ELSEVIER

Artificial Intelligence 105 (1998) 209–261

Artificial Intelligence

# Multiple perspective dynamic decision making

Tze Yun Leong [1]

*Medical Computing Laboratory, Department of Computer Science, School of Computing, National University of Singapore, Lower Kent Ridge Road, Singapore 119260, Singapore*

Received 15 September 1997; received in revised form 17 April 1998

**Abstract**

Decision making often involves deliberations in different perspectives. Distinct perspectives or views support knowledge acquisition and representation suitable for different types or stages of inference in the same discourse. This work presents a general paradigm for multiple perspective decision making over time and under uncertainty. Based on a unifying task definition and a common vocabulary for the relevant decision problems, this new paradigm balances the trade-off between model transparency and solution efficiency in current decision frameworks.

The new paradigm motivates the design of DynaMoL (Dynamic decision Modeling Language), a general language for modeling and solving dynamic decision problems. The DynaMoL framework differentiates inferential and representational support for the modeling task from the solution or computation task. The dynamic decision grammar defines an extensible decision ontology and supports complex problem specification with multiple interfaces. The graphical presentation convention governs parameter visualization in multiple perspectives. The mathematical representation as semi-Markov decision process facilitates formal model analysis and admits multiple solution methods. A set of general translation techniques is devised to manage the different perspectives and representations of the decision parameters and constraints. DynaMoL has been evaluated on a prototype implementation, via some comprehensive case studies in medicine. The results demonstrate practical promise of the framework. 

*Keywords:* Decision making; Knowledge representation; Multiple perspective reasoning; Probabilistic reasoning; Temporal reasoning; Semi-Markov decision processes

## 1. Introduction

Dynamic decision making concerns problems in which time and uncertainty are explicitly considered. For example, a common medical decision is to choose an optimal

[1] Email: leongty@comp.nus.edu.sg.

0004-3702/98/$ – see front matter 
PII: S0004-3702(98)00082-4

course of treatment for a patient whose physical conditions may vary over time; a common financial investment decision is to determine an optimal portfolio with respect to fluctuating market factors over time.

Reasoning about dynamic decision problems often involves integrating information from different perspectives or viewpoints. For instance, at one stage of problem deliberation, it might be essential to consider the possible physiological states that a patient would go through; at another stage, it would be illuminating to estimate the uncertain effects of a treatment that would lead to different physiological states.

Research in control theory, operations research, decision analysis, artificial intelligence (AI) and other disciplines has led to various techniques for formulating, solving, and analyzing dynamic decision problems. They adopt different assumptions, support different ontologies, and have different strengths and weaknesses.

One major difficulty of dynamic decision making is to fit the complex decision factors into simple, parameterized models. Many assumptions and constraints are implicit in the resulting models. The limited decision vocabulary contributes toward solution efficiency, but obscures model transparency. Moreover, all the current frameworks support single perspective reasoning; the decision models conform to the fixed vocabulary or graphical presentation convention of the specific framework. In decision analysis, for instance, influence diagrams [37] and decision trees [60] provide alternate perspectives to the same decision problem. The respective strengths of the different frameworks are best suited for different stages of decision formulation, solution, and analysis. In AI, various efforts have addressed issues in reasoning at multiple levels of abstraction. None of these frameworks, however, integrates and supports different representational views.

### *1.1. Research objectives and approaches*

This work introduces a new paradigm of dynamic decision making. We present a uniform way to reason about a general class of dynamic decision problems and a common vocabulary for addressing such problems. We describe a methodology that supports multiple perspective reasoning and incremental language extension. Multiple perspective reasoning allows the modeler to visualize and examine the same information in different ways; it facilitates effective modeling and analysis of dynamic decision problems. Incremental language extension provides a framework that can be customized through the use of *translators*; it allows the scope of the dynamic decision problems addressed to be gradually expanded.

The proposed methodology motivates a language design, called DynaMoL (Dynamic decision Modeling Language). To balance the trade-off between model transparency and solution efficiency, the DynaMoL design differentiates representational and inferential requirements for the modeling task from those for the solution or computation task. The relevant support for the modeling task facilitates specification and derivation of the correspondences and relationships among the decision parameters and constraints; the relevant support for the solution task accelerates computation of the optimal course of action in a well-formed model.

We evaluate DynaMoL via a prototype implementation with some comprehensive case studies in medicine.We demonstrate that DynaMoL is expressive enough to handle a class

of real-life dynamic decision problems. We claim that the proposed methodology is more general and more effective than most existing techniques. The exercise also illuminates how the proposed methodology motivates a new generation of dynamic decision making tools and techniques and how it can be put into practical use.

### *1.2. Guide to the paper*

This paper is organized as follows: Section 2 describes the class of dynamic decision problems addressed in this work. Section 3 introduces the main concepts in multiple perspective reasoning and the desiderata of a multiple perspective dynamic decision modeling language. Section 4 presents the design of the DynaMoL framework. Section 5 sketches the domain background and the decision problems for a case study. Based on the case study, Section 6 describes decision model formulation in DynaMoL; it examines in detail the syntax and the semantics of the language. Section 7 discusses how the different perspectives in DynaMoL are translated among each other and how consistency is maintained in the translation process. Sections 8 and 9 briefly examine the solution methods and the analyses supported by the language. Section 10 illustrates the design of a prototype system for DynaMoL and documents some of our experiences in applying the system to different dynamic decision problems in some practical domains. Section 11 compares the methodology with related work and proposes some ideas for future research. It also examines the implications of the methodology in general AI and decision making research. Finally, Section 12 summarizes the achievements and limitations of this work.

## 2. Dynamic decision making under uncertainty: an overview

### *2.1. The dynamic decision problem*

The general dynamic decision problem is to select a course of *action* that satisfies some *objectives* in an *environment*. The main distinguishing feature of a dynamic decision problem from a static one is the explicit reference to time. The environment description, the action or decision points, and the objective measures are specified with respect to a time horizon. Fig. 1 depicts the factors involved in a dynamic decision problem.

This work addresses dynamic decision problems with the following properties:

- The time horizon is defined as a set of discrete time points.
- The environment comprises a finite set of discrete, or reasonably assumed to be discrete, phenomena. A patient is either "well" or "sick" on the third day after being treated; a can that a robot is about to grasp is "full", "half-full", or "empty".
- There is a finite set of discrete actions. These actions are context-dependent; they have varying preconditions, usually with respect to the environment or time or both. A patient can only go through three open heart surgeries, and each surgery can only be carried out if the patient's physical conditions permit.
- Each action has a finite set of discrete, or reasonably assumed to be discrete, effects. After a treatment, a patient who was previously "sick" is "well" again; moving forward from its current position, a robot "gets closer to the target position". The

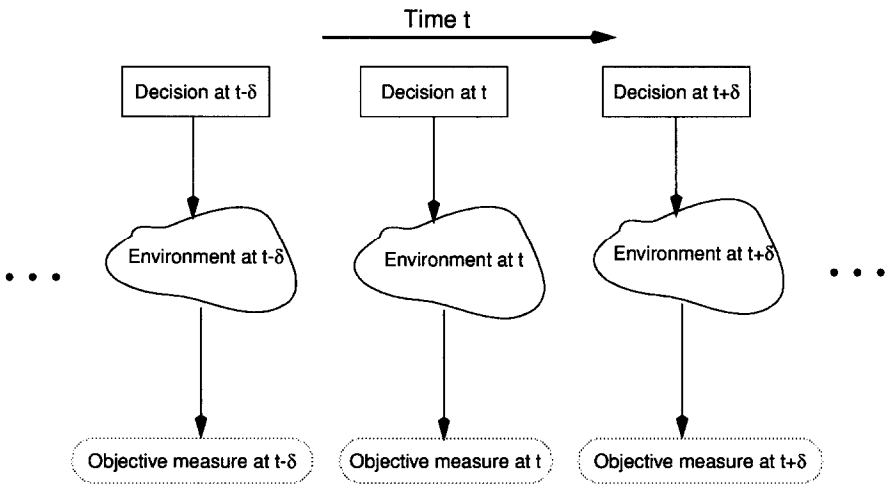


Fig. 1. A dynamic decision problem.

nature of the effects are often uncertain. A treatment either cures a disease or leads to some undesirable side-effects. Moreover, the time at which the effects may occur are also uncertain. A patient who is cured of peptic ulcer may have a relapse sooner than another patient.

- The effects of an action have measurable desirability. Such a measure can be multi-dimensional, e.g., the desirability of staying in a hospital and being well, versus staying at home and being sick, but it must be *time separable*, i.e., the total desirability can be calculated by summing the desirability functions over time.

Given an initial environment, the problem is solved by choosing a course of action that optimizes the expected desirability of their potential effects. The decisions are made in stages; the stages may vary in duration.

### 2.2. *Current approaches*

The major approaches to addressing the class of dynamic decision problems described above include the following:

#### 2.2.1. *Markov and semi-Markov decision processes*

Markov decision processes (MDPs) are mathematical models of sequential optimization problems with stochastic formulation and state structure. In the 1950s, research in MDPs began with the ideas of Shapley [64] and Bellman [3], and the formalization by Howard [34] and others. Jewell [39] extended these results to semi-Markov decision processes (SMDPs), which are more general than MDPs. Much progress has occurred since, both in extending the basic mathematical definitions and in improving the optimization algorithms [59]. The small and simple vocabulary involved in the methodology, however, necessitates many assumptions and constraints to be implicitly incorporated, and hence renders it quite formidable to directly formulate complex dynamic decision problems in practice.

In medicine, for instance, while Markov and semi-Markov processes are often used in survival and prognosis analyses, MDPs and SMDPs are seldom applied directly in clinical decision making [38].

### 2.2.2. *Dynamic decision analysis*

Emerging in the 1960s from operations research and game theory [60], decision analysis is a useful decision making technique under risk and uncertainty. Based on probability theory and utility theory, the normative, analytical framework not only helps the decision maker determine an optimal course of action, but also gain insights into the complex decision situation by systematically constructing and analyzing a graphical decision model [36]. In recent years, some new decision analytic formalisms have been devised to deal with dynamic decision problems. These include *dynamic influence diagrams* [70], *Markov cycle trees* [2,33] and *stochastic trees* [32]; they are based on structural and semantical extensions of conventional decision models such as influence diagrams and decision trees, with the mathematical definitions of stochastic processes.

Dynamic decision analysis is used widely in real world applications. For instance, it is a common tool in clinical decision making [40,54,56,57]. Nevertheless, the methodology is difficult to apply. In particular, model formulation is knowledge-intensive and labor-intensive, and the graphical structures restrict the admissible solution methods [44].

### 2.2.3. *Planning in artificial intelligence*

Motivated by the studies of human problem solving [52], operations research, and logical theorem proving, AI planning emerged in the 1960s with the works of Newell et al. [51] and Fikes and Nilsson [26]. Early research in this area focuses on representing and reasoning with complete and deterministic information. Recent progress introduces imperfect information, extends the planning ontology, and improves the plan-generation and plan-execution algorithms [69]. Analytical studies on the theoretical foundations of the various planning frameworks, in terms of their expressiveness and complexity, are also gaining attention [12,25,47].

The AI planning works that have modest impact on realistic applications, however, are special-purpose algorithms [48]. While there are some general-purpose planning systems in use, e.g., SIPE [75] and O-Plan [13], they are difficult to be applied in practice [20]. The impracticality of AI planners is partly due to the complexity of the problems addressed, and partly due to the difficulty to encode domain-dependent planning problems and heuristics in the fixed planning vocabularies. As compared to, say, an MDP, where the solution algorithms operate directly on the fully specified set of actions, states, and the probabilistic relationships among them, an AI planner allows more flexible and expressive action and problem descriptions. In hierarchical task network (HTN) planning, for instance, partial plans with hierarchical representation of the actions, their effects, and the relevant constraints are represented with various syntactic abbreviations [20]. Such expressiveness facilitates formulation of highly complex problems, but may significantly complicate search control for the optimal solutions.

#### 2.2.4. Recent development in decision-theoretic planning

In the late 1980s, research efforts in decision-theoretic planning (DTP) began to integrate techniques from the different approaches described above. The general DTP problem involves planning in a stochastic domain, where the action effects are uncertain. Most works in DTP are based on the semantic and computational framework of MDPs [16]. The major focus in DTP research is, however, on speeding up computation of the optimal solutions with approximation techniques. Relevant approaches include using probabilities to evaluate potential plans in achieving goals [23,43]; restricting search to local regions of the state space and allowing existing MDPs solution techniques to be applied on the reduced problems [1,19,67]; and using abstraction or aggregation of the state space [7,17, 22] or the action space or both [31] to reduce the search space for optimal plans.

DTP frameworks usually employ more compact and structured representations of the problem components than MDPs. For instance, a common representation of uncertain action effects adopted in such frameworks is the probabilistic STRIPS-like operators [43]. More general or expressive definitions of the state space [30] and the utility functions or goal descriptions [6,30,74] may also be involved. The expressive representations adopted in these frameworks mainly aim to support search control for the optimal solutions; they sometimes also facilitate formulation of the planning problems, but this is a usually a secondary concern.

### 2.3. A uniform task definition

The three major tasks in dynamic decision making under uncertainty are problem formulation, solution, and analysis. These tasks may be iterated many times for a given problem.

#### 2.3.1. Problem formulation

Formulating a dynamic decision problem begins with specifying the problem type, the decision parameters, and the constraints involved. In *discriminatory* problems, different actions or strategies are compared in terms of their effectiveness. In *optimization* problems, an optimal course of action is determined. Examples of decision parameters include alternative actions and strategies, possible state transitions, uncertain events that constitute the state transitions, and relevant probability distributions with respect to time. Examples of decision constraints include applicability conditions of the actions, validity conditions of the states, and logical relationships among the states and uncertain events.

The formulation task is guided by the decision ontology or vocabulary with the following defining characteristics:

- Expressiveness of a decision ontology defines how accurately we can specify the parameters and constraints in a decision situation; it determines the types and organization of the relevant factors and their interrelationships. For example, a decision ontology may include only actions and states as basic concepts and temporal dependences as basic relations, while another may also include uncertain events and probabilistic dependences. Similarly, a decision ontology may represent actions as individual concepts, while another may support reasoning with classes of actions.

- Succinctness of a decision ontology determines how easily we can specify the parameters and constraints in a decision situation. For example, in SMDPs, all the states in a dynamic decision problem have to be explicitly enumerated; in AI planning, the state descriptions can be factored into the relevant conditions or attributes.
- Transparency of a decision ontology reflects how efficiently we can draw inferences on the information involved in a decision situation, and how easily we can comprehend such information. An ontology with a simple syntax or semantics does not necessarily imply that it is transparent; unless it is succinct, many expressions may be needed to specify a small piece of knowledge.

#### 2.3.2. *Problem solution*

The solution to a well-formed dynamic decision problem is a course of action that optimizes some objectives. A *closed-loop* solution is conditional on the observation of the decision situation; it solves the problem with respect to all possible initial points and all possible end points, e.g., a policy determined in an SMDP. An *open-loop* solution does not employ feedback from the decision situation; it solves the problem with respect to a specific initial point and a specific end point, e.g., a plan generated by a classical AI planner for a starting state and a goal state. A solution method is a computational model that manipulates the information in the well-formed model; it does not need to know how the information organization arises in the first place.

#### 2.3.3. *Problem analysis*

Both the formulation and the solution of a dynamic decision problem may require sensitivity analysis of the decision parameters and constraints involved to ensure accuracy, relevance, and clarity. Sensitivity analysis can be divided into structural sensitivity and numerical sensitivity. Structural sensitivity is reflected through adjustments of the qualitative information involved; it ensures that the problem is framed correctly, and the relevant alternatives, objectives, and uncertain events are specified properly. Numerical sensitivity is reflected through adjustments of the quantitative information involved; it includes threshold analysis and stochastic analysis that reveal how the uncertain events affect the choice and the value of the decision.

### 2.4. *The application domain*

While the issues we address in this work are general, the application domain we examine is diagnostic test and therapy planning in medicine. The relevant dynamic decision problems involve the risk of some adverse events that may continue or vary over time; the events may recur and the timing of such recurrences are uncertain. The relevant actions are diagnostic tests and treatments, or combinations of them. The environment comprises the physical conditions of a patient or a class of patients. These conditions include the physiological states of the patient, or any observable or unobservable events that would lead to the states. For instance, a patient can be in a state with a stroke, and a cerebral hemorrhage may have caused the stroke. Some of these events are the effects of the actions. The decision objectives usually include both prolonging the patient's life expectancy and maximizing the cost-effectiveness of the actions.

## 3. Multiple perspective reasoning

Decision making often involves deliberations in different perspectives. Distinct perspectives or views support knowledge acquisition and representation suitable for different types or stages of inference in the same discourse. Many decision making frameworks provide support for visualizing the decision parameters and constraints in specific graphical perspectives. Different frameworks, however, may display the same information in different ways.

### *3.1. Advantages and disadvantages of different perspectives*

To illustrate the strengths and weaknesses of visualizing in different graphical perspectives, consider the following example:

**Example.** A patient is in one of three states: WELL, SICK or DEAD. A doctor must decide between treatments A and B, each of which will have a different effect on the patient in either the WELL or the SICK state. The patient can only stick with one course of treatment, which may be repeated many times. Treatment A has a higher initial success rate in treating the disease, but the patient will die almost immediately once the treatment fails, i.e., does not improve the patient's condition within the fixed period before the next treatment decision. Treatment B, on the other hand, may cause a serious complication C in an already sick patient.

