Extending Temporal Reasoning with Hierarchical Constraints

Fei Song
Dept. of Computing and Information Science
University of Guelph
Guelph, Ontario, Canada N1G 2W1

Abstract

Existing reasoning algorithms for Allen's interval algebra may produce weak results when applied to temporal networks that involve decompositions of intervals. We present a strengthened procedure for reasoning about such hierarchical constraints, which works interactively with an existing algorithm for temporal reasoning, to produce the desired stronger results. We further apply our algorithm to the process of plan recognition and show that such an application can both reduce the number of candidate plans and make the constraints in the remaining plans more specific.

1. Introduction

Allen's (1983a) interval algebra has shown to be useful for such applications as knowledge-based systems, natural language processing, and planning (as described in Vilain, et al., 1989). For example, a simplified plan for making a pasta dish can be represented as the temporal network in figure 1, where a node corresponds to the time interval over which a state holds or an event occurs, and a link label represents the temporal constraint between two intervals[1].

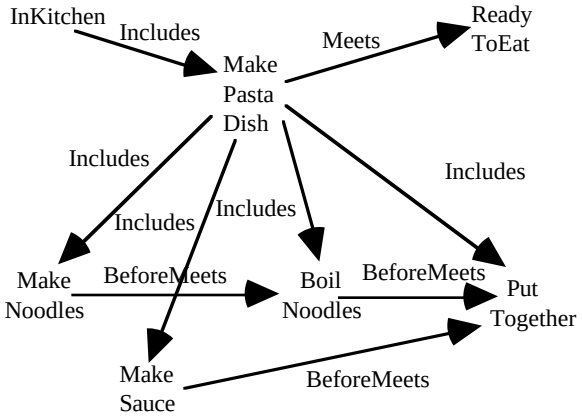

Figure 1: Temporal network of a cooking plan

[1] Here, "Includes" and "BeforeMeets" are high-level constraints, defined as the sets {si, di, fi, eq} and {b, m}, where b, m, eq are the basic relations "before", "meets", "equal", and si, di, fi are the inverses of "starts", "during", and "finishes" in Allen's interval algebra.

Given a temporal network, an important reasoning task is to compute the so-called minimal labeling, that is, to find the set of minimal constraints if the network is consistent (Vilain and Kautz, 1986). A constraint (or a link label) is minimal if each of its basic relations is part of a consistent singleton labeling, for which each link is labeled by a basic relation and all the links in the network are satisfied. Vilain and Kautz (1986) show that the time complexity of such a reasoning task is NP-complete for the interval algebra, where the problem size is the number of intervals. However, this does not prevent people from proposing polynomial algorithms that are approximate for the interval algebra (Allen 1983; Van Beek 1989). Allen's algorithm has $O(n^3)$ time complexity and is shown to be exact only for a subset of the interval algebra (Van Beek 1989). Van Beek then proposes an $O(n^4)$ algorithm which is exact for a larger subset (the subset of the interval algebra that can be translated into the point algebra). To get more exact results for the full interval algebra, one may have to use some exponential-time algorithms (e.g., Valdés-Pérez 1987).

One problem with these existing algorithms is that they may produce weak results when applied to temporal networks that involve hierarchical constraints (i.e. the decompositions of intervals into low-level subintervals). In Song and Cohen (1991), we proposed a strengthened algorithm for temporal reasoning about hierarchical constraints. The algorithm guarantees that the result is no weaker than that obtained from the existing temporal reasoning algorithms. However, whether it can derive the minimal labeling for the hierarchical constraints depends on the order in which lower-level constraints are combined. In this paper, we present a new order-independent reasoning procedure for hierarchical constraints, along with formal proofs for its associated properties. We further apply our

algorithm to the problem of plan recognition, and show that the observed temporal constraints can both reduce the number of candidate plans and make the constraints in the remaining candidate plans more specific.

2. Weak results from the existing algorithms

A hierarchical constraint corresponds to the decomposition of an interval to a set of subintervals. In terms of Allen's interval algebra, this means that the interval temporally includes all the subintervals. Suppose that initially there is no specific constraint between a1 and a2 in figure 2(a). Then, all we can decide is that A Includes a1 and A Includes a2, where Includes stands for the constraint {si, di, fi, eq}.

