. POLLACK
University of Pittsburgh
Pittsburgh, PA, USA

R. QUINLAN
University of Sydney
Sydney, Australia

R. REITER
University of Toronto
Toronto, Ont., Canada

E. SANDEWALL
University of Linköping
Linköping, Sweden

Y. SHOHAM
Stanford University
Stanford, CA, USA

H. TANAKA
Tokyo Institute of Technology
Tokyo, Japan

W. WAHLSTER
Universität des Saarlandes
Saarbrücken, Germany

Y. WILKS
Sheffield University
Sheffield, England, UK

W.A. WOODS
Sun Microsystems Laboratories, Inc.
Chelmsford, MA, USA