Fig. 2 illustrates the structural representations of the problem in (a) a Markov state transition diagram of an MDP; (b) a dynamic influence diagram over a finite time period; and (c) a Markov cycle tree. Each representation projects the problem in a specific perspective; each perspective explicates the problem structure at a different level of detail, from a different point of view. Problem solution and analysis in each framework are conducted with respect to the chosen perspective.

#### *3.1.1. State transition diagram*

In a state transition diagram, the nodes denote the states, and the arcs, with their associated transition functions, denote the possible transitions given the action. A value function is defined for each state in the diagram.

As shown in Fig. 2(a), each state transition diagram explicitly shows the possible state transitions given an action. The diagram does not reflect, however, the uncertain events, e.g., complication C for treatment B, that may occur during state transitions.

#### *3.1.2. Dynamic influence diagram*

In a dynamic influence diagram, the squares represent the decision nodes, the circles the chance nodes, and the diamonds the value nodes. The number at the end of each node indicates the decision stage in which the decision/event/value is considered. The arcs leading into the chance and value nodes indicate probabilistic dependences, and the arcs leading into the decision nodes indicate informational dependences. Embedded in each chance node or value node is a list of the possible values or outcomes of the node, e.g.,

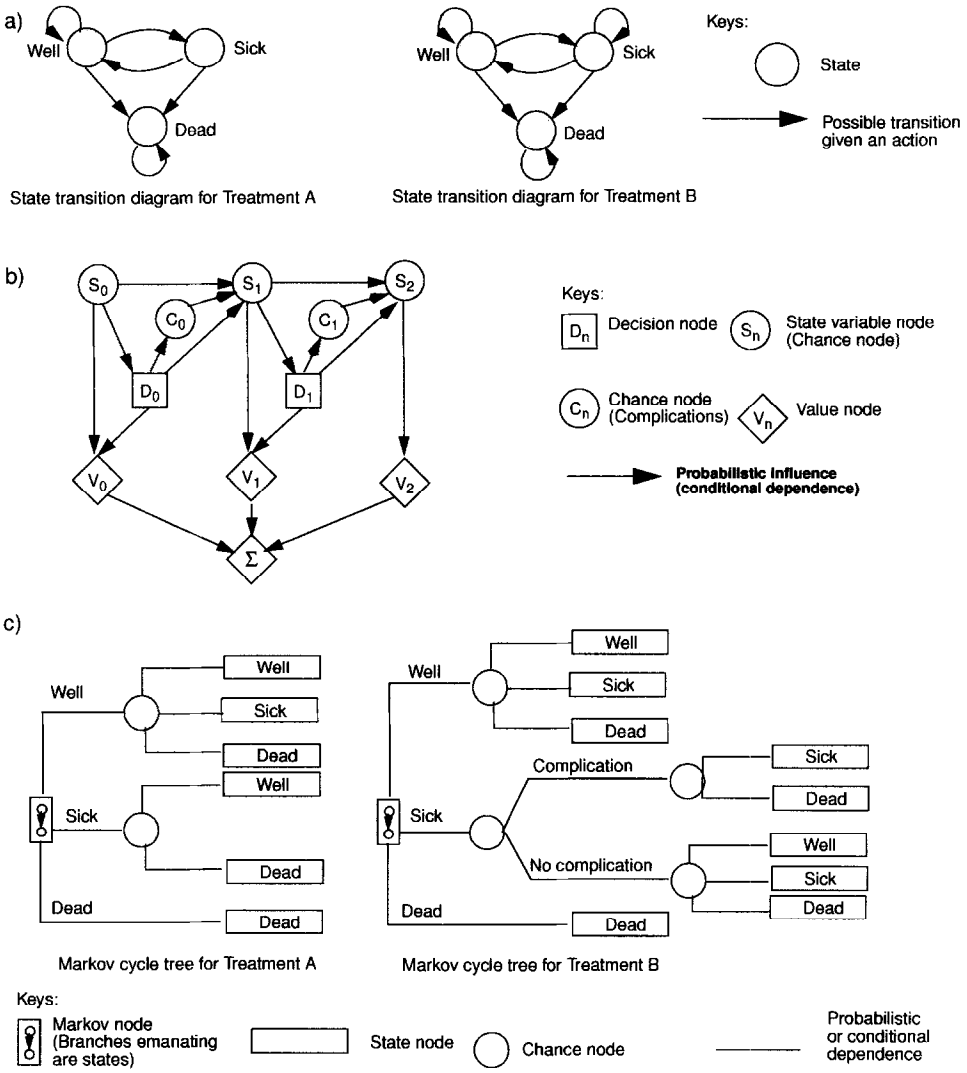


Fig. 2. Representation of a simple decision problem in: (a) Markov state transition diagrams; (b) a dynamic influence diagram; and (c) Markov cycle trees.

"Well", "Sick" and "Dead" for $S_n$ and "Complication C" and "No complication C" for $C_n$, and a table of probabilities conditional on its probabilistic predecessors; embedded in each decision node is a list of the alternate treatments and a list of its informational predecessors.

As shown in Fig. 2(b), a dynamic influence diagram explicitly shows the sequential structure of the decision problem, and may capture the varying uncertain events in-between state transitions. But the possible patterns of state transitions in particular, and the relations among the chance outcomes and/or the decision alternatives in general, are kept only in

the table entries embedded in the nodes. For instance, the structure of the diagram does not reflect that only treatment B may lead to complication C.

#### *3.1.3. Markov cycle tree*

In a Markov cycle tree, the branches emanating from the root node, called the Markov node, and the leaf nodes, respectively, represent the states at the beginning and at the end of a decision stage, given an action. The intermediate nodes are chance nodes and the arcs indicate the possible outcomes of as well as the probabilistic dependencies among the nodes. All the possible combinations of uncertain events that could happen between any two state transitions are represented in between the root and the leaves of a cycle tree. A conditional probability is associated with each branch of the tree. A value function is defined for each state in the diagram.

As shown in Fig. 2(c), a Markov cycle tree explicitly shows the possible state transitions given an action, and reflects the uncertain events in-between state transitions; it also depicts asymmetric relations among the chance outcomes and decision alternatives. For instance, complication C may occur only in an already sick patient if treatment B is prescribed. The tree, however, becomes extremely complex if the state transitions and the uncertain events vary with time; it also cannot represent a mixture of sequential actions effectively.

### *3.2. Desiderata of an integrated decision language*

Based on the above observations, we postulate that an effective language for dynamic decision making should provide representational and inferential support for the different tasks involved, while balancing the trade-off between model transparency and solution efficiency. In particular, such a language should satisfy the following desiderata:

- The decision ontology must address a wide range of issues in typical dynamic decision problems. The language components and their organization should be expressive, succinct, and transparent.
- The language should have a formal theoretical basis with simple and rigorous syntax and semantics. These qualities would facilitate examining and analyzing the formal properties of the problems and their solutions.
- The language must support reasoning at multiple levels of abstraction, so that a user can deal mainly with the relevant ontological concepts, instead of the specific mathematical definitions.
- The language must support graphical visualization of the decision parameters and constraints in multiple perspectives and across different levels of abstraction. Visualization in multiple perspectives reflects a common pattern in human decision making. It is analogous to viewing the world with different pairs of lenses, each providing a perspective most natural to a particular task.
- The language should be incrementally extensible and adaptable. Extensions involve incorporating additional language constructs for new types of decision parameters and constraints. Adaptation involves changes in the organization of the language constructs; it does not necessarily affect the expressiveness of the language.
- The language should support implementations that can be put into practical use. A practical language is modular, comprehensive and explicit.

## 4. DynaMoL: the language

The DynaMoL design is motivated by an in-depth analysis of existing frameworks [44] and based on the desiderata outlined in the previous section. In this section, we explain the design approach and summarize the language features. Detailed definitions of the design will be presented in Sections 6–9.

### *4.1. Design approach*

To address the transparency-efficiency trade-off, the DynaMoL design aims to separate the representation and inference support for modeling and solving dynamic decision problems. It incorporates the general idea of programming language design: translating a high-level language for modeling into an object language for execution.

The language comprises four components: the *dynamic decision grammar* defines an extensible decision ontology and supports complex problem specification with multiple interfaces; the *graphical presentation convention* governs parameter visualization in multiple perspectives; the *mathematical representation* as SMDP facilitates formal model analysis and admits multiple solution methods; a set of general *translation* techniques is devised to manage the different perspectives and representations of the decision parameters and constraints.

The syntax of the language is defined by the dynamic decision grammar and the graphical presentation convention. The semantics is defined by the mathematical representation of an SMDP and the translations that bridges the grammar, the presentation, and the mathematical representation.

In DynaMoL, therefore, we formulate and examine a dynamic decision model in terms of the high-level decision grammar with different graphical visualization perspectives; this high level model is then translated into a formal mathematical representation for solution and analysis.

### *4.2. Basic decision ontology*

We adopt a knowledge formalization approach similar to first-order predicate calculus (FOPC), as presented in [27]. In DynaMoL, a *conceptualization* of a decision situation includes the relevant decision parameters and the interrelationships among them. Based on the conceptualization, we define the expressions in DynaMoL in terms of the following components: *basic concepts*, *variable concepts*, *functions*, *relations* and *constraints*.

#### *4.2.1. Basic concepts*

A *basic concept* is a partial description of the decision situation at a single point in time. There are three types of basic concepts: events, actions and states.

- An *event* describes a property of a phenomenon. It corresponds to a proposition of the form: $P(\tau)$ with unary predicate $P$ and constant $\tau$. In the clinical context, "presence of complication C", "absence of complication C", "number of stroke (that a patient has suffered) is 1", and "number of stroke (that a patient has suffered) is 2" are all examples of events, which can be expressed as the propositions:

```
Complication-C(Present),
Complication-C(Absent),
Number-of-stroke(1),
Number-of-stroke(2),
```

respectively.

- An *action* is an externally controlled phenomenon. It corresponds to a single or a conjunction of events which involves an actor. In the clinical context, relevant actions include different tests and treatments, and their combinations, e.g., "Test A", "Treatment B", "Test X followed by Treatment Y." The effects of an action are expressed in terms of events, e.g., a side-effect of prescribing Treatment B is the event "presence of complication C". These actions and their effects can be expressed as the propositions and conjunctive propositional sentences:

```
Test(A),
Treatment(B),
Test(X)&Treatment(Y),
Complication-C(Present),
```

respectively.

- A *state* describes a phenomenon with properties relevant to the overall decision objectives, brought about by performing certain actions. It corresponds to a conjunction of special events called the *state attributes*. For instance, the state WELL in a typical clinical decision problem can be expressed as the conjunctive propositional sentence:

```
Vital-status(Alive)&Disease-X(Absent)&
  Disease-Y(Absent).
```

Every state has an associated *utility* or *value*, indicating the desirability of being in that state.

#### 4.2.2. *Variable concepts*

A *variable concept* represents uncertainty in a partial description of the decision situation at a single point in time. It corresponds to a propositional function of the form: $P(x)$ with unary predicate $P$ and variable $x$ in FOPC or a random variable in probability theory. There are two types of variable concepts: chance variables and action variables. We only consider discrete variables in this work.

(a) An instantiation of a *chance variable* in the FOPC interpretation and an outcome of it in the probabilistic interpretation is an event or a state as defined earlier. Each possible instantiation or outcome occurs only by chance. A chance variable also corresponds to a chance node in a decision model. There are three types of chance variables:

(i) *Event variable*. The possible instantiations or outcomes of an event variable are events. For instance, "number of stroke (that a patient has suffered)" is

an event variable that can be denoted as an atomic sentence: `Number-of-stroke(x)`, with possible instantiations `Number-of-stroke(1)` and `Number-of-stroke(2)`, or a random variable: Number-of-stroke $= x$, with possible outcomes $x = 1$ or 2.

(ii) A *state attribute variable* is a special kind of event variables; it represents a characteristic or property directly relevant to describing the states. The possible instantiations or outcomes of a state attribute variable are the state attributes, which are special events. Given a set of such variables, a state is usually defined in terms of the Cartesian product of their outcomes. Some heuristics, however, may be used to reduce the number of the resulting states. For instance, the state of a patient, described as WELL, SICK, or DEAD, can be defined in terms of the state attribute variables: "vital status (alive or dead)", "disease X (present or absent)", and "disease Y (present or absent)". In particular, as mentioned, the state WELL can be expressed as the conjunction of the state attributes: `Vital-status(Alive)&Disease-X(Absent)&Disease-Y(Absent)`.

(iii) A *state variable* represents the uncertainty about the actual state that is reachable at a single point in time. It corresponds to a *state space* in an SMDP and a state variable node in a dynamic influence diagram at that time point. All the outcomes of a state variable are states. For example, the possible instantiations or outcomes of the state variable for a clinical decision problem are the states that a patient can be in: WELL, SICK and DEAD.

(b) An *action variable* denotes a decision or choice point at a single point in time. An instantiation of an action variable in the FOPC interpretation and an alternative of it in the probabilistic interpretation is an action as defined earlier. An action variable also corresponds to part or all of the action space in an SMDP or a decision node in decision model. For example, in a clinical decision problem, the possible alternatives for an action variable can be denoted as an atomic sentence: `Treatment(x)`, with possible instantiations `Treatment(A)` and `Treatment(B)`, or a random variable: Treatment $= x$, with possible outcomes $x =$ A or B.

*4.2.3. Functions*

There are two types of basic *functions* defined over the basic and variable concepts:

(a) A *probability function* is either a probability mass function (PMF) or a cumulative distribution function (CDF). The probability functions express the conditional probabilities among the events and/or the transition characteristics among the states, given an action.

(b) A *value function* measures the desirability of being a state in the decision situation. It may have different dimensions, e.g., monetary cost and life expectancy in the clinical context, and is usually conditional on an action.

*4.2.4. Relations*

There are two types of *basic* relations defined among the basic and variable concepts, which are to be distinguished from the domain relation constants used in the descriptions of the basic concepts in Section 4.2.1:

(a) A *probabilistic dependence* relation between two concepts corresponds to the conditional dependence notion in probability theory. Such a relation indicates that a concept is conditionally dependent on another, given the decision situation. The direction of a probabilistic dependence relation reflects the definition of the underlying conditional dependence; it has no temporal implication. For example, "presence of complication C" is probabilistically dependent on the action "Treatment B" and the state "(the patient is) SICK".

(b) A *temporal dependence* relation between two concepts indicates that one temporally precedes another; it has no causal implication. For example, the state variable at time $t$ is temporally dependent on the state variable at time $t-1$.

#### 4.2.5. Constraints

A *constraint* is a meta-level descriptive or prescriptive condition imposed on the concepts, the relations, and the functions as defined above. It correspond to a logical or quantification sentence in FOPC. For instance, the constraint that all the actions $a \in \boldsymbol{A}$ are assumed to be applicable in all the states $s \in \boldsymbol{S}$ at all time points $t \in \boldsymbol{T}$ can be expressed as a quantification sentence:

$$\forall a \forall s \forall t \big(\text{Applicable}(a, s, t)\big).$$

### 4.3. Dynamic decision grammar

The dynamic decision grammar for DynaMoL is an *abstract grammar*. The grammar defines the structure of a dynamic decision model in terms of its components; the structures of these components are recursively defined in a similar manner. The abstract grammar can support multiple interface implementations. Moreover, the grammar can be extended by defining appropriate translators for the new constructs.

The complete dynamic decision grammar for DynaMoL is documented in [45]; it contains the following components:

- A finite set of names of *constructs*;
- A finite set of *productions*, each associated with a construct.

An example of a production that defines the construct "model" is as follows:

*Model* → *name*: *Identifier*;
*contexts*: *Context-list*;
*definitions*: *Definition-list*;
*constraints*: *Constraint-list*;
*solution*: *Optimality-policy*.

Each construct describes the structure of a set of objects, called the specimens of the construct. The construct is the (syntactic) *type* of its specimens. The constructs/types appearing on the right-hand side of the above definition are similarly defined by different productions. The structure of a construct can be specified in terms of "aggregate" productions, i.e., the constructs have specimens comprising a fixed number of components, "choice" productions, "list" productions, etc.

### 4.4. Graphical presentation

The presentation convention in DynaMoL prescribes how the decision parameters and constraints in the grammar are displayed; it determines how the same information can be visualized in multiple perspectives. The current convention includes three established graphical representations.

#### 4.4.1. Transition view

The transition view for a given action depicts the possible state transition patterns. It corresponds to the Markov state transition diagram in the MDP framework. As shown in Fig. 3, each node represents a state or condition with an associated utility or value function. Conditional on the action, each arc represents a possible transition in the state of origin; a one-step transition function is defined on each arc to govern the transition characteristics.

#### 4.4.2. Influence view

An influence view for a given action shows the nature of uncertainties involved in the state transitions and allows the state descriptions to be factored into the relevant state attribute variables. It corresponds to a Bayesian or probabilistic network; it also corresponds to a slice of a dynamic influence diagram for a specific decision stage, conditional only on one alternative.