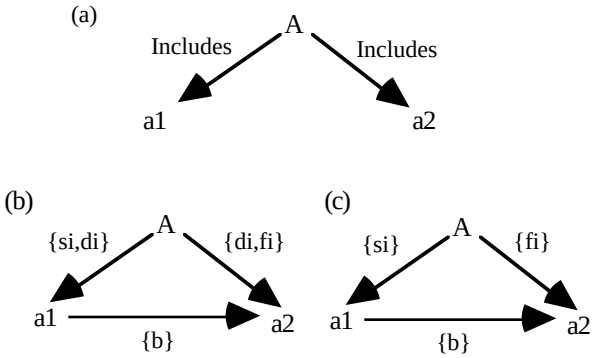

Figure 2: (a) One decomposition, (b) Weak results from Allen's algorithm, and (c) Strong results desired

Now, if we add a new constraint {b} between a1 and a2, then we can use Allen's algorithm to propagate the constraint and produce the results shown in figure 2(b). However, since a1 and a2 are the only subintervals of A and we know that a1 is located before a2, we should be able to decide that a1 is the starting part of A and a2 is the finishing part. In other words, we should get the desired results shown in figure 2(c).

Such weak results can be carried further for networks that consist of more than one decomposition. Suppose that initially we have the network shown in figure 3(a). Later, if we add the constraints a1 {b} a2 and a2 {b} a3, we get the results shown in figure 3(b) using Allen's algorithm, where "Common" stands for the constraint: {o, oi, s, si, d, di, f, fi, eq}. However, using a similar argument as made for the previous example, we should be able to get the stronger results shown in figure 3(c).

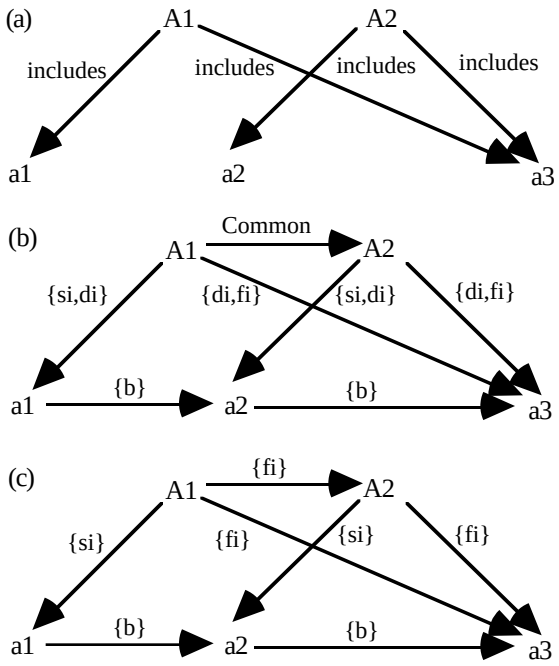

Figure 3: (a) Two decompositions, (b) Weak results from Allen's , and (c) Strong results desired

There are also networks that are considered to be consistent by Allen's algorithm but in fact are not when decompositions are involved. For example, the network in figure 4(a) is regarded as consistent by Allen's algorithm, since we get the same network after applying the algorithm. However, this is actually not true because if a1 and a2 are the only subintervals of A and a1 is located before a2, a2 should be the finishing part of A, not an interior part, as shown in figure 4(b).

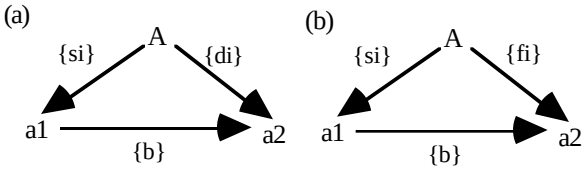

Figure 4: (a) Weak results from Allen's, and (b) Strong results desired

Such weak results are not simply caused by the inexactness of Allen's algorithm. In fact, Allen's algorithm is exact for all these examples since the constraints used fall into a subset of the interval algebra for which Allen's algorithm is guaranteed to find the set of minimal labels (Van Beek 1989). The reason for these weak results is that Allen's algorithm treats all the intervals as independent of each

other. This is certainly not true for decompositions, since the abstract intervals are temporally dependent on their subintervals. To make these dependencies explicit in the reasoning process, we need to assume that the decomposition of an abstract interval into its subintervals is complete, that is, no more subintervals can be added to the decomposition. As a result, we can compute how an abstract interval is temporally bounded by its subintervals based on the constraints between all the subintervals. For instance, if there is a linear ordering between all the subintervals, then we can clearly decide that the abstract interval is temporally bounded by the subintervals that occur the earliest and the latest. We say that a decomposition is closed if the constraints between the abstract interval and its subintervals are minimal with respect to the constraints between all the subintervals.