As shown in Fig. 4, each node denotes an event variable with a few possible outcomes, e.g., the outcomes for **Disease-X?** are "X" (disease X present) and "No-X" (disease X absent). The state space at each decision stage is represented as a state variable. The possible outcomes of the state variable are all the states in the corresponding state space, e.g., the outcomes of the state variable node $S_n$ (state at decision stage $n$) and the next-state variable node $S_{n+1}$ (state at decision stage $n+1$) are the four states depicted in Fig. 3. In this example, the event variables **Disease-X?** and **Disease-Y?** are also the state attribute variables that contribute to the following state definitions: WELL = {No-X, No-Y}, DISCOMFORT = {No-X, Y}, SICK = {X} and DEAD. Each arc in the figure indicates conditional or probabilistic dependence. A conditional probability distribution table is associated with each node. The diagram can be interpreted as follows: given that the action is taken at any of the state of origin at decision stage $n$, the state of the patient at decision stage $n+1$ is conditionally dependent on the presence or absence of disease X and the presence or absence of disease Y, as characterized by the underlying conditional probability distributions.

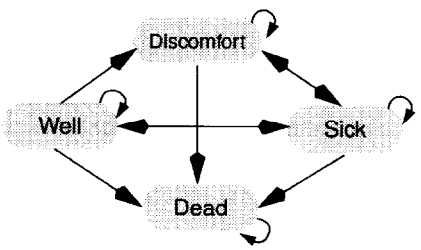


Fig. 3. A transition view for an action.

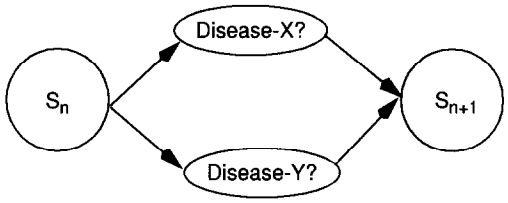


Fig. 4. An influence view for an action.

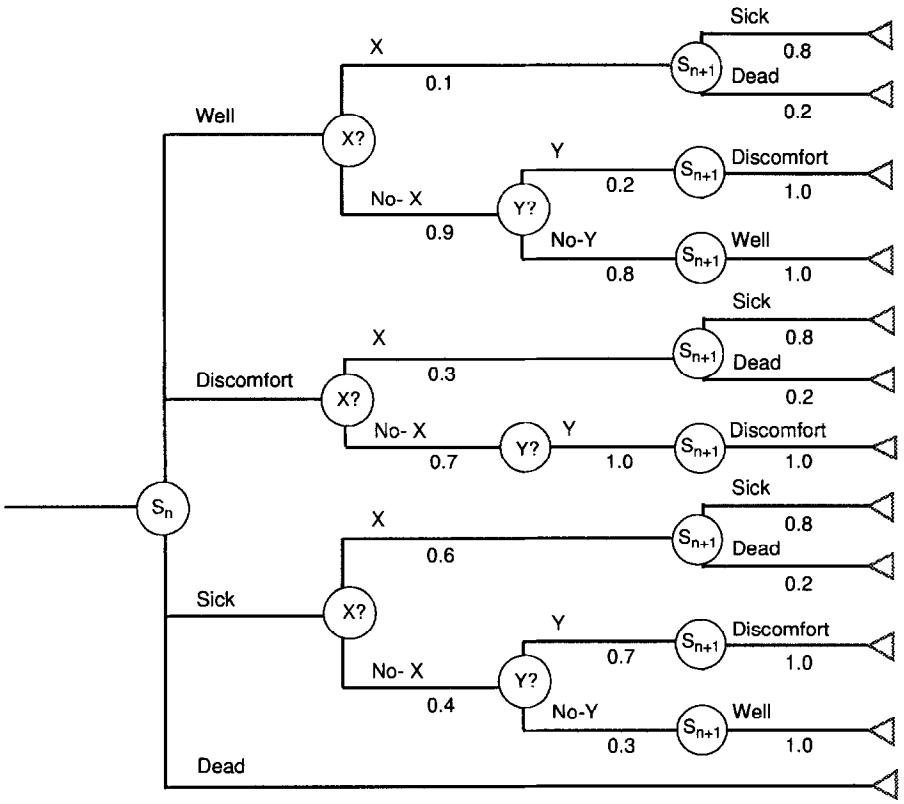


Fig. 5. A tree view for an action.

### 4.4.3. *Tree view*

The tree view for a given action is a decision tree expansion of the influence view. It corresponds to a Markov cycle tree for a specific action. It explicitly shows the uncertain events in-between possible state transitions in a chronological manner; it also depicts asymmetric relations among the chance outcomes.

As shown in Fig. 5, the nodes correspond to the event variables in the influence views, and the arcs capture both the possible outcomes of the event variables and probabilistic dependences. In particular, the root node $S_n$ and the leaf nodes $S_{n+1}$ represent the state

variables at decision stages $n$ and $n+1$, respectively, and the branches of these nodes denote the possible states at the corresponding decision stages. A probability entry is associated with each outcome branch of an event variable, conditional on the given action and all the states and/or events to the left of the event variable.

### 4.5. Mathematical representation

The central idea in supporting multiple perspective reasoning is to establish a common basis among the different views or aspects of a decision situation. In DynaMoL, the theoretical basis of a dynamic decision model is an SMDP.

Briefly, an SMDP is a mathematical model of a sequential decision process. An SMDP has the following basic components: a *time index set* $\boldsymbol{T}$, an *action space* $\boldsymbol{A}$, a *state space* $\boldsymbol{S}$, a set of *one-step transition functions* with probability mass functions (PMFs) $q_{ij}^{(a)}(m,t)$ and cumulative distribution functions (CDFs) $Q_{ij}^{(a)}(m,t)$, and a set of *value functions* $v_i^{(a)}(m)$, where $i,j \in \boldsymbol{S}$, $a \in \boldsymbol{A}$, $t \in \boldsymbol{T}$, $m = \{0,1,2,3,\ldots\}$. The stochastic nature of the transitions are reflected in both the *destination* (i.e., state $j$) of the transition and the *time lapsed* (i.e., duration $m$) before the transition, both governed by the one-step transition functions. An MDP is an SMDP with constant inter-transition times at one time unit (i.e., $m=1$).

A solution of an SMDP is an *optimal policy*; it is a guideline for choosing the optimal actions over the decision horizon, for all possible evolutions of the states, that maximize the expected value or reward. Many solution methods are available for SMDPs.

### 4.6. Translation convention

Two types of translation are involved in co-ordinating multiple perspective reasoning in DynaMoL: *inter-level translation* and *inter-perspective translation*.

In inter-level translation, a model specified in the dynamic decision grammar is translated into an SMDP. The model may involve more constructs than those defining an SMDP. A set of translators or correspondence rules are employed to establish the proper correspondence. The general idea is to map a construct or a set of constructs in the grammar to an entity or relation in the mathematical representation.

Inter-perspective translation, on the other hand, establishes correspondence among the different representational formats. Since the same information is involved, inter-perspective translation can be defined in terms of any covering set of representation constructs among the different perspectives. For example, the state variable in the influence view corresponds to the root node with all its branches in the tree view, which also corresponds to the state space of the transition view.

### 4.7. Dynamic decision making in DynaMoL

Dynamic decision making in DynaMoL is an iterative process; it involves interleaving and repeating the problem formulation, problem solution, and problem analysis tasks:

- The dynamic decision problem is formulated as a dynamic decision model. The model specifies the relevant decision parameters and constraints, including the uncertainties and the preferences involved.

- The dynamic decision model is solved with respect to an evaluation or optimal criterion, to derive an optimal policy or course of action.
- The model and the solution are analyzed to ensure correctness of the formulation and robustness of the results.

In the next few sections, we discuss the modeling, solution, and analysis processes in the DynaMoL framework with a case study in medicine.

## 5. A case study

Atrial fibrillation (AF) is a kind of cardiac arrhythmia or abnormal heartbeat. It has two major undesirable side-effects: hemodynamic deterioration and embolization, i.e., formation of blood clots. AF can occur in paroxysms, i.e., sudden, periodic episodes. Both the frequency and the length of the AF episodes may increase with time; constant fibrillation often develops eventually.

AF management involves using antiarrhythmic agents to control heart rates and to restore normal sinus rhythm. Because of the risk of embolization, anticoagulants such as warfarin and aspirin are often used to prevent blood clot formation in the atria. The treatments, however, have corresponding undesirable side-effects. In particular, a common antiarrhythmic agent, Quinidine, increases the risk of sudden death; an anticoagulant may cause excessive bleeding, which in turn may lead to strokes or death.

The case in question is based on an actual clinical consult at the Tufts-New England Medical Center in Boston, Massachusetts. The patient is a 52-year old white male with a history of paroxysmal AF. He has been on warfarin and on Quinidine. The clinical decision problems are as follows:

**Question 1.** Quinidine decreases the proportion of time that the patient spends in AF. Does the decreased risk of embolic complications justify the increased risk of sudden death?

**Question 2.** What is the optimal course of treatment in the next five years, taking into account the varying relative risk of bleeding for warfarin with respect to the duration of treatment?

The case study aims to illuminate the theoretical and practical capabilities and limitations of the DynaMoL framework. The original clinical consult, which addresses only the first clinical question above, is modeled and solved in the Markov cycle tree framework. To show that DynaMoL can handle an actual dynamic decision problem in medicine, we adhere closely to the original clinical consult in addressing the first clinical question, reusing the exact qualitative and quantitative parameters of the model whenever appropriate. To show that DynaMoL offers additional flexibility to existing frameworks, we focus on illustrating the ontological and computational features involved in addressing the second clinical question; the clinical significance of the data and assumptions adopted in this latter problem is less important. For ease of exposition, we formulate the two dynamic decision problems in terms of a single dynamic decision model with two sets of slightly different numerical parameters.

We assess the effectiveness of DynaMoL for modeling and solving the dynamic decision problem with respect to three criteria. First, we demonstrate how the relevant decision

parameters and constraints can be expressed in the DynaMoL ontology. Second, we compare the solutions to the results of the actual clinical consult or the clinical judgment of domain experts or both. Third, we conduct sensitivity analysis on the solutions to ensure reasonable behavior of the parameters involved. The detailed discussion, data interpretation, and analysis for the case study are documented in [45].

## 6. Model formulation

Model formulation in DynaMoL typically comprises the following tasks:

(1) Specify a problem type, its duration, and its optimality and evaluation criteria;
(2) Define the alternative actions and the states involved;
(3) Conditional on each action, identify possible progression patterns of significant states or final outcomes and the values achievable in the states;
(4) When such transitions and values are difficult to be directly assessed, identify uncertain effects of actions by specifying the possible event variables and their relations that would constitute the transitions;
(5) Identify special assumptions on the actions, effects, and states and impose relevant constraints among the decision parameters when appropriate.

There is no strict order in performing the tasks involved. The specifications and definitions may and often change in the modeling process. A major challenge for the modeling language, therefore, is to provide adequate and direct support for the changes. In DynaMoL, we formulate a dynamic decision model via interfaces that implement the dynamic decision grammar, through multiple perspective visualization.

### 6.1. Definition of a dynamic decision model

A dynamic decision model in DynaMoL is a well-formed or complete model when it corresponds to a well-formed SMDP with an optimality criterion. Some model constructs directly correspond to the underlying mathematical definitions, others need to be translated. The default optimality criterion is finite-horizon discounted total expected value.

**Definition 1** (*Dynamic decision model*). A dynamic decision model $M$ in DynaMoL is an 10-tuple $\langle \boldsymbol{T}, \boldsymbol{A}, \boldsymbol{S}, \boldsymbol{E}, \Omega_{\boldsymbol{E}}, \Xi, \Psi, \zeta, K, \Gamma \rangle$ together with an *evaluation* or *optimality criterion*, where:

- The *decision horizon* or *time index set* $\boldsymbol{T} = \{0, 1, 2, 3, \ldots\}$ denotes the time frame for the decision problem.
- The *action space* $\boldsymbol{A} = \{a_1, a_2, \ldots, a_{|\boldsymbol{A}|}\}$ denotes the set of alternative actions to be considered.
- The *state space* $\boldsymbol{S} = \{s_1, s_2, \ldots, s_{|\boldsymbol{S}|}\}$ denotes the set of states or conditions that would affect the value functions and in which the actions can take place.
- For each action $a \in \boldsymbol{A}$, its *action-specific event variable space* $\boldsymbol{E}_a = \{e_1, e_2, \ldots, e_{|\boldsymbol{E}_a|}\}$ denotes the set of event or chance variables that constitute its effects. The *event variable space* $\boldsymbol{E} = \bigcup_{a \in \boldsymbol{A}} \boldsymbol{E}_a$ denotes the set of all event variables for all the actions.

- Each event variable $e$ has an associated *event* or *outcome space* $\Omega_e = \{\omega_1, \omega_2, \ldots, \omega_k\}$; $k \geqslant 1$, which denotes the set of possible outcomes for $e$. The *action-specific event space* $\Omega_{E_a}$ and the *overall event space* $\Omega_E$ are similarly defined.
- Conditional on each action, a set of possible *transitions* $\Xi = \{\xi(a, i, j, t) \mid i, j \in S, a \in A, t \in T\}$ is defined among the states.
  A transition relation $\xi \subseteq A \times S \times S \times T$ between any two states $i, j \in S$, given an action $a \in A$ at time $t \in T$, denotes the accessibility of $j$ from $i$ given $a$ at $t$. The nature of this accessibility is characterized by a *one-step transition function* with PMF $q_{ij}^{(a)}(.)$ and CDF $Q_{ij}^{(a)}(.)$ as defined for an SMDP.
- Conditional on each action, a set of *probabilistic influences* $\Psi = \{\psi(a, \boldsymbol{x}, \boldsymbol{y}, t) \mid a \in A, \boldsymbol{x}, \boldsymbol{y} \in E_a, t \in T\}$ is defined among the event variables.
  A probabilistic influence relation $\psi \subseteq A \times E_a \times E_a \times T$ between any two event variables $\boldsymbol{x}, \boldsymbol{y} \in E_a$, given an action $a \in A$ at time $t \in T$, reflects the probabilistic dependences, conditional on the action, among the outcomes of the two event variables at time $t$.
- Conditional on each action, and with respect to the evaluation criterion, a set of *value functions* $\zeta = \{v_i^{(a)}(m) \mid i \in S; a \in A, m = 0, 1, 2, \ldots\}$ is defined for the states.
  A value function $v : A \times S \times \aleph \to \mathbb{R}$ determines the objective value achievable in $i \in S$ state over time duration $m \in \aleph$, conditional on action $a \in A$.
- The above components can be subjected to a set of general or domain-specific constraints $K = \{\kappa \mid \kappa \subseteq \{S \cup \Omega_E \cup \Xi \cup \Psi \cup \zeta\}^n; n \geqslant 2\}$.
  A constraint $\kappa$ corresponds to a logical or quantification sentence or *well-formed formula* (wff) in FOPC.
- $\Gamma$ is the set of *translators*, i.e., the set of correspondence rules or functions that establish equivalence relations among the language constructs.

### 6.2. *Basic problem specification*

The basic characteristics of a dynamic decision problem are determined as follows:

#### 6.2.1. *Problem type*

Both optimization problems and discrimination problems are supported by DynaMoL. An optimization problem is solved by constructing an optimal policy

$$\pi^* = \{\mu_0^*, \mu_1^*, \mu_2^*, \mu_3^*, \ldots\}$$

where $\mu_t^* : S \to A$; $t \in T$ is an optimizing function from the state space $S$ to the action space $A$ at time $t$ with respect to the time index set $T$.

An optimization problem compares the independent effects of actions; the problem solution answers the question: at *each* decision stage, what is the best alternative action to take?

Contrarily, a discrimination problem is solved by choosing the best policy:

$$\pi^* = \pi^{a^*}$$

from a predetermined set of single-action stationary policies $\{\pi^a\}$, i.e., policies which involve a single action over the entire decision horizon, such that

$$\pi^{a^*} = \{\mu^{a^*}, \mu^{a^*}, \mu^{a^*}, \mu^{a^*}, \ldots\}$$

where $\mu^{a^*} : \boldsymbol{S} \rightarrow a^*$; $a^* \in \boldsymbol{A}$, is a function from the state space $\boldsymbol{S}$ to an optimizing action $a^*$.

The main type of discrimination problems delineates *strategies* or combinations of actions with dependent effects. This formulation designates a single action as externally *controlled* in a strategy, and incorporates the application pattern of some other relevant actions, called the *embedded* actions, in the state descriptions. To solve the problem, an explicit enumeration of the policies associating each controlled action and all possible combinations of the embedded actions is necessary. The problem solution answers the question: for the *entire* decision horizon, what is the best alternative strategy to take?

The first dynamic decision problem in the case study is formulated as a discrimination problem. Four strategies are to be discriminated:

(1) start without any drug, with warfarin administered and stopped as necessary;
(2) start with warfarin, which can be stopped when necessary;
(3) start with Quinidine, with warfarin administered and stopped as necessary; and
(4) start with both Quinidine and warfarin, with warfarin stopped when necessary.

The second dynamic decision problem in the case study is formulated as an optimization problem. The problem is to determine a course of optimal initial action, e.g., start with or without warfarin or Quinidine or both, with strategic implications as defined above.

Other dynamic decision modeling frameworks support formulation of optimization and discrimination problems to different extents. Dynamic influence diagrams support direct formulation of both optimization and discrimination problems. Without augmenting computational structures such as bindings or flags, and assumptions on the solution methods, Markov cycle trees only support direct formulation of discrimination problems. They do not support direct formulation of optimization problems because they do not allow decisions to be made in stages; a decision is made only at the final stage by comparing the values for the alternatives over the entire decision horizon.

### 6.2.2. *Decision horizon*

The decision horizon can be finite or infinite. A finite horizon with a long duration and small time units may be approximated as an infinite horizon. In the case study, the decision horizon $\boldsymbol{T}$ is 600 months or 50 years for the first dynamic decision problem, and 60 months or 5 years for the second.

### 6.2.3. *Evaluation criterion*

The evaluation criterion can be in any unit of interest, e.g., life expectancy, monetary cost, and cost-benefit ratio; it is defined by one or more sets of value functions. More than one evaluation criterion may be involved. The value functions are assumed to be linear and additive. The evaluation criterion for the case study is quality-adjusted life expectancy (QALE).

### 6.3. Structure specification

DynaMoL currently assumes a *static* action space and a *static* state space, i.e., they remain unchanged over the entire decision horizon. An action indicates a single unit of behavior; it may involve more than one activity, e.g., administer two drugs. The constituent activities in an action are assumed to be simultaneous and constant at every decision stage.

In the case study, the action space $A = \{\text{NoDrug, Quinidine}\}$ consists of the two externally controlled actions for the strategies involved. The effects of warfarin are modeled as embedded actions in the state descriptions. The state space $S$ includes the states defined in terms of the following state attribute variables: **vital status** indicates whether the patient is alive or dead; **sinus rhythm** indicates heartbeat pattern; **cerebral vascular accident** indicates history of obstruction of blood flow to the brain, which may result in a stroke or temporary weakness; **warfarin eligibility** affects the applicability of the action warfarin; **warfarin status** reflects the status of the embedded action warfarin.

Conditional on each controlled action, a set of value functions $\zeta$ is defined for the states. A value function $v_i^{(a)}(\cdot)$ indicates the QALE achievable in a state over a time duration, given the action.

In dynamic influence diagrams, the state attribute variables are represented as chance nodes directly influencing the value nodes; the implicit state descriptions, therefore, are the strict Cartesian product of the outcomes of these chance nodes. In Markov cycle trees, state attribute variables are captured in the bindings; state space definition in these frameworks often involves logical combinations of the relevant bindings.

#### 6.3.1. Modeling in the transition view

Based on the state attribute variables, a total of 20 states are defined in the case study. For instance, a relevant state is: AF-STR-WNE, which indicates that the patient is in atrial fibrillation (AF), has a history of stroke (STR), and is not eligible for warfarin (WNE).

Conditional on each controlled action, a set of possible transitions $\Xi$ is defined among the states. Each transition $\xi$ is associated with a one-step transition function with PMF $q_{ij}^{(a)}(\cdot)$ and CDF $Q_{ij}^{(a)}(\cdot)$, governing the destination of the transition and the time remaining before the transition.

Transition functions for an action can be specified directly on the links representing the transitions in a transition view. A simplified transition view for the action Quinidine, including only the state attributes related to the embedded action warfarin, is shown in Fig. 6.

The transition view above expresses, among others, the following facts for a patient on Quinidine:

- If a patient is not on warfarin, he may either be put on warfarin later or may be determined to be ineligible for warfarin.
- If a patient is on warfarin, he will never be put off warfarin unless he is later determined to be ineligible.
- Once a patient is determined to be ineligible for warfarin, he will never be considered for warfarin treatment again.

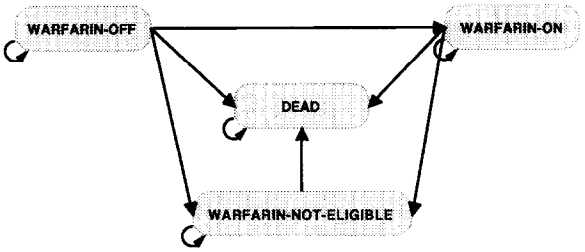


Fig. 6. Transition view for action Quinidine.

*6.3.2. Modeling in the influence view*

In the transition view, all the states are assumed to be fully observable. If the action effects are fully observable, they can be expressed in terms of the state attributes; the transition functions can then be specified directly. At times it is difficult to specify the transition functions directly, especially when the action effects are partially observable, unobservable, or interact in a complex manner. Such effects are modeled as event variables and the probabilistic influences among them.

In the case study, the event variable space $\boldsymbol{E}$ includes event variables indicating the presence or absence of **thromboembolization**, **cerebral hemorrhage**, **gastro-intestinal bleeding** and whether the patient survives the events should they occur. Conditional on an action, these events probabilistically influence the state attribute variables such as **warfarin eligibility**, **cerebral vascular accident**, etc., which in turn define the physiological states of the patient.

Conditional on each action, a set of *probabilistic influences* $\Psi$ is defined among the event variables; the absence of a probabilistic influence relation $\psi$ between any two event variables indicates conditional independence between them, given the decision context.

Given an action, conditional probabilities can be defined directly on the event variable nodes connected by probabilistic influences in the influence view. Fig. 7 shows an influence view for the action Quinidine in the case study. The outcomes of the two state variables, represented by the nodes named **state-n** and **state-n+1**, are the different states defined in the model. The state attribute variables are represented by the nodes directly influencing the state variable node **state-n+1**. While they need not be explicitly represented in the influence view, their incorporation facilities separate and independent specifications of the probabilistic dependencies among the event variables and the state attribute variables.

*6.3.3. Modeling in the tree view*

Operating on the same set of event variables and probabilistic influences in the influence view for an action, the tree view imposes a chronological order and explicit display the asymmetric relations. Conditional probabilities can be defined directly on the tree branches representing both the outcomes of the event variables and the probabilistic dependences. Fig. 8 shows a simplified version of a tree view for the action Quinidine in the case study, showing only the simplified states and one subtree for each branch. In the figure, all the branches with the terminating triangles are assumed to lead directly to the DEAD state in $S_{n+1}$. All the subtrees grouped by the brackets can be attached to all the branches in

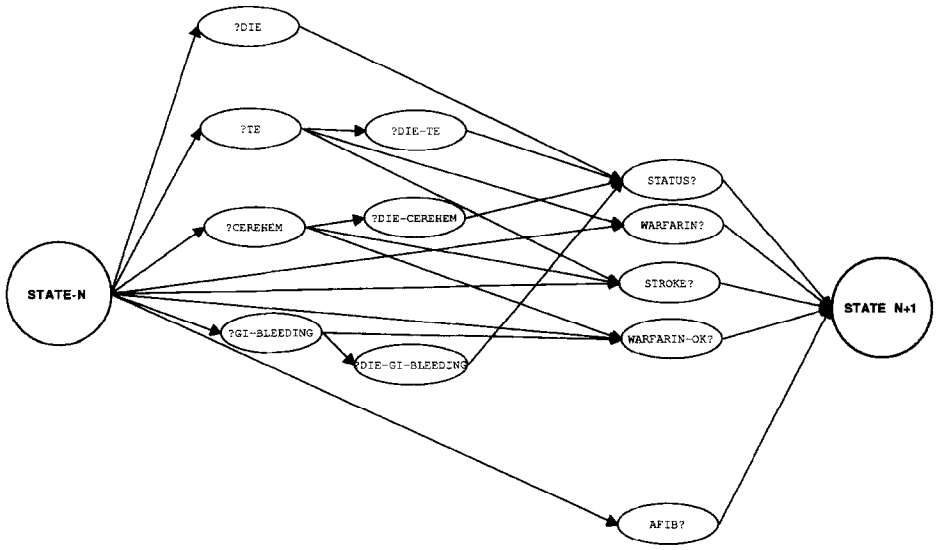


Fig. 7. Influence view for action Quinidine.

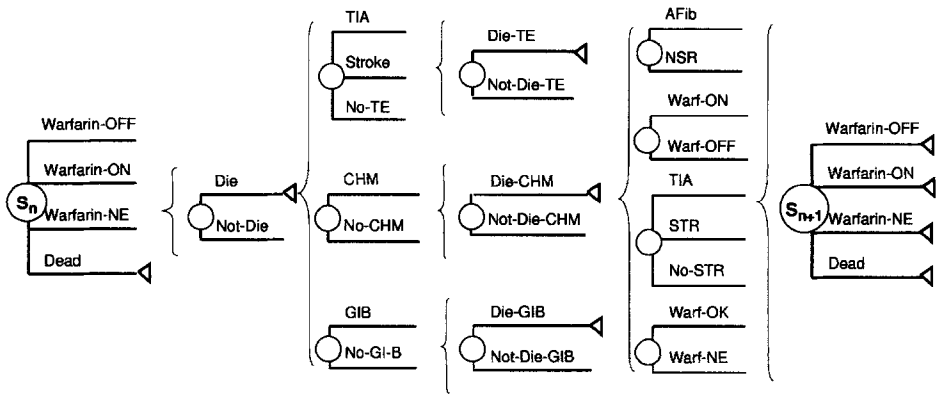


Fig. 8. Tree view for action Quinidine.

the directly preceding group of subtrees in any order. Asymmetries in the tree are reflected through the subtrees that can only be attached to selected groups of preceding branches, and some of the zero conditional probabilities along the branches, e.g., $P(\text{Die-TE} \mid \text{No-Te}) = 0$.

The strictly imposed ordering of the event variable nodes in the tree view may obscure the actual relations and render the expansion too complex to manipulate. DynaMoL also incorporates a *partial* tree view that supports partial expansion and focused specification of a particular event variable node; the expansion involves the chosen node and all its predecessor nodes in the influence view. In other words, a partial tree view facilitates filling in the conditional distribution table for the corresponding event variable node in the influence view.


Conditional distribution for ?Die-TE event variable:

```
(distribution ?die-te/NoDrug)
#<Distribution-Table #:G721
     Type:                DISCRETE
     Matrix: #:G721
        "P(c|r)"                 DIE-TE              NOT-DIE-TE
      PERM-STR                   PDIE-PERM-STR       (- 1 PDIE-PERM-STR)
      TRANS-STR                  PDIE-TRANS-STR      (- 1 PDIE-TRANS-STR)
      TIA                        0.00000             1.00000
      NO-TE                      0.00000             1.00000
```

Fig. 9. Asymmetric relations represented in an influence view.

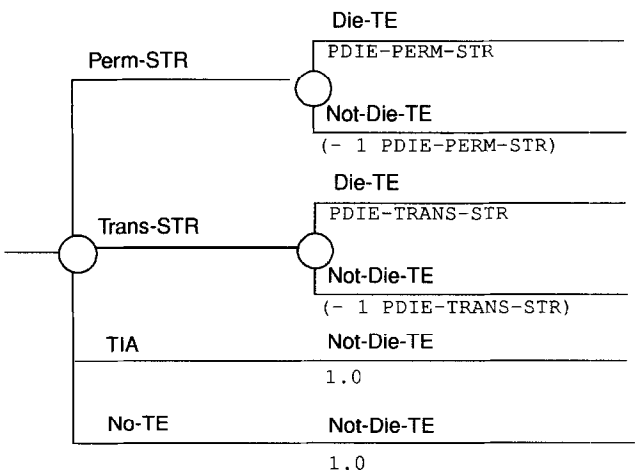


Fig. 10. A partial tree view for the event variable **?Die-TE** for the action NoDrug.

Consider the piece of knowledge: "a patient may have a chance of thromboembolization, which may be a stroke or a transient ischemic attack; if he has a stroke, it may be permanent or transient, and there is a chance that he will die from the stroke". Fig. 9 shows the partial influence view that represents the above knowledge. The asymmetry involved is embedded in the conditional distribution table.

The same piece of knowledge can be expressed in a partial tree view that expands the relevant event variables, as shown in Fig. 10. In the partial tree view, the asymmetry is obvious from the tree structure. The tree structure reflects the logical flow of represented knowledge more directly. Numerical or functional probabilities can be specified on the terminating branches. Only the probabilities on the terminating branches are relevant in the partial tree view.

### 6.4. Numerical parameters assessment

The numbers, value functions and probability distributions associated with the variables and relations are usually assessed after the structure of the model is in place.

#### 6.4.1. Value function definitions

The value function $v_i^{(a)}(m)$ is defined in terms of the associated transition value functions $v_{ij}^{(a)}(m)$ for each possible transition from state $i$ given action $a$. A transition value function $v_{ij}^{(a)}(m)$ determines the objective value achievable in a transition from state $i$ to state $j$ over time duration $m$, conditional on action $a$. It has two components: a *yield rate* $y_i^{(a)}(l)$ and a *bonus* $b_{ij}^{(a)}$. The yield rate indicates the value achievable at time $l$ after entering state $i$ over time interval $(l, l+1)$; the bonus is a one-time value achievable when the transition takes place from state $i$ [35]. The value functions are assumed to be independent of the absolute time $t$.

Formally

$$v_i^{(a)}(m) = \sum_j q_{ij}^{(a)}(m,t) v_{ij}^{(a)}(m) = \sum_j q_{ij}^{(a)}(m,t) \left( \left[ \sum_{l=0}^{m-1} y_i^{(a)}(l) \right] + b_{ij}^{(a)} \right);$$

$$a \in \boldsymbol{A},\ i, j \in \boldsymbol{S},\ l, m \geqslant 0.$$

In the case study, we adopt a special case for the value function: the yield rates and the bonus are constant for all states and transitions, and for all actions. In other words, $v_i^{(a)}(m) = v_{ij}(m) = my_i + b_{ij}$, for all $a \in \boldsymbol{A}$, $i, j \in \boldsymbol{S}$, $m \geqslant 0$. The yields are specified for the individual states, those for the absorbing states being zero. Bonuses are negative in this case, indicating short-term morbidities; they are associated only with transitions in which some adverse events may occur, e.g., the presence of stroke, cerebral hemorrhage, or gastro-intestinal bleeding.

In DynaMoL, the bonuses are currently specified with respect to the transitions, instead of their constituent event variables. Both dynamic influence diagrams and Markov cycle trees allow more detailed bonus specifications. In dynamic influence diagrams, the value functions usually contain only the yield rates for fixed time intervals; the bonuses can be incorporated by adding influence links from the relevant chance nodes to the value nodes. Similarly, the "toll" structures in Markov cycle trees allow bonuses to be associated with the chance nodes. A toll is a binding to an outcome of a chance node, indicating a positive or negative value associated with that outcome, e.g., $-0.25$ quality-adjusted life months for a stroke. Tolls can be added or subtracted from the total expected value accumulated as the branches of the tree structure are traversed. This feature, however, is feasible only with forward induction based solution algorithms.

#### 6.4.2. Transition functions and probabilistic influences

Some guidelines for directly specifying one-step transition functions in an SMDP, corresponding to the transition view in DynaMoL, are documented in [45]. When direct estimation of the one-step transition functions is difficult, the parameters can be specified in the influence views and/or the tree views, in terms of the conditional distributions of their constituent event variables instead.

For the Markov case, in the influence view, each entry $C[x, y]$ in a conditional distribution table or matrix $C$ associated with an event variable is:

$$P\{y \mid x, a, t\} \tag{1}$$

subject to

$$\sum_{y} P\{y \mid x, a, t\} = 1 \tag{2}$$

where $x$ is a row index representing a combination of outcome events of all the probabilistic predecessors for an event variable $\boldsymbol{y}$, $y \in \Omega_{\boldsymbol{y}}$ is a column index representing an outcome of $\boldsymbol{y}$, $a \in \boldsymbol{A}$, and $t \in \boldsymbol{T}$. The entry, which is usually a function $f(t)$ of time, indicates the probability of event $y$ in the next decision stage, given that event $x$ occurs and action $a$ taken at the decision stage at time $t$.

Similarly, in the tree view, (1) and (2) hold for the conditional probability associated with each branch or outcome $y \in \Omega_{\boldsymbol{y}}$ of an event variable $\boldsymbol{y}$, where $x$ is an index representing a combination of outcome events of all the probabilistic predecessors that lie to the left of, and leading to $y$ in the tree structure, given $a \in \boldsymbol{A}$ and $t \in \boldsymbol{T}$.

For the semi-Markov case, there are two ways to assess the one-step transition functions in terms of the constituent event variables. Consider specifying the PMFs of one-step transition functions, $q_{ij}^{(a)}(\dot{m}, t)$. The PMFs of one-step transition functions can be defined either in terms of the transition probabilities $P_{ij}^{(a)}(t)$ and holding time PMFs $h_{ij}^{(a)}(m, t)$, such that

$$q_{ij}^{(a)}(m, t) = P_{ij}^{(a)}(t) h_{ij}^{(a)}(m, t); \quad a \in \boldsymbol{A},\ i, j \in \boldsymbol{S},\ t \in \boldsymbol{T},\ m \geqslant 0 \tag{3}$$

or the conditional transition probabilities $p_{ij}^{(a)}(m, t)$ and waiting time PMFs $w_i^{(a)}(m, t)$, such that

$$q_{ij}^{(a)}(m, t) = p_{ij}^{(a)}(m, t) w_i^{(a)}(m, t); \quad a \in \boldsymbol{A},\ i, j \in \boldsymbol{S},\ t \in \boldsymbol{T},\ m \geqslant 0. \tag{4}$$

The first approach, corresponding to (3), is to assess the probabilities of the transition destinations first, then decide on the holding times with respect to those transitions. In this case, we interpret the probability given in (1), which is a function $f(t)$ of time, as an eventual transition probability, i.e., the probability of event $y$ over those of other outcomes of $\boldsymbol{y}$, given that event $x$ occurs and action $a$ is taken at the decision stage at time $t$. The collection of conditional distributions in the influence view will then constitute the eventual transition probabilities $P_{ij}^{(a)}(t)$, where $i, j \in \boldsymbol{S}$, $a \in \boldsymbol{A}$, $t \in \boldsymbol{T}$ for the underlying SMDP. We then estimate the holding time PMFs directly for the corresponding transitions. In the clinical context, this approach first determines the transition destination of a patient, and then depending on this destination, estimates the time duration spent in the current state before making the transition. For instance, there is a probability of 0.3 that a patient who has undergone surgery A will develop complication B; the probability for such a patient to develop the complication is 0.2 in the first month after surgery, 0.5 in the second month, and 0.3 in the third month.