More formally, we describe an abstract interval as the convex hull or the minimal cover of its subintervals, denoted by the equation:

$$A = x_1 + x_2 + \cdots + x_n$$

where A denotes the abstract interval and $x_1, x_2, ..., x_n$ denote the subintervals. For the example in figure 3, the two decompositions can be represented as: $A_1 = a_1 + a_3$ and $A_2 = a_2 + a_3$. Closing a decomposition means closing every decomposition edge between the abstract interval and its subintervals, which further implies computing the minimal labels on the decomposition edges. We formally define a closure operation by the following first-order formula:

$$A(C_{i1}, C_{i2}, ..., C_{im})^c x_i \Leftrightarrow$$
$$x_i C_1 x_1 \wedge x_i C_2 x_2 \wedge \cdots \wedge x_i C_m x_m \wedge$$
$$(A = x_1 + x_2 + \cdots x_m)$$

where $C_{i1}, C_{i2}, ..., C_{im}$ are the constraints between the subinterval x_i and all the subintervals from x_1 to x_m. This formula suggests that a decomposition edge can be closed by using the existing constraints between a subinterval and all the subintervals and the fact that the abstract interval is the minimal cover of all its subintervals.

3. Development of a strengthened temporal reasoning algorithm

To describe our strengthened algorithm for temporal reasoning with hierarchical constraints, we start with the simple case of closing one and two subintervals, and then, we generalize the result to close more than two subintervals. After that, we provide a recursive procedure to close more than one decomposition, and finally, we present the strengthened algorithm that closes all the hierarchical structures in a temporal network.

3.1. Closing one and two subintervals

As described earlier, closing a decomposition means computing the minimal labels on all the decomposition edges. If an abstract interval has only one subinterval, then the minimal label on the decomposition edge is obviously {eq}, that is:

$$A(C_{ii})^c x_i \Leftrightarrow x_i C_{ii} x_i \wedge (A = x_i) \Leftrightarrow$$
$$x_i \{eq\} x_i \wedge (A = x_i) \Leftrightarrow A\{eq\} x_i$$

In the case of two subintervals, we can derive from the definition:

$$A(C_{ii}, C_{ij})^c x_i \Leftrightarrow x_i C_{ii} x_i \wedge x_i C_{ij} x_j \wedge (A = x_i + x_j)$$
$$\Leftrightarrow x_i \{eq\} x_i \wedge x_i C_{ij} x_j \wedge (A = x_i + x_j)$$

Since $x_i \{eq\} x_i$ always holds for an interval (the so-called node consistency), we define the basic closure operation on a constraint as:

$$A\, C_{ij}^c x_i \Leftrightarrow x_i C_{ij} x_j \wedge (A = x_i + x_j).$$

Lemma 1: Given C_{ij} as a set of basic relations $R_1, R_2, ..., R_m$, the basic closure C_{ij}^c can be computed as $\{R_1^c, R_2^c, ..., R_m^c\}$, where R^c is one of the four basic relations: si, di, fi, and eq, as defined in table 1.

Table 1: basic closure on basic relations

R	b	bi	m	mi	o	oi	s	si	d	di	f	fi	eq
R^c	si	fi	si	fi	si	fi	si	eq	di	eq	fi	eq	eq

The validity of table 1 can be easily verified. For example, if $x_i \{b\} x_j$, then the closed edge between A and x_i is $\{si\}$, since if A consists of only x_i and x_j and x_i is located before x_j, then x_i must be the starting part of A. This basic closure operation also applies to

the case of one subinterval, i.e., $(C_{ii})^c \Leftrightarrow C_{ii}^c$, since $AC_{ii}^c x_i \Leftrightarrow A\{eq\}^c x_i \Leftrightarrow A\{eq\}x_i$.

3.2. Closing more than two subintervals

Having defined the basic closure operation, we can now extend it to close a decomposition of more than two subintervals. More specifically, if $C_{i1}, C_{i2}, ..., C_{im}$ are the constraints between the subinterval x_i and all the subintervals from x_1 to x_m, including the subinterval x_i, then we can close x_i and another subinterval x_j using the basic closure operation: C_{ij}^c. To get the final closed decomposition edge to x_i, however, we need to somehow combine all of the C_{ij}^c's. It turns out that these C_{ij}^c's can be combined with the normal composition operation in Allen's interval algebra.