The second approach, corresponding to (4), is to first decide on the waiting times, and then assess the probabilities of transition destinations with respect to those waiting times.

In this case, we directly estimate the waiting time PMFs for each state $s \in \boldsymbol{S}$. We then interpret the probability given in (1), which is now a function $f(m,t)$ of both duration and time, as a conditional transition probability, i.e., the probability of event $y$ over those of other outcomes of $\boldsymbol{y}$, given that event $x$ occurs, action $a$ is taken, and the waiting time is $m$ at the decision stage at time $t$. The collection of conditional distributions in the influence view will then constitute the conditional transition probabilities $p_{ij}^{(a)}(m,t)$, where $i, j \in \boldsymbol{S}$, $a \in \boldsymbol{A}$, $t \in \boldsymbol{T}$, $m \geqslant 0$ for the underlying SMDP. In the clinical context, this approach first estimates the time duration a patient spends in the current state before making a transition, and then depending on the this duration, determines the transition destination. For instance, the probability that a patient who has undergone surgery A will remain well is 0.2 for a month after surgery, 0.5 for two months and 0.3 for three months; if the patient is well, the probability for him to develop complication B is 0.6 in the first month after surgery, 0.5 in the second month, and 0.3 in the third month.

Validity, adequacy, and relevance of the numeric parameters are determined by established techniques when appropriate. Such techniques could be incorporated into the DynaMoL ontology in future.

In the case study, the distributions involved are estimated from the medical literature and derived from statistical tables such as life tables. Since the relevant data for the occurrence of an event are usually reported in terms of yearly or monthly rates instead of probabilities, the following equation is used to convert a constant rate $r$ into a probability

$$P = 1 - \mathrm{e}^{-r\tau} \tag{5}$$

where $\tau$ is the unit of the rate with respect to the time unit of the decision horizon, e.g., to derive a monthly probability, $\tau = 1$ for a monthly rate, and $\tau = 1/12$ for a yearly rate.

In the case study, the first dynamic decision problem is formulated as a Markov decision problem, the second as a semi-Markov decision problem. Most of the numerical parameters are identical in both problems. For the second problem, we assume that the relative rate of bleeding for warfarin varies with the duration $m$ since entering a state $s$ according to a function of the form

$$A + B\mathrm{e}^{-Cm} \tag{6}$$

where $A, B, C \in \mathbb{R}^+$. With respect to this varying risk, we assess the corresponding conditional distributions of the event variables as the conditional transition probabilities. We assume the waiting time PMFs for the states affected by the varying relative risk of bleeding for warfarin are functions of duration $m$ with the same form as indicated in (6). This means that a patient on warfarin is more likely to make a transition out of that state sooner than later.

### 6.5. *Constraint declaration*

Definitions of the model components as described can be subjected to a set of general or domain-specific constraints $K$. Many constraints are inherent in the definitions supported by DynaMoL. For instance, conditional on an action $a \in \boldsymbol{A}$, an absorbing state $i \in \boldsymbol{S}$ has a zero value function and has no outgoing transitions to other states $j \in \boldsymbol{S}$. A similar example is the relevance of a particular set of event variables in describing a state transition

conditional on an action. These constraints are imposed during model construction; their effects are explicitly encoded in the definitions of the structural or numerical parameters in the dynamic decision model.

There are some constraints whose effects need to be explicitly interpreted during inter-level or inter-perspective translations. A major example is the *disjunctive definition*, or "*partial* OR-gate", for event combinations. The partial OR-gate is analogous to a *canonical model* for specifying Bayesian networks. Canonical models are default strategies for specifying the conditional distributions of the chance variables in a Bayesian network; they are used when detailed interactions among the variables are unavailable, too numerous to elicit, or too complex to be determined precisely [55]. A common canonical form used to reduce the number of conditional probabilities that need to be assessed is *disjunctive interaction*, or "*noisy-OR-gate*".

The partial OR-gate is devised to facilitate conditional distribution specifications of event variables. Formally, the constraint has the general form:

$$e_1 \vee e_2 \vee \cdots \vee e_k \Rightarrow e_c$$

where $e_1, e_2, \ldots, e_k$ are the *constraining events* and $e_c$ is the *consequent event*. The constraint is read as: "if $e_1$ or $e_2$ or ... or $e_k$, then $e_c$". The constraint is imposed on the specification and interpretation of the conditional distribution table of an event variable $\boldsymbol{x}$. The constraining events are outcomes of the probabilistic predecessors of $\boldsymbol{x}$; the consequent event is an outcome of $\boldsymbol{x}$.

With respect to the generalized noisy OR model developed by Srinivas [65], of which the binary noisy OR-gate as described in [55] is a special case, the partial OR-gate is different in the following manners. First, in the DynaMoL vocabulary, the former is a function on the event variable space $\boldsymbol{E}$, while the latter is a function on the event space $\Omega_E$. Second, in the former, the combination constraint is imposed on an event variable and all its predecessors, while in the latter, the combination constraint can be imposed on an event variable and any subset of its predecessors, hence the name "partial".

In DynaMoL, a user can explicitly specify the partial OR-gate constraints for the different event variables in terms of a set of logical statements as shown above. The conditional distribution tables with the correct dimensions and labels are automatically created. The numerical parameters involved can then be specified accordingly. For instance, we want to express: "a patient is dead if he dies from natural causes or dies from a stroke or dies from cerebral hemorrhage or dies from gastro-intestinal bleeding". This can be expressed as the partial OR-gate constraint:

If Die or Die-TE or Die-CHM or Die-GIB then Dead-A

where the consequent event Dead-A is an outcome of the state attribute variable **vital status**, and the constraining events are outcomes of the probabilistic predecessors of **vital status** in the influence view as shown in Fig. 7.

Numerical restriction and order restriction on the actions are currently expressed directly in terms of extra state attribute variables. Additional grammar constructs and their corresponding translators may be similarly incorporated to manage such constraints in DynaMoL.

## 7. Model translation

DynaMoL employs a set of translators to support multiple perspective reasoning and incremental language extension. Structural and numerical parameters specified in one perspective are automatically translated and updated into other perspectives, so that the modeler can work with the most convenient and effective perspective or view at any stage of the modeling process.

### *7.1. Translating transition view into SMDP*

When completely and consistently specified, the components in the transition views correspond directly to the definitions of an SMDP. Hence no special translators are needed in this case. Details of the model, however, usually cannot be easily specified directly in the transition views; inter-perspective translations are almost always involved in translating constructs in other views into those of the transition views.

Once translated into a transition view, reversed translation back to the influence view or tree view is not supported. This is because the structural and numerical information of the influence view or tree view is "compiled" or aggregated into the transition functions in the transition view. If the transition functions are subsequently changed, the loss of information cannot be recovered with respect to the original influence view or tree view.

### *7.2. Translating influence view into transition view*

The mathematical definitions of SMDP do not include event variables and influences. The conditional probabilities captured in the event variables and the corresponding influences should be translated into the one-step transition functions characterizing the transitions among the states.

The *influence view translator* algorithm is based on the expectation of conditional probabilities; it is analogous to the chance node reduction algorithm in influence diagrams [62]. Given random variables $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$, conditional expectation of $\boldsymbol{z}$ given $\boldsymbol{x}$ with respect to $\boldsymbol{y}$ means, for $x \in \Omega_{\boldsymbol{x}}$, $y \in \Omega_{\boldsymbol{y}}$, $z \in \Omega_{\boldsymbol{z}}$, where $\Omega_{\boldsymbol{e}}$ is the outcome space of the event variable $\boldsymbol{e}$:

$$P\{z \mid x\} = \sum_i P\{z \mid y_i, x\} \cdot P\{y_i \mid x\}. \tag{7}$$

With reference to Fig. 4, the overall idea of the algorithm is to reduce the intermediate event variable nodes between the two state variable nodes, so that the final diagram contains only a direct influence between the two state variables. Assuming static action space and static state space, the conditional distribution table associated with the next-state variable on

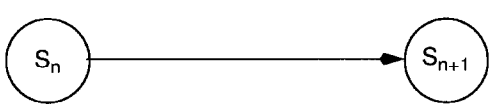


Fig. 11. Final influence view after reduction.

the right contains the set of PMFs or CDFs for the one-step transition functions or their components, conditional on an action.

In essence, the influence view translator iteratively identifies an event variable node to be reduced, updates the conditional distributions of other event variables affected by it, and removes it. The main algorithm is as follows:

```
INFLUENCE-VIEW-TRANSLATOR (ID)
  ◊ ID is a sequence of event-variable-nodes
  x ← FIND-REMOVABLE-EVENT-VARIABLE (ID)
  while x
    do ID ← ABSORB-EVENT-VARIABLE (ID, x)
       x ← FIND-REMOVABLE-EVENT-VARIABLE (ID)
  return ID
```

An event variable node is removable only if it has a single probabilistic successor. A simple heuristic is employed to remove event variable nodes with smaller conditional distribution dimensions first. This heuristic helps keep the size of the conditional distribution tables to be updated as small as possible.

```
FIND-REMOVABLE-EVENT-VARIABLE (ID)
◊ ID is a sequence of event-variable-nodes
E-list ← ∅
for each x ∈ ID
  unless x = state-variable or x = next-state-variable
    do if  length[successors[x]] = 1
       then  E-list ← E-list ∪ {x}
◊ Sort elements of E-list according to the increasing number of predecessors nodes
and the increasing size of (lone) successor.
return  first[E-list]
```

To remove an event variable node $x$, its lone successor $y$ must inherit its predecessors, and the conditional distribution of $y$ must be updated accordingly. Consequently, $x$ will no longer be a successor to all its predecessors.

```
ABSORB-EVENT-VARIABLE (ID, x)
  ◊ ID is a sequence of event-variable-nodes, x is an event-variable-node.
  D_x ← distribution-table[x]
  y ← first[successors[x]]      ◊ Assume x has only one successor.
  D_y ← distribution-table[y]
  predecessors[y] ← predecessors[x] ∪ predecessors[y]
  distribution[y] ← new-distribution ()
  UPDATE-DISTRIBUTION-TABLE (distribution-table[x], D_x, D_y)
  for each p ∈ predecessors[x]
    do successors[p] ← successors[p] ∪ {y}\x
  return ID\x
```


Conditional distribution table for b:

| "P(c\|r)" | B1 | B2 | B3 |
|---|---|---|---|
| A | 0.70 | 0.20 | 0.10 |
| NOT-A | 0.30 | 0.50 | 0.20 |

Conditional distribution table for c:

| "P(c\|r)" | C | NOT-C |
|---|---|---|
| (A B1) | 0.70 | 0.30 |
| (NOT-A B1) | 0.60 | 0.40 |
| (A B2) | 0.45 | 0.55 |
| (NOT-A B2) | 0.10 | 0.90 |
| (A B3) | 0.20 | 0.80 |
| (NOT-A B3) | 0.30 | 0.70 |

Updated conditional distribution table for c, after b is removed:

| "P(c\|r)" | C | NOT-C |
|---|---|---|
| A | 0.60 | 0.40 |
| NOT-A | 0.29 | 0.71 |

Calculation methods:

P(**C**|**A**) = (0.70)(0.70) + (0.45)(0.20) + (0.20)(0.10) = 0.60
P(**NOT-C**|**A**) = 1 - P(**C**|**A**) = (0.30)(0.70) + (0.55)(0.20) + (0.80)(0.10) = 0.40
P(**C**|**NOT-A**) = (0.60)(0.30) + (0.10)(0.50) + (0.30)(0.20) = 0.29
P(**NOT-C**|**NOT-A**) = 1 - P(**C**|**NOT-A**) = (0.40)(0.30) + (0.90)(0.50) + (0.70)(0.20) = 0.71

Fig. 12. An example to show conditional distribution table updating.

Updating the conditional distribution table of the successor event variable begins with the proper combination of conditioning events with respect to its new predecessors. Recall that the default row and column indices in the conditional distribution table are conjunctions of the conditioning events; each event in the sequence constituting an index is an outcome of a predecessor event variable. The conditioning events from the event variable to be removed must be filtered from each combination appropriately. The conditional probability entries are then updated according to (7). Fig. 12 shows a simple example of the calculations involved.

Assuming that each event variable involved has $m$ outcomes and $n$ predecessors on average, the number of conditional distribution table entries to be updated is $O(m^{n+1})$.

### *7.3. Translating tree view into transition view*

Translation from the tree view into the transition view is again based on the expectation of conditional probabilities as shown in (7). With reference to Fig. 5, the main idea is to reduce the intermediate event variable nodes between the two state variable nodes, so that the final diagram contains only direct probabilistic dependences between the two state variable nodes as shown in Fig. 13. Assuming static action space and static state space, the conditional distribution entries associated with the state variable outcomes on the right

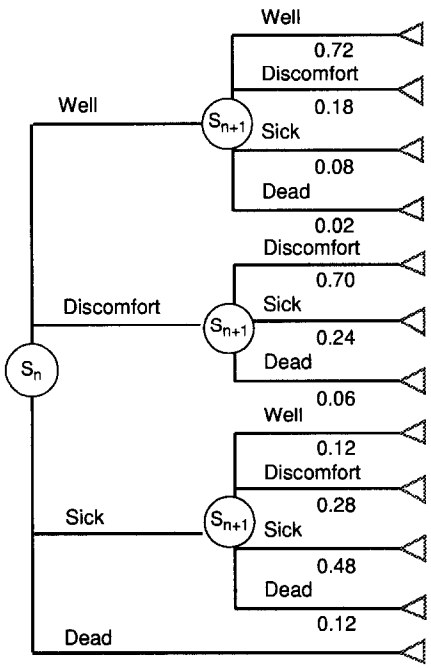


Fig. 13. Final tree view after reduction.

indicate the PMFs or CDFs for the one-step transition functions or their components, conditional on an action.

The tree view translator updates the corresponding conditional distribution entries by traversing depth-first and calculating the expected conditional probabilities along the branches leading to each of the next-state variable outcomes concerned. The path probabilities are simply propagated forward by multiplication into the outcomes of the down-stream event variable nodes in a breadth-first manner. The final path probabilities are then summed up to established the conditional distributions for the outcomes or states of the next-state variable node. The main algorithm is as follows:

**Tree-VIEW-TRANSLATOR** (TD)

◊ TD is a sequence of event-variable-nodes
**for** each x ∈ TD
  **do** slist ← successors[x]
    PROPAGATE (x, slist)
◊ Assume SUM-PATHS sums up the final path probabilities
olist ← outcomes[state-variable]
**for** each o ∈ olist
  **for** each ns ∈ next-state-variable[o]
    **do** SUM-PATHS (ns)
**return** TD

The propagation procedure simply multiplies the conditional probability of each outcome of an event variable into those of its successors.

**PROPAGATE** (x, Elist)

◊ x is an event variable node, Elist is a sequence of event-variable-nodes
olist ← outcomes[x]
◊ olist is a sequence of ordered pairs <l, prob>, where l is the label of an outcome, and prob is the conditional probability associated with the outcome,
**for** each o ∈ olist
  **for** each e ∈ Elist
    **do** solist ← outcomes[e]
      **for** each s ∈ solist
        **do** prob[s] = prob[s]*prob[o]

Assuming that each event variable involved has $m$ outcomes and $n$ predecessors on average, the number of conditional distribution entries to be updated is again $O(m^{n+1})$.

*7.3.1. Translating between influence view and tree view*

The influence view can be regarded as a special form of influence diagram, and the tree view a special form of decision tree. Both the influence view and the tree view do not contain any decision nodes; the values are associated with the outcomes of the special chance nodes, namely, the next-state variable nodes. Moreover, both the influence view and the tree view are always *oriented* with the state variable node as the source, and the next-state variable node as the sink. Many theoretical results concerning the influence diagram and the decision tree, therefore, can be applied directly or in a simplified manner to the influence view and the tree view.

The main idea in direct translation between the representations of any two event variable nodes in the influence view and the tree view is shown in Fig. 14. The number of entries in the conditional distribution tables in the influence view and the number of branches in the tree view can be generalized to non-binary event variables. As mentioned in Section 6.3.3, asymmetric relations can be captured similarly with the proper zero-valued entries reflected in the conditional distribution tables in the influence view, and the proper omission of branches in the tree view.

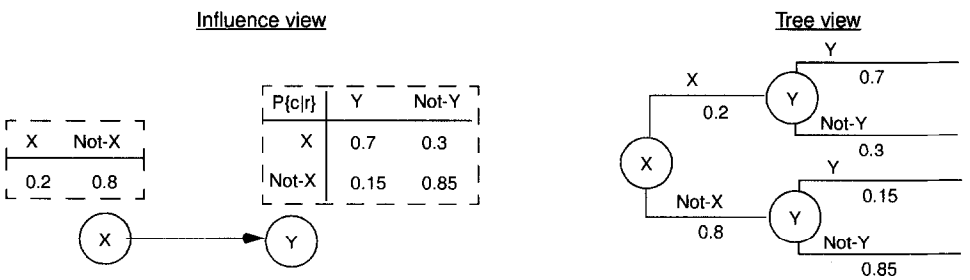


Fig. 14. Direct translation between an influence view and a tree view for an action.