Lemma 2 Given the basic relations $R_{i1}, R_{i2}, ..., R_{in}$, we have

$$A(R_{i1}, R_{i2}, \cdots, R_{in})^c x_i \Leftrightarrow$$
$$A(R_{i1}^c \times R_{i2}^c \times \cdots \times R_{in}^c)x_i.$$

Proof. We prove this lemma by induction on the number of subintervals. For n = 1, we showed in the last subsection that $A(R_{ii})^c x_i \Leftrightarrow AR_{ii}^c x_i$. For n = 2, we have:

$$A(R_{ii}, R_{ij})^c x_i \Leftrightarrow AR_{ij}^c x_i$$
$$A(R_{ii}^c \times R_{ij}^c)x_i \Leftrightarrow A(\{eq\}^c \times R_{ij}^c)x_i \Leftrightarrow AR_{ij}^c x_i$$

So, the lemma holds for both n = 1 and n = 2.

Assume the lemma holds for n = k, that is, $A'(R_{i1}, R_{i2}, ..., R_{ik})^c x_i \Leftrightarrow A' R_{i1}^c \times R_{i2}^c \times \cdots \times R_{ik}^c x_i$, where $A' = x_1 + x_2 + \cdots + x_k$, we need to prove that the lemma also holds for n = k+1.

We know from table 1 that R_{ij}^c can only be one of the four basic relations: si, di, fi, and eq. By checking table 2, a sub-multiplication table drawn from Allen's (1983a), we see that these four basic relations are closed under multiplication.

Table 2: A Sub-Multiplication Table

×	si	di	fi	eq
si	si	di	di	si
di	di	di	di	di
fi	di	di	fi	fi
eq	si	di	fi	eq

It follows that $R_{i1}^c \times R_{i2}^c \times \cdots \times R_{ik}^c$ can only be one of the four basic relations: si, di, fi, and eq. Let us denote $R_{i1}^c \times R_{i2}^c \times \cdots \times R_{ik}^c$ as R, and R_{ik+1}^c as S.

Now, from the definition of the closure operation, we have:

$$\begin{aligned}
&A(R_{i1}, ..., R_{ik}, R_{ik+1})^c x_i \Leftrightarrow x_i R_{i1} x_1 \wedge \cdots \wedge x_i R_{ik} x_k \wedge \\
&\quad x_i R_{ik+1} x_{k+1} \wedge (A = x_1 + \cdots + x_k + x_{k+1}) \\
&\Leftrightarrow x_i R_{i1} x_1 \wedge \cdots \wedge x_i R_{ik} x_k \wedge (A' = x_1 + \cdots + x_k) \wedge \\
&\quad x_i R_{ik+1} x_{k+1} \wedge (A'' = x_i + x_{k+1}) \wedge (A = A' + A'') \\
&\Leftrightarrow A'(R_{i1}, ..., R_{ik})^c x_i \wedge A'' R_{ik+1}^c x_i \wedge (A = A' + A'') \\
&\Leftrightarrow A'(R_{i1}^c \times \cdots \times R_{ik}^c)x_i \wedge A'' R_{ik+1}^c x_i \wedge \\
&\quad (A = A' + A'') \\
&\Leftrightarrow A' Rx_i \wedge A'' Sx_i \wedge (A = A' + A'')
\end{aligned}$$

To further evaluate the above expression, we need to consider the following special cases:

(1) If $A'\{si\}x_i \wedge A''\{si\}x_i$, then we have $A\{si\}x_i$, since if x_i is the starting part of both A' and A'' and $A = A' + A''$, then x_i should also be the starting part of A.

(2) If $A'\{fi\}x_i \wedge A''\{fi\}x_i$, then we have $A\{fi\}x_i$. The reason is similar to case (1) above.

(3) If $A'\{si\}x_i \wedge A''\{fi\}x_i$, then we have $A\{di\}x_i$. The reason for this is that if x_i is the starting part of A', then there must be another interval that finishes after x_i. Similarly, if x_i is the finishing part of A'', then there must be another interval that starts before x_i. Thus, there are intervals that starts before x_i and finishes after x_i, and x_i must be an interior part of the covering interval A.

(4) If $A'\{di\}x_i \wedge A''Sx_i$, then we have

$A\{di\}x_i$, since if x_i is an interior part of A', it is also an interior part of A.

(5) If $A'\{eq\}x_i \wedge A''Sx_i$, then we have ASx_i. This is obviously true since A' equals x_i.

Since conjunctions are commutative, it is easy to see that these results are exactly the same as table 2 above. In other words, we have proved that:

$$A(R_{i1},...,R_{ik},R_{ik+1})^c x_i \Leftrightarrow A(R \times S)x_i \Leftrightarrow A(R_{i1}^c \times \cdots \times R_{ik}^c \times R_{ik+1}^c)x_i.$$

that is, lemma 2 also holds for n = k+1.