In translating an influence view into a tree view, the event variable nodes in the influence view are first ordered in terms of their maximal distances from the state variable node, i.e., the source; the next-state variable node, i.e., the sink, is always at the end of the list. These nodes and their respective outcomes and associated conditional probabilities are then expanded into the nodes and branches in a connected tree. The main algorithm is as follows:

```
INFLUENCE-VIEW-TO-TREE-VIEW-TRANSLATOR (ID)
  ◊ ID is a sequence of event-variable-nodes
  ID ← ORDER (ID)
  TD ← ∅
  ◊ Assume EXPAND properly attaches a node and all its outcomes and the
  corresponding conditional probabilities as branches to each of the leaves
  in the tree
  for each e ∈ ID
    do EXPAND (e, TD)
  return TD
```

The ordering procedure performs two functions: first, it determines the minimum and maximum distances of the event variable nodes from the source node; second, it sorts the event variable nodes in increasing order, first with respect to their maximum distances and then, for those nodes of equal maximum distances, with respect to their minimum distances from the source. This heuristic ensures that all the predecessors of any event variable node are positioned to the left side of that node in the resulting tree.

```
ORDER (Elist)
  ◊ Elist is a sequence of event-variable-nodes
  ◊ dis[e] for an event variable node e is an ordered pair <min, max>, where min
  is the minimum distance and max is the maximum distance from the
  state-variable node.
  for each e ∈ Elist
    do dis[e] ← <0,0>
  for each e ∈ Elist
    unless e = state-variable
      do slist ← successors[e]
      for each s ∈ slist
          do if min[dis[s]] = 0
            then min[dis[s]] = max[dis[e]] + 1
                 max[dis[s]] = MAX(max[dis[s]], max[dis[e]] + 1)
            else max[dis[s]] = MAX(max[dis[s]], max[dis[e]] + 1)
  ◊ Assume DIS-SORT properly sorts the marked event-variable nodes according
  to their distances from the state-variable node.
  DIS-SORT (Elist)
  return Elist
```

The complexity for the marking part of the ordering algorithm is $O(|\boldsymbol{E}| + |\Psi|)$, where $\boldsymbol{E}$ is the set of event variables and $\Psi$ is the set of probabilistic influences in the influence view,

and that for the sorting part is $O(|\boldsymbol{E}|^2|)$ (or $O(|\boldsymbol{E}|\log|\boldsymbol{E}|)$ on average) using the quicksort algorithm.

In translating a tree view into an influence view, the event variable nodes in the tree view are first aligned so that all the nodes representing the same event variable are at the same depth in the tree from the state variable node, i.e., the root. These nodes and their respective outcomes or branches and associated conditional probabilities are then extracted into the nodes and tables in the influence view. The main algorithm is as follows:

```
TREE-VIEW-TO-INFLUENCE-VIEW-TRANSLATOR (TD)
  ◊ TD is a sequence of event-variable-nodes
  TD ← ALIGN (TD)
  ID ← ∅
  ◊ Assume EXTRACT collects all the information pertaining to a single
  event-variable at a particular depth in the tree view and updates the outcomes and
  the corresponding conditional probabilities in the influence view.
  for each x ∈ TD
    do EXTRACT (x, ID)
  return ID
```

The aligning procedure simply assigns a depth number to all the event variable nodes in the tree view. The asymmetric cases are automatically taken care of by the relative positions of the nodes in the tree. The event variable nodes are then sorted according to their depths from the root node in the tree view.

```
ALIGN (Elist)
  ◊ Elist is a sequence of event-variable-nodes
  ◊ depth[e] for an event variable node e is the distance from the root node
  for each e ∈ Elist
    do depth[e] ← 0
  for each e ∈ Elist
    unless e = state-variable
      do slist ← successors[e]
        for each s ∈ slist
          do depth[s] = depth[e] + 1
  ◊ Assume DEPTH-SORT properly sorts the marked event-variable nodes
  according to their depths from the state-variable node.
  DEPTH-SORT (Elist)
  return Elist
```

The complexity for the marking part of the aligning algorithm is $O(|\boldsymbol{E}|^m)$, where $\boldsymbol{E}$ is the set of event variables and $m$ is the average number of outcomes, and that for the sorting part is $O(|\boldsymbol{E}|^{2m})$ (or $O(|\boldsymbol{E}|^m \log|\boldsymbol{E}|^m)$ on average) using the quicksort algorithm. The inherent exponential nature of the tree representation is reflected in the complexity of these algorithms.

The translations as described above are complicated by the following situations:

- Translating an influence view into a tree view loses the information on conditional independence among the event variables.
- Translating a tree view into an influence view loses the information on chronological ordering among the event variables.

The current translators in the DynaMoL framework employ interface heuristics to "memorize" the relevant information in each view, and prompt the user for the required specifications if necessary. A general technique for automated detection and conversion of conditional independence and chronological ordering will be incorporated into the framework in the near future [78].

### *7.4. Handling constraints in translations*

Most constraints have direct correspondence in the SMDP representation. Each state variable, for instance, corresponds to the state-space at a particular time or decision stage in the SMDP. No explicit translators are devised for such constraints. The only explicit translator currently defined in DynaMoL is a translator for the partial OR-gate constraint in the influence view. It is encoded as a special version of the update distribution table algorithm described earlier.

In translating the influence view into the transition view, a new algorithm is used to update the conditional distribution entries when the event variable $\boldsymbol{x}$ to be removed, or its lone successor $\boldsymbol{y}$, or both contain conditional distribution tables with the partial OR-gate constraints. The algorithm simply propagates the disjunctive outcome labels properly into the successor of the event variable to be removed in the influence-view-translator.

In translating the influence view into the tree view, the disjunctive outcome labels in a conditional distribution table with the partial OR-gate constraint is properly interpreted into the corresponding asymmetric branches in the tree view.

The number of table entries or branches to be updated is of about the same order as before: $O((m-1)^{n+1})$ where $m$ is the average number of outcomes and $n$ is the average number of predecessors of an event variable. For the influence view of the case study, the state variable and the next-state variable have 18 and 19 outcomes each, and the other event variables have an average number of 3 outcomes; the average number of predecessors for an event variable is 3. To reduce an event variable from an influence view, the number of table entries that need to be calculated is of the order of 8700; the addition of one more predecessor would make it to the order of 26000! The smaller dimensions of the conditional distribution tables rendered by the partial OR-gate, however, drastically cuts down the sizes of the intermediate conditional distribution tables in practice.

### *7.5. Maintaining consistency*

Decision modeling in DynaMoL involves working with different graphical views at the same time. Modifying the constructs in one view may render the information in another view obsolete. There are two types of consistency management involved in DynaMoL:

- *Intra-view consistency* ensures that the structural and numerical parameters in a particular view are specified correctly, e.g., the probabilities add up to 1, the number

of states displayed are in accordance with the state-space, etc. Such consistency is maintained by incorporating simple checking mechanisms in the interfaces.

- *Inter-view consistency* ensures that updated information is reflected in all the relevant views when the translators are invoked. For unidirectional translators, e.g., translating event variables and influences into transition functions, we have to make sure that the target view is updated, e.g., new transition arcs with associated one-step transition functions are displayed in the transition view. Under this convention, however, information of the translation origin may become obsolete when the target is modified, e.g., updating a transition function directly after translating the influence view. A consistency table can keep track of the translations invoked during a modeling session, and a flag is signalled if the relevant target information is modified.

For bi-directional translations, e.g., updating parameters with direct correspondence in different views, modifications are reflected throughout the model; inconvenience is minimized by requiring that changes are reflected only when they are saved or confirmed in the interfaces.

## 8. Model solution

Once the dynamic decision model is formulated and translated into an SMDP, the model can be solved by any solution methods for SMDPs. A variety of solution methods are available for SMDPs. Based on the characteristics of the dynamic decision problem concerned, we can choose the most efficient solution method available. The major solution methods incorporated in the current version of DynaMoL are based on *value iteration*.

### 8.1. Value iteration

A dynamic decision problem can be expressed as the *dynamic programming equation*, or Bellman optimality equation, of an SMDP. The main idea in solving the equation is through solving the subproblems at each decision stage via an optimizing "divide and conquer" approach, working backward from the final outcomes at the end of the decision horizon, pruning sub-optimal branches along the way.

In the specialized case of an MDP, all the state transitions out of any state $i \in \boldsymbol{S}$ occur at the unit time interval or duration of 1. For a decision horizon of $n$ stages, with a discount factor $0 \leqslant \beta < 1$, the optimal value achievable in any state, $i \in \boldsymbol{S}$, $V_i^*$, given an initial value $V_i(0)$ and current time $t \in \boldsymbol{T}$, is:

$$V_i^*(n, \beta, t) = \max_a \left\{ v_i^{(a)}(1) + \beta \sum_j P_{ij}^{(k)}(t) V_j^*(n-1, \beta, t+1) \right\};$$
$$n \geqslant 0,\ i, j \in \boldsymbol{S},\ a \in \boldsymbol{A},\ t \in \boldsymbol{T} \tag{8}$$

where $P_{ij}^{(a)}(t)$ is the transition probability from state $i$ to $j$, conditional on action $a$ at time $t$, and $v_i^{(a)}(1)$ is the value achievable in state $i$, conditional on action $a$, for one unit of time.

In the general case of an SMDP, the state transitions out of any state $i \in \boldsymbol{S}$ may occur at different time durations after entering the state. For a decision horizon of $n$ stages, with a

discount factor $0 \leqslant \beta < 1$, the dynamic programming equation for calculating the optimal value achievable in any state, $i \in \boldsymbol{S}$, $V_i^*$, given an initial value $V_i(0)$ and current time $t \in \boldsymbol{T}$, consists of two parts:

- The first part indicates the expected value achievable if the next transition out of state $i$ occurs after duration $n$, where the values achievable beyond the decision horizon are ignored; and
- The second part indicates the expected value achievable if the next transition occurs before duration $n$, where the values achievable in the state $i$ for the duration in the state and the values achievable for transitions out of the state $i$ are both taken into account.

The equation, which incorporates the first part above as the first addend and the second part as the second addend, is as follows [35]:

$$V_i^*(n,\beta,t) = \max_a \left\{ \sum_j \sum_{m=n+1}^{\infty} q_{ij}^{(a)}(m,t) \left[ \sum_{l=0}^{n-1} \beta^l y_i^{(a)}(l) + \beta^n V_i(0) \right] + \sum_j \sum_{m=1}^{n} q_{ij}^{(a)}(m,t) \left[ \sum_{l=0}^{m-1} \beta^l y_i^{(a)}(l) + \beta^m b_{ij}^{(a)} + \beta^m V_j^*(n-m,\beta,t+m) \right] \right\};$$
$$n \geqslant 0,\ i,j \in \boldsymbol{S},\ a \in \boldsymbol{A},\ t \in \boldsymbol{T} \tag{9}$$

where $q_{ij}^{(a)}(m,t)$ is the one-step transition function and $v_{ij}^{(a)}(m) = [\sum_{l=0}^{m-1} y_i^{(a)}(l)] + b_{ij}^{(a)}$ is the transition value function such that

$$v_i^{(a)}(m) = \sum_j q_{ij}^{(a)}(\cdot) v_{ij}^{(a)}(m)$$

for all $i, j \in \boldsymbol{S}$, $a \in \boldsymbol{A}$, $t \in \boldsymbol{T}$ and $m \geqslant 0$.

The value iteration solution method is to solve the optimality equation shown in (8) or (9). The solution to such an equation is an optimal policy

$$\pi^* = \{\mu_0^*, \mu_1^*, \mu_2^*, \mu_3^*, \ldots\}$$

where $\mu_t^* : \boldsymbol{S} \to \boldsymbol{A}$ and $t \in \boldsymbol{T}$, that maximizes $V_{init}^*(N,\beta,0)$, the optimal expected value or reward for an initial state over duration $N$, at time $t = 0$. Recall that $n = N - t$ is the remaining duration for the decision horizon.

For a decision horizon of duration $N$, the complexity of the value iteration algorithm for SMDP is $O(N^2 \cdot |\boldsymbol{A}| \cdot |\boldsymbol{S}|^2)$, since in each decision stage, we generally need to consider all time points $t'$ such that $t \leqslant t' \leqslant t + n$. The complexity of the corresponding algorithm for MDP is $O(N \cdot |\boldsymbol{A}| \cdot |\boldsymbol{S}|^2)$; in this case there is a one-to-one correspondence between the duration $n$ and the current time $t$: $n = N - t$.

All the default solution methods in existing dynamic decision modeling frameworks are based on value iteration. While probabilistic inference techniques can also be employed for solving dynamic influence diagrams [63], the graph reduction algorithm [70] corresponds directly to Markov value iteration. Cohort analysis and Monte Carlo simulation in Markov

cycle trees, however, are based on conditional expectation in a forward manner; the complexity of these algorithms is $O(|S| \cdot (|A| \cdot |S|)^N)$ since they do not make use of the memorized optimal substructures inherent in dynamic programming techniques.

The first clinical question for the case study is posed as a discrimination problem; the numerical parameters are assessed in accordance with an MDP, i.e., with constant holding times. The solution produced by Markov value iteration is a set of two policies, corresponding to the four strategies considered; each policy includes the expected value achievable for all possible starting states and all possible decision stages. We assume that the patient has a probability of 0.25 to be in atrial fibrillation, has no history of thromboembolism, and is not on warfarin at the beginning of the decision horizon. The results indicate that administering only warfarin initially is the preferred strategy over a long-term decision horizon of 50 years or 600 months.

The second clinical question for the case study is posed as an optimization problem; the numerical parameters are assessed in accordance with an SMDP. The solution produced by semi-Markov value iteration is an optimal policy for all possible starting states and all possible decision stages. We adopt the same assumptions about the condition of the patient as mentioned. For a short-term decision horizon of 5 years or 60 months, the results indicate that administering only warfarin initially is the preferred strategy up to 8 months. After 8 months, if the patient remains in the same condition, not administering any drug initially is preferred.

### 8.2. Other methods

Solution methods reported in the MDPs and SMDPs literature such as *fundamental matrix solution*, *policy iteration* [34], *adaptive aggregation* [4], or *linear programming* are directly applicable if certain assumptions or conditions are met. These conditions include stationary policies, constant discount factors, homogeneous transition functions, etc.

As mentioned in Section 2.2.4, recent research in DTP has also led to new solution methods for MDPs [1,7,17–19,22,23,29–31,43,67]. These methods address the trade-off between solution optimality and computation cost in complex and time-critical decision domains. Some of them, however, may required that the MDPs be formulated in special formats specific to each framework.

Besides the solution methods for MDPs and SMDPs, other solution methods that take advantage of the high-level DynaMoL ontology can also be employed.

### 8.3. Separating modeling and solution support

In summary, by separating the modeling task supported by the decision grammar and graphical presentation, and the solution task supported by the mathematical representation, a large collection of solution methods can be employed in DynaMoL. Moreover, employing a new solution method does not involve any change to the high-level modeling language itself; all solution methods reference only the mathematical representation of a model. Similarly, extending or adapting the decision grammar and graphical presentation do not affect the solution methods already applicable; these may, however, admit other solution methods that make use of the additional constructs.

## 9. Model analysis

Model analysis is performed on a dynamic decision model to ensure "correctness" of the model and robustness of the solution. Many sensitivity analysis techniques in decision analysis, e.g., clarity test [36], risk profiles, tornado diagrams, one- or multi-way analyses are directly applicable. In DynaMoL, structural parameter analysis is usually done by removing or adding some event variables; numerical parameter analysis by changing some of the transition and value functions in the SMDP and invoking the solution methods. The influence view supports structural parameter analysis by allowing direct manipulation of the event variables. The tree view supports careful examination of the numerical parameters by explicitly displaying all such information at the same level as the structural organization.The transition view helps to detect improperly specified structural and numerical parameters. For instance, there should not be a transition from a state where warfarin is ineligible to one where warfarin is administered.

The quality of the model can be assessed in terms of its "requisiteness", which can in turn be interpreted in terms of its accuracy, conciseness and clarity. A model is *accurate* if it is well-formed and contains all the relevant information that sufficiently reflects the decision situation. A model is *concise* if it contains no redundant information. A model is *clear* if it contains information that can be easily accessed by an inference process to produce "meaningful" answers. Therefore, the quality metrics should not only be theoretically meaningful, but also indicate how easily the model can be "debugged" during problem formulation and problem analysis in practice.

Some relevant metrics for assessing model accuracy include the equivalence decision class analysis for determining relevant chance events [58], the value of information, and the "confidence" measure of the recommended decision, which compares the fidelity of subjective probabilities with objective relative frequencies in the model [50,76]. While most of these metrics are applicable to any dynamic decision model type, actual implementations of the metrics, and hence their ease of use, vary with different model types. For instance, equivalence decision class analysis is usually employed in updating dynamic influence diagrams; the structural complexity of the tree-based decision models renders such updating much more difficult. In addition, some metrics are designed with specific model types in mind, e.g., structural controllability and observability in influence diagrams [11].

The main difficulty in formulating and analyzing an SMDP is the combinatorially increasing dimension of the state space with each relevant state attribute variable. One way to ensure conciseness of the state space description is to explicitly manipulate the combinations of state attributes with either formal *canonical forms* or domain-specific heuristics. Another way is to develop quantities called *sufficient statistics*, which would be of smaller dimension than the original state space, and yet summarize all the essential information for problem solution and analysis [4]. Currently, however, there are no effective guidelines for developing such techniques in general.

The quality of the solution to a dynamic decision model is usually determined, with respect to domain-specific knowledge, using standard statistical techniques such as one-way and two-way sensitivity analysis. The mathematical representation of SMDP facilitates analysis of the solution nature. Much insights can be gained about the solution

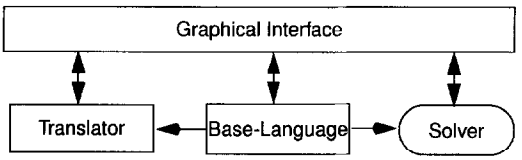


Fig. 15. The DYNAMO system architecture.

by examining the nature of the state space, action space, and other decision parameters. For instance, if absorbing states are present in a well-formed SMDP, convergence is guaranteed in value iteration. The accuracy of the SMDP can also be assessed in terms of the mathematical properties that can be proved about it, e.g., existence and convergence of optimal policies. Some methods that facilitate such proofs are the certainty equivalence principle, the controllability and observability conditions, and the monotonicity and contraction mapping principle [4]. Due to the advanced mathematical definitions, however, such techniques may be difficult to apply and to understand in practice.

For the long-term discrimination problem in the case study, the sensitivity results indicate that administering only warfarin but not Quinidine to the patient is the dominant strategy for all reasonable ranges of numerical parameters involved. For the short-term optimization problem, the sensitivity results demonstrate that over the decision horizon, the preferred strategy shifts in a reasonable manner from initially administering warfarin to no drug, depending on the desirable and the undesirable effects of warfarin.

## 10. The DYNAMO system

The system architecture of a prototype implementation of DynaMoL, called DYNAMO, is shown in Fig. 15. The system is implemented in Common Lisp with the GARNET graphics package [49]. In the figure, the blocks indicate system components; the arrows indicate information inflows. The graphical user interface allows interactive model specification. The base-language defines the model components. The translator contains the correspondence rules among the model components and the SMDP. The solver implements several solution methods for SMDPs.

### *10.1. System implementation*

All the components of DynaMoL described in the preceding sections have been implemented in the current version of DYNAMO. In particular, the system supports the full dynamic decision ontology as described in Section 4, the complete graphical presentation of transition view, influence view, and the corresponding translators, and a functioning version of the tree view, partial tree view, and the corresponding translators from/to the influence view and to the transition view.[2] The solution methods implemented include value iteration for solving discrimination problems and optimization problems, policy

[2] The current version of implemented algorithms and preliminary evaluation results for the tree views and their translators are documented in [78].

iteration, adaptive aggregation, and linear programming[3] for both the Markov and the semi-Markov cases. A sophisticated set of practical support tools including editors and interfaces for the decision parameters and constraints, data visualizing tools, and sensitivity analysis tools are also being developed.

Figs. 6 and 7 are actual screen snapshots from the DYNAMO system. The system is under licensing negotiations to be commercialized; a copy of the software on restricted release for non-commercial use is available upon request.

### *10.2. System evaluation*

The DYNAMO system, and hence the DynaMoL framework, has been evaluated in several practical domains as follows:

#### *10.2.1. Atrial fibrillation management*

As illustrated in Section 6, we have conducted a comprehensive case study based on an actual clinical decision analysis consult in atrial fibrillation management with the graphical and translation support of the transition views and the influence views. The dynamic decision problems involve 4 different strategies, 20 states, 10 event variables, over time horizons of 600 months or 50 years and 60 months or 5 years. The solution methods employed include both Markov and semi-Markov value iterations. Numerical sensitivity analyses are done on various clinically significant parameters, e.g., rate of embolism, rate of cerebral hemorrhage, etc., to ensure robustness of the optimal policies derived.

The effectiveness of multiple perspective reasoning is demonstrated through the modeling process. Modeling begins with specifying the 20 states in the transition views. Since it is difficult to specify the transition functions directly, the corresponding influence views, one of which is shown in Fig. 7, are created. In this case study, the influence views for the different actions include the same set of event variables with different numerical parameters, but in general they can also be structurally different. Although not performed in this exercise, deliberations on the relationships among the event variables and the underlying numerical parameters can be facilitated by the partial tree views which support the specification of asymmetric relations, as shown in Fig. 10. Translations from the influence views into the corresponding tree views, one of which is shown in Fig. 8, would illuminate the chronological order, if any, of the various event variables, as well as explicitly showing the numerical parameters specified. Visualizing the parameters in the tree views would also allow us to compare the structure of the model with the original model in the Markov cycle tree format. When the fully specified influence views are translated back into the corresponding transition views, a simplified version of which is shown in Fig. 6, the numerical parameters specified are further verified by making sure that the transition links are correctly defined, e.g., the states in which warfarin is being administered should not be reachable from those in which warfarin is ineligible. If discrepancies are detected, the contribution distribution tables of the influence views, or the tree views and the partial tree views, are examined to help rectify the errors.

[3] The implemented algorithms and evaluation results for the policy iteration, adaptive aggregation, and linear programming methods included in the DYNAMO solver are documented in [53].

As compared to the original solutions for the clinical consult, our framework achieves correct answers according to expert judgment and sensitivity analyses, and provides extra capabilities in terms of expressiveness and efficiency for modeling and solving the decision problem. For instance, DynaMoL allows direct accounting of varying relative risk of bleeding for warfarin with respect to the duration of treatment; expression of such duration-dependent factors are difficult in most existing decision modeling frameworks.

*10.2.2. Colorectal cancer follow-up*

We have also conducted a comprehensive case study on deciding the optimal follow-up schedule for colorectal cancer patients who have undergone surgery. The decision context is based on a group of Dukes Stage 3 colorectal cancer patients in the Singapore General Hospital. The objective is to determine the optimal course of diagnostic test and treatment for early detection and management of cancer recurrence, metastasis, i.e., spreading of the cancer, or both recurrence and metastasis. The dynamic decision model is constructed with the graphical and translation support of the transition views, the influence views, and the tree views. In this case, the tree views and the partial tree views directly facilitate the modeling process as described earlier. The chronological order of cancer recurrence, metastasis, and both recurrence and metastasis is explicitly captured in the tree views. The solution method employed is Markov value iteration. Preliminary evaluation results show that the implemented translation algorithms work correctly, and that the optimal policies produced are reasonable as judged by clinical experts [78].

*10.2.3. Dynamic decision problems in other domains*

Finally, we have also evaluated all the implemented solution methods in DYNAMO with a benchmark test suite [53]. The test suite includes randomized examples and decision problems in general domains, e.g., the automobile problem [34]. The sizes of the action space and the state space range from 5 to 25. Both finite-horizon and infinite-horizon problems are considered, with or without discounting.

The exercise has demonstrated the correctness of the implemented algorithms; it has also illuminated the suitable problem characteristics for applying the different solution methods. For instance, linear programming facilitates sensitivity analyses in infinite-horizon problems; for such problems with large state space and/or action space, however, policy-iteration and adaptive state aggregation are preferred. On the other hand, for finite-horizon problems, solution methods involving only stationary policies, i.e., the same strategies to be applied in all the decision stages, are not appropriate for heterogeneous optimization problems with time-dependent transition functions; value iteration is the only suitable method in this case.

## 11. Discussion

### *11.1. Related work*

The DynaMoL framework is mainly motivated by the ideas of and the approaches to dynamic decision making in SMDPs, dynamic decision analysis, and DTP as described in Section 2.2.

Egar and Musen [24] have examined the grammar or higher-level language approach to specifying decision model variables and constraints. This grammar is based on the graphical structure of influence diagrams; it captures prototypical patterns at a high-level abstraction for modeling trade-offs or dilemmas in clinical decision problems. The grammar, however, has not been extended to handle dynamic decision models.

Graphical representation of continuous-time semi-Markov processes has been explored by Berzuini et al. [5] and Dean et al. [21]. Similarly, recent work on dynamic network models by Dagum et al. [14,15] captures the general time-series analysis techniques in Bayesian networks. These frameworks, however, focus only on single perspective presentation of the relevant decision and probabilistic variables as Bayesian networks or influence diagrams.

On integrating dynamic decision models and SMDPs, Provan et al. [57,58] have developed techniques that automatically construct dynamic influence diagrams from the data of the underlying SMDPs, but they only employ algorithms for evaluating dynamic influence diagrams. In this framework, the state attribute variables involved in each decision stage are represented as a Bayesian network; the overall structure corresponds to the dynamic influence diagram representation of an SMDP. Tailoring of the parameters, however, has to be explicitly done for each decision stage.

Magni and Bellazzi [46] have recently adopted the influence view representation in DynaMoL to develop the DT-Planner environment for representing and solving MDPs. The influence view supports parsimonious specification of an MDP in this interactive framework.

### *11.2. Future work and extension*

Besides addressing the trade-off between model transparency and solution efficiency, the language features and system capabilities of DynaMoL can be extended, adapted, and developed for general practical applications. On the theoretical side, we discuss the language enhancement that would allow the system to model and solve a larger class of problems. On the practical side, we propose some research issues that would make the system more effective and convenient to use.

#### *11.2.1. Supporting language extension and adaptation*

Incremental language enhancement is supported in DynaMoL by introducing a set of new grammar constructs or presentation formats or both, and devising a set of translators for them.

(a) *Static versus dynamic spaces.* The DynaMoL decision grammar can be extended to incorporate dynamic state space and dynamic action space. New productions for the corresponding constructs can be written to incorporate the valid time or duration for each state and action. The graphical presentation convention remains mostly unchanged; only the valid entities are displayed for particular time points or decisions stages. Solution methods are readily available for such general classes of SMDPs.

(b) *Automatic state augmentation.* State augmentation is a basic technique for incorporating extra information in SMDPs. Most of the declaratory and strategic constraints, for instance, can be represented as state attribute variables or additional dimensions

of the state space. Many constraints, however, affect only part of the original state space. For example, if a state attribute variable indicates whether the patient has a stroke before, the constraint on the number of stroke only affects those states that involve a stroke. Therefore, it is more convenient to specify such constraints and their associated parameters separately, and have a set of translators to automatically incorporate the information. This involves working with an abstract state space, which will eventually be translated into the full state space for solution and analysis. This approach would allow us to incrementally and separately impose various constraints, without directly dealing with the resulting large and complex state space.

This idea of modeling in an abstract state space for transparency is directly opposite to the idea of execution or solution in an abstract state space for efficiency. The latter idea is adopted as *aggregation* methods in control theory [4], and as HTN planning methods in AI [41,42,61,66,68] and DTP [7,17,22].

Automatic state augmentation can be achieved in DynaMoL by adding a set of grammar constructs for specifying the constraints, and a set of translators for translating them to form a larger state space. Another set of reversed translators may be needed to facilitate modeling at multiple levels of abstraction.

(c) *Limited memory*. In a semi-Markov process, there is a limited memory of the time duration since entry into a state. In some cases, memory about previous states or actions is important in a dynamic decision problem. For example, if a patient had a heart attack before, he would be more susceptible to a second heart attack during surgery. Such limited memory can be incorporated into a semi-Markov or Markov process by state augmentation.

Repeated applications of state augmentation usually lead to an explosion of the state space size. To avoid such explosion, we can relegate the solution methods to keep track of limited memory. Only forward induction based solution methods, however, are applicable. In this approach, a new counter or binding called *memory* or *history* can be introduced to keep track of the relevant states that a process has visited. Calculations of the expected value with respect to the model parameters are now conditional on the history accumulated so far.

A set of history constructs can be introduced in DynaMoL to keep track of the limited memory, and a set of translators for the conditional parametric specification are needed. The solution methods can then operate on both the SMDP and the history constructs.

(d) *Numerical and ordering constraints*. There are several types of numerical and ordering constraints: event or state numerical constraints specify that a certain number of events or states will lead to a specific consequence, e.g., a patient is assumed dead if he has had three strokes. Action numerical constraints restrict the number of times an action may be applied. Action ordering constraints specify that an action must always or never follow another action.

The first method to incorporate such constraints into a dynamic decision model is to augment the state space to keep track of the number or the order of the decision factors. The second method is to introduce a set of counters and let the solution methods take care of the constraints. Again new constructs and translators are needed to incorporate the constraints. To facilitate proper translations, an

external knowledge based system can be employed to choose the right translators by examining the nature of the constraints and solution methods. Other general or domain specific constraints or canonical models can be introduced into DynaMoL in a similar manner to the definition of the partial-OR constraint on event combination described earlier.

(e) *Presentation convention.* In addition to the three graphical presentation perspectives in DynaMoL, other useful presentation perspectives, e.g., 2-D or 3-D visualization of parametric evolution, can be incorporated by introducing new presentation conventions and translators.

*11.2.2. Supporting automated dynamic decision making*

The DynaMoL framework can also serve as the anchor of a new paradigm of dynamic decision making that effectively integrates automatic and interactive formulation, solution and analysis of the relevant problems.

(a) *Automatic derivation of numerical parameters.* One of the most daunting task in dynamic decision modeling is to estimate and specify the numerical parameters involved. In general, given a state-space of size $|S|$, an action-space of size $|A|$, and a decision horizon of duration $n$, the number of probabilistic parameters to be assessed is of the order of $O(|A||S|^2 n)$. Subjective assessments from expert physicians may be adequate in some cases. When the decision situations are complex or the decision dimensions are large, however, the practicality of the modeling approach is limited by the lack of realistic estimations. On the other hand, given a large set of data, objective probabilities may not be easily calculated to support decision modeling; the recording formats, the measurement assumptions, and the processing errors associated with the data may complicate such derivations.

We wish to investigate the issues involved in automatic construction of one-step transition functions from databases. This exercise will provide insights into the feasibility of such a task for supporting dynamic decision modeling. It will also illuminate the limiting constraints inherent in available databases. The lessons learned will contribute toward bridging the expectation gap between the dynamic decision modeler and the database builder. This will in turn encourage integration and advancement of the techniques and facilities provided by both fields.

By incorporating a set of statistical and Bayesian learning methods, we have implemented a parameter learning system that automatically construct probability distributions in the influence views and the transition views of a dynamic decision model in DynaMoL [9]. We have reported some encouraging preliminary insights and results from experimenting with large medical databases in the management insulin-dependent diabetes mellitus [77] and the follow-up management of colorectal cancer surgery patients [9,10].

(b) *Supporting knowledge based model construction.* Knowledge-based decision systems employing knowledge-based model construction (KBMC), e.g., ALTERID [8], FRAIL [28], SUDO-PLANNER [72], and DYNASTY [56], advocate that the decision models for different problems should be constructed on demand from a knowledge base [73]. Currently, each KBMC system synthesizes only one type of decision models, e.g., influence diagrams. In dynamic decision models, the time-dependent con-

straints are usually translated into numbers, equations, or complicated substructures hardwired into specific decision models; subsequent retrieval of the constraints is quite difficult, if not impossible, from these models.

In an on-going project [71], we explore how the high level decision ontology in DynaMoL can facilitate KBMC. DynaMoL provides an expressive and explicit language for formulating dynamic decision problems. This relieves the knowledge base representation and organization from being restricted by the graphical structure of the target model. The resulting models will also support more detailed analysis and more efficient solution methods as argued before.

(c) *Automated knowledge acquisition from multiple knowledge sources.* Automating various model formulation tasks is essential for effective dynamic decision making. With rapid advances in biomedical knowledge and electronic information delivery, support for model building should address such issues as the wide variety of available information sources, their imprecision, and possible disagreements among the different sources.

In another on-going project, we investigate how to acquire and integrate information from different sources such as domain experts, knowledge bases, electronic medical records, on-line registries, and test and treatment protocols on the Internet. The major research issues include selecting relevant information from different sources for model formulation, representing and organizing the integrated information to support model refinement and structural analysis, and facilitating collaborative model formulation among groups of experts, including dealing with issues such as incompleteness of knowledge, possible inconsistency, and disagreements among the experts.

## 12. Conclusions

We conclude this paper by summarizing the achievements and limitations of this work.

### *12.1. A unifying view*

The analysis of three major approaches to dynamic decision making under uncertainty highlighted their similarities and differences, as well as their strengths and weaknesses. Based on a uniform task definition for dynamic decision making under uncertainty, we have explicated the representational and inferential support involved. We propose that SMDPs be regarded the same role in dynamic decision making under uncertainty as first-order predicate calculus (FOPC) in deterministic knowledge representation. Recently, Dean [16] has independently come up with the same idea to regard MDPs as the basis for DTP. We have further illustrated the motivation for our proposal by devising a new methodology based on the idea.

### *12.2. A general paradigm*

Building on the common basis of SMDPs, we have introduced a new general methodology for dynamic decision making under uncertainty. We propose a novel language

design that integrates the desirable features of current techniques. By introducing a new paradigm of multiple perspective reasoning, this design breaks the mold of single perspective reasoning supported in all existing graphical dynamic decision modeling languages. We have also established methods to systematically extend the language ontology. This is in contrast to the fixed vocabularies in most existing techniques.

Model specification in DynaMoL is in terms of a higher-level language than that in existing dynamic decision modeling frameworks. In frameworks such as dynamic influence diagrams or Markov cycle trees, the model parameters need to be explicitly specified in detail. In particular, the declaratory and strategic constraints are explicitly incorporated into the graphical structure of the model; the probabilistic and temporal parameters are explicitly encoded for each time slice or period considered. The dynamic decision grammar in DynaMoL, on the other hand, supports abstract statements about the decision situation, e.g., statements about how the events and states are logically related to each other. These abstract statements are analogous to the macro constructs in conventional programming languages. By focusing on the decision problem ontology instead of the decision model components, DynaMoL provides a more concise and yet more transparent platform for supporting model construction.

The advantages of the graphical nature of existing dynamic decision modeling languages are preserved and extended in DynaMoL. By extending the graphical presentation convention, most model components and constraints can potentially be visualized. The different presentation perspectives further contribute to the visualization ease and clarity.

Theoretically, SMDPs can approximate most stochastic processes by state augmentation or other mechanisms. The resulting state space, however, may be too complex for direct manipulation or visualization. On the other hand, efficient solution methods may not exist for more general stochastic models. By distinguishing the specification grammar and the underlying mathematical model, DynaMoL preserves the clarity and expressiveness of the model structure, while at the same time admits a spectrum of solution methods.

While the different perspectives illuminate the model in different ways, loss of information may occur when the information is stored in a normal form, e.g., as SMDPs. Unless all the perspective information is kept around, later retrieval of the information may be difficult. Therefore, there is an extra burden of information storage. There is also an overhead in translation time.

Moreover, the SMDP framework has some inherent limitations. Explosion of the state space size seems unavoidable when we introduce more problem attributes or constraints. Choosing an appropriate solution method and adapting it to handle the constraints can be a daunting task. We have proposed some ideas on how such issues can be addressed.

### *12.3. A prototype system*

To evaluate the feasibility and effectiveness of the new paradigm, we have developed a prototype system that can handle a general class of dynamic decision problems. We have conducted some detailed case studies to evaluate the DynaMoL design and the DYNAMO implementation. The modeling experiences involved and the solution results produced gave us confidence that the methodology works well for a large class of dynamic decision problems in practical domains. Besides providing superior modeling support and a variety

of solution methods, the performance of the DYNAMO system is at least on par with existing programs. The system is being used to model various real-life decision problems in clinical and pharmaceutical decision analysis. Applications in other domains are also being planned.

We are interested in putting the system into practical use. Towards this end, we have documented the experiences of performing some detailed case studies in complex domains; these lessons have illuminated some relevant research issues and the required support tools.

## Acknowledgements

This paper reports work done partly at the MIT Laboratory for Computer Science, Cambridge, MA, USA. This research is supported in part by the National Institutes of Health Grant No. 5 R01 LM04493 from the National Library of Medicine, USA. It is also supported by a Strategic Research Grant No. RP960351 from the National Science and Technology Board and the Ministry of Education, Singapore. I would like to thank Drs. Charles E. Ellis and Stephen G. Pauker for guidance in conducting the case study in atrial fibrillation management. I would also like to thank Peter Szolovits, Alvin Drake, David Harmanec, Eric Horvits, Cungen Cao and Jianye Zheng for useful discussions and comments. Jianye Zheng has incorporated a functioning version of the tree views and their translators into the DYNAMO architecture; Mun Cheong Ng has analyzed and implemented several solution methods in the system; David Harmanec, Suman Sundaresh and Jianye Zheng have worked on the release version of DYNAMO for non commercial use.

## References

[1] A.G. Barto, S.J. Bradtke, S.P. Singh, Learning to act using real-time dynamic programming, Artificial Intelligence 72 (1–2) (1995) 81–138.

[2] J.R. Beck and S.G. Pauker, The Markov process in medical prognosis, Medical Decision Making 3 (1983) 419–458.

[3] R.A. Bellman, Dynamic Programming, Princeton University Press, Princeton, NJ, 1957.

[4] D.P. Bertsekas, Dynamic Programming: Deterministic and Stochastic Models, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[5] C. Berzuini, R. Bellazzi, S. Quaglini, Temporal reasoning with probabilities, in: Proceedings 5th Workshop on Uncertainty in Artificial Intelligence, Association for Uncertainty in Artificial Intelligence, 1989, pp. 14–21.

[6] C. Boutilier, R.I. Brafman, C. Geib, Prioritized goal decomposition of Markov decision processes: toward a synthesis of classical and decision theoretic planning, in: Proceedings 15th International Joint Conference on Artificial Intelligence (IJCAI-97), Nagoya, Japan, 1997.

[7] C. Boutilier, R. Dearden, M. Goldszmidt, Exploiting structure in policy construction, in: Proceedings 14th International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Quebec, 1995, pp. 1104–1111.

[8] J.S. Breese, Construction of belief and decision networks, Computational Intelligence 8 (1992) 624–647.

[9] C. Cao, T.Y. Leong, Learning conditional probabilities for influence views, in: Working Notes IJCAI Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-97), 1997, pp. 11–19.

[10] C. Cao, T.Y. Leong, A. Peng Kiong Leong, F. Choen Seow, Dynamic decision analysis in medicine: a data-driven approach, Internat. J. Medical Informatics, to appear.

[11] B.Y. Chan, R.D. Shachter, Structural controllability and observability in influence diagrams, in: D. Dubois, M.P. Wellman, B.D. D'Ambrosio, P. Smets (Eds.), Proceedings 8th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1992, pp. 25–32.

[12] D. Chapman, Planning for conjunctive goals, Artificial Intelligence 32 (1987) 333–377.

[13] K. Currie, A. Tate, O-Plan: the open planning architecture, Artificial Intelligence 51 (1) (1991) 49–86.

[14] P. Dagum, A. Galper, Forecasting sleep apnea with dynamic network models, in: D. Heckerman, A. Mamdami (Eds.), Proceedings 9th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1993, pp. 64–71.

[15] P. Dagum, A. Galper, E. Horvitz, Dynamic network models for forecasting, in: D. Dubois, M.P. Wellman, B.D. D'Ambrosio, P. Smets (Eds.), Proceedings 8th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1992, pp. 41–48.

[16] T. Dean, Decision-theoretic planning and Markov decision processes, Tutorial presented at the Summer Institute on Probability and Artificial Intelligence, Corvalis, OR, 1994.

[17] T. Dean, R. Givan, S. Leach, Model reduction techniques for computing approximately optimal solutions for Markov decision processes, in: Proceedings 13th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1997, pp. 124–131.

[18] T. Dean, L.P. Kaelbling, J. Kirman, A. Nicholson, Deliberation scheduling for time-critical sequential decision making, in: D. Heckerman, A. Mamdami (Eds.), Proceedings 9th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1993, pp. 309–316.

[19] T. Dean, L.P. Kaelbling, J. Kirman, A. Nicholson, Planning with deadlines in stochastic domains, in: Proceedings 11th National Conference on Artificial Intelligence (AAAI-93), Washington, DC, 1993, pp. 574–579.

[20] T. Dean, S. Kambhampati, Planning and scheduling, in: A.B. Tucker (Ed.), The Computer Science and Engineering Handbook, CRC Press, Rockville, MD, 1997, pp. 614–636.

[21] T. Dean, J.Kriman, K. Kanazawa, Probabilistic network representations of continuous-time stochastic processes for applications in planning and control, in: Proceedings 1st International Conference on AI Planning Systems, 1992.

[22] R. Dearden, C. Boutilier, Abstraction and approximate decision-theoretic planning, Artificial Intelligence 89 (1–2) (1997) 219–283.

[23] M. Drummond, J. Bresina, Ayntime synthetic projection: maximizing the probability of goal satisfaction, in: Proceedings 8th National Conference on Artificial Intelligence (AAAI-90), Boston, MA, 1990, 138–144.

[24] J.W. Egar, M.A. Musen, Graph-grammar assistance for automated generation of influence diagrams, in: D. Heckerman, A. Mamdami (Eds.), Proceedings 9th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1993, pp. 235–242.

[25] K. Erol, J. Hendler, D.S. Nau, HTN planning: complexity and expressitivity, in: Proceedings 12th National Conference on Artificial Intelligence (AAAI-94), Seattle, WA, 1994.

[26] R.E. Fikes, N.J. Nilsson, STRIPS: a new approach to the application of theorem proving to problem solving, Artificial Intelligence 2 (1971) 189–208.

[27] M.R. Genesereth, N.J. Nilsson, Logical Foundations of Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1987.

[28] R.P. Goldman, E. Charniak, Dynamic construction of belief networks, in: Proceedings 6th Conference on Uncertainty in Artificial Intelligence, 1990, pp. 90–97.

[29] V. Ha, P. Haddawy, Theoretical foundations of abstraction-based probabilistic planning, in: Proceedings 12th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1996, pp. 291–298.

[30] P. Haddawy, A.H. Doan, R. Goodwin, Efficient decision-theoretic planning: techniques and empirical analysis, in: P. Besnard, S. Hanks (Eds.), Proceedings 11th Conference on Uncertainty in Artificial Intelligence, 1995, pp. 229–236.

[31] M. Hauskrecht, N. Meuleau, C. Boutilier, L.P. Kaelbling, T. Dean, Hierarchical solution of markov decision processes using macro-actions, in: Proceedings 14th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1998.

[32] G.B. Hazen, Stochastic trees: a new technique for temporal medical decision modeling, Medical Decision Making 12 (1992) 163–178.

[33] J.P. Hollenberg, Markov cycle trees: a new representation for complex Markov processes, Medical Decision Making 4 (4) (1984). Abstract from the 6th Annual Meeting of the Society for Medical Decision Making.
[34] R.A. Howard, Dynamic Programming and Markov Processes, MIT Press, Cambridge, MA, 1960.
[35] R.A. Howard, Dynamic Probabilistic Systems, Vol. 1 & 2, Wiley, New York, 1971.
[36] R.A. Howard, Decision analysis: practice and promise, Management Science 34 (1988) 679–695.
[37] R.A. Howard, J.E. Matheson, Influence diagrams, in: R.A. Howard, J.E. Matheson (Eds.), The Principles and Applications of Decision Analysis, Vol. 2, Strategic Decisions Group, Menlo Park, CA, 1984, pp. 719–762.
[38] J. Janssen (Ed.), Semi-Markov Models: Theory and Applications, Plenum Press, New York, 1986.
[39] W.S. Jewell, Markov renewal programming: I. formulations, finite return models II. infinite return models example, Operations Research 11 (1963) 938–971.
[40] J.P. Kassirer, A.J. Moskowitz, J. Lau, S.G. Pauker, Decision analysis: a progress report, Annals of Internal Medicine 106 (1987) 275–291.
[41] C.A. Knoblock, Search reduction in hierarchical problem solving, in: Proceedings 9th National Conference on Artificial Intelligence (AAAI-91), Anaheim, CA, 1991, pp. 686–691.
[42] R.E. Korf, Planning as search: a quantitative approach, Artificial Intelligence 33 (1987) 65–88.
[43] N. Kushmerick, S. Hanks, D. Weld, An algorithm for probabilistic least-commitment planning, in: Proceedings 12th National Conference on Artificial Intelligence (AAAI-94), Seattle, WA, 1994, pp. 1073–1078.
[44] T.Y. Leong, Dynamic decision modeling in medicine: a critique of existing formalisms, in: Proceedings 17th Annual Symposium on Computer Applications in Medical Care, IEEE, November 1993, pp. 478–484.
[45] T.Y. Leong, An integrated approach to dynamic decision making under uncertainty, Technical Report TR 631, MIT Laboratory for Computer Science, Cambridge, MA, 1994.
[46] P. Magni, R. Bellazzi, DT-Planner: an environment for managing dynamic decision problems, Computer Methods and Programs in Biomedicine 54 (1997) 183–200.
[47] D. McAllester, D. Roseblitt, Systematic nonlinear planning, in: Proceedings 9th National Conference on Artificial Intelligence (AAAI-91), Anaheim, CA, 1991, pp. 634–639.
[48] D. McDermott, J. Hendler, Planning: What it is, What it could be, An introduction to the Special Issue on Planning and Scheduling, Artificial Intelligence 76 (1–2) (1995) 1–16.
[49] B.A. Myers, D.A. Giuse, B. Vander Zanden Dannenberg, D.S. Kosbie, E. Pervin, A. Mickish, P. Marchal, Garnet: comprehensive support for graphical, highly-interactive user interfaces, IEEE Computer 23 (11) (1990) 71–85.
[50] R.E. Neapolitan, Computing the confidence in a medical decision obtained from an influence diagram, Artificial Intelligence in Medicine 5 (1993) 341–363.
[51] A. Newell, J.C. Shaw, H.A. Simon, Report on a general problem-solving program, in: Proceedings International Conference on Information Processing, UNESCO, 1960, pp. 256–264.
[52] A. Newell, H.A. Simon, Human Problem Solving, Prentice-Hall, Englewood Cliffs, NJ, 1972.
[53] M.C. Ng, A solution suite for a dynamic decision modeling system, National University of Singapore, Department of Information Systems and Computer Science, Honours Year Project Report, 1997.
[54] S.G. Pauker, J.P. Kassirer, Medical progress: decision analysis, New England Journal of Medicine 316 (1987) 250–258.
[55] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo, CA, 1988.
[56] G.M. Provan, Modeling the evolution of acute abdominal pain using temporal influence diagrams, Technical Report MIS-CS-92–69, University of Pennsylvania, Department of Computer and Information Science, Philadelphia, PA, 1992.
[57] G.M. Provan, J.R. Clarke, Dynamic network construction and updating techniques for the diagnosis of acute abdominal pain, IEEE Trans. Pattern Analysis and Machine Intelligence 15 (3) (1993) 299–307.
[58] G.M. Provan, D. Poole, A utility-based analysis of consistency-based diagnosis, in: J. Allen, R. Fikes, E. Sandewall (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings 2nd International Conference (KR-91), Morgan Kaufmann, San Mateo, CA, 1991, pp. 461–472.
[59] M.L. Puterman, Markov decision processes, in: D.P. Heyman, M.J. Sobel (Eds.), Handbooks in Operations Research and Management Science, Vol. 2, Elsevier, North-Holland, Amsterdam, 1990, pp. 331–434.
[60] H. Raiffa, Decision Analysis: Introductory Lectures on Choices Under Uncertainty, Addison-Wesley, Reading, MA, 1968.

[61] E.D. Sacerdoti, Planning in a hierarchy of abstraction spaces, Artificial Intelligence 5 (1974) 115–135.
[62] R.D. Shachter, Evaluating influence diagrams, Operations Research 34 (1986) 871–882.
[63] R.D. Shachter, M.A. Peot, Decision making using probabilistic inference methods, in: D. Dubois, M.P. Wellman, B.D. D'Ambrosio, P. Smets (Eds.), Proceedings 8th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1992, pp. 276–283.
[64] L.S. Shapley, Stochastic games, Proceedings National Academy of Science 39 (1953) 1095–1100.
[65] S. Srinivas, Generalizing the noisy or model to $n$-ary variables, Technical Memorandum 79, Rockwell International Science Center, Palo Alto Laboratory, Palo Alto, CA, April 1992.
[66] M. Stefik, Planning with constraints (MOLGEN: Part 1), Artificial Intelligence 16 (1981) 111–139.
[67] J. Tash, S. Russell, Control strategies for a stochastic planner, in: Proceedings 12th National Conference on Artificial Intelligence (AAAI-94), Seattle, WA, 1994, pp. 1079–1085.
[68] A. Tate, Generating project networks, in: Proceedings 5th International Joint Conference on Artificial Intelligence (IJCAI-77), Cambridge, MA, 1977, pp. 888–893.
[69] A. Tate, J. Hendler, M. Drummond, A review of ai planning techniques, in: J. Allen, J. Hendler, A. Tate, (Eds.), Readings in Planning, Morgan Kaufmann, San Mateo, CA, 1990, pp. 26–49.
[70] J.A. Tatman, R.D. Shachter, Dynamic programming and influence diagrams, IEEE Trans. Systems Man Cybernet. 20 (2) (1990) 365–379.
[71] C. Wang, Knowledge-based formulation of dynamic decision models in medicine, M.Sc. Thesis, National University of Singapore, Department of Information Systems and Computer Science.
[72] M.P. Wellman, Formulation of Tradeoffs in Planning under Uncertainty, Pitman, London/Morgan Kaufmann, San Mateo, CA, 1990.
[73] M.P. Wellman, J.S. Breese, R.P. Goldman, From knowledge bases to decision models, The Knowledge Engineering Review 7 (1) (1992) 35–53.
[74] M.P. Wellman, J. Doyle, Modular utility representation for decision-theoretic planning, in: Proceedings 1st International Conference on AI Planning Systems, 1992.
[75] D.E. Wilkins, Practical Planning: Extending the Classical Planning Paradigm, Morgan Kaufmann, San Mateo, CA, 1988.
[76] K.E. Willard, G.C. Critchfield, Probabilistic analysis of decision trees using symbolic algebra, Medical Decision Making 6 (1986).
[77] S.S. Yeh, T.Y. Leong, Automatic generation of transition probabilities in dynamic decision modeling: a case study, in: Proceedings AAAI Spring Symposium on Artificial Intelligence in Medicine, 1994.
[78] J. Zheng, Consistency management in multiple-perspective dynamic decision modeling, M.Sc. Thesis, National University of Singapore, Department of Information Systems and Computer Science.