Lemma 2 implies that if the constraints between one subinterval and all the subintervals are one of the basic relations, the decomposition edge to the subinterval can be closed by multiplying the basic closures of these constraints.

Theorem 1: Given $C_{i1},C_{i2},...,C_{im}$ as the constraints between x_i and all the subintervals from x_1 to x_m, the closed edge between A and x_i can be computed as follows:

$$A(C_{i1},C_{i2},...,C_{im})^c x_i \Leftrightarrow AC_{i1}^c \circ C_{i2}^c \circ \cdots \circ C_{im}^c x_i.$$

Proof. The theorem can be proved by expanding constraints into sets of basic relations, converting the result into disjunctions of conjunctions, and applying lemma 2 to all the conjunctions:

$$\begin{aligned}
&A(C_{i1},C_{i2},...,C_{im})^c x_i \Leftrightarrow x_i C_{i1} x_1 \wedge x_i C_{i2} x_2 \wedge \cdots \wedge \\
&\quad x_i C_{im} x_m \wedge (A = x_1 + x_2 + \cdots + x_m) \\
&\Leftrightarrow x_i\{R_{11},R_{12},...,R_{1n_1}\}x_1 \wedge \\
&\quad x_i\{R_{21},R_{22},...,R_{2n_2}\}x_2 \wedge \\
&\quad \cdots\cdots \\
&\quad x_i\{R_{m1},R_{2m2},...,R_{mn_m}\}x_m \wedge \\
&\quad (A = x_1 + x_2 + \cdots + x_m) \\
&\Leftrightarrow (x_i R_{11} x_1 \wedge x_i R_{21} x_2 \wedge \cdots \wedge x_i R_{m1} x_m \wedge \\
&\quad A = x_1 + x_2 + \cdots + x_m) \vee \\
&\quad (x_i R_{11} x_1 \wedge x_i R_{21} x_2 \wedge \cdots \wedge x_i R_{m2} x_m \wedge \\
&\quad A = x_1 + x_2 + \cdots + x_m) \vee \\
&\quad \cdots\cdots \\
&\quad (x_i R_{1n_1} x_1 \wedge x_i R_{2n_2} x_2 \wedge \cdots \wedge x_i R_{mn_m} x_m \wedge \\
&\quad A = x_1 + x_2 + \cdots + x_m) \\
&\Leftrightarrow A(R_{11},R_{21},...,R_{m1})^c x_i \vee \\
&\quad A(R_{11},R_{21},...,R_{m2})^c x_i \vee \\
&\quad \cdots\cdots \\
&\quad A(R_{1n_1},R_{2n_2},...,R_{mn_m})^c x_i \\
&\Leftrightarrow AR_{11}^c \times R_{21}^c \times \cdots \times R_{m1}^c x_i \vee \\
&\quad AR_{11}^c \times R_{21}^c \times \cdots \times R_{m2}^c x_i \vee \\
&\quad \cdots\cdots \\
&\quad AR_{1n_1}^c \times R_{2n_2}^c \times \cdots \times R_{mn_m}^c x_i \\
&\Leftrightarrow AC_{i1}^c \circ C_{i2}^c \circ \cdots \circ C_{im}^c x_i.
\end{aligned}$$

Lemma 3 Given constraints as subsets of $\{si, di, fi, eq\}$, the composition is commutative and associative, that is, $C_1 \circ C_2 = C_2 \circ C_1$, and $(C_1 \circ C_2) \circ C_3 = C_1 \circ (C_2 \circ C_3)$.

Proof. Given two subsets of $\{si, di, fi, eq\}$, the composition is both commutative and associative since for each pair of the basic relations, the results of multiplications are symmetric, as shown previously in table 2.

Theorem 2: In closing a decomposition constraint using theorem 1, we get the same result no matter what order we do the compositions.

Proof. This follows directly from lemma 3, since the composition is both commutative and associative for subsets of $\{si, di, fi, eq\}$.

Based on theorems 1 and 2, we now present a new procedure for closing a decomposition of any number of subintervals.

```
procedure CLOSE(k, S)
begin
  for each i ∈ S do begin
    t ← {eq}
    for each j ∈ S do
      t ← t ∘ C_ij^c
    t ← t ∩ C_ki
    if t ≠ C_ki then begin
      C_ki ← t
      C_ik ← INVERSE(t)
      Q ← Q ∪ RELATED_PATHS(k, i)
    end
  end
end
```

Figure 5: Procedure for closing a decomposition

The above procedure closes all the decomposition edges in turn, and if a closed edge is more specific than the existing edge, the existing edge will be updated and all the related paths will be queued for further propagation.

Theorem 3. The time complexity of the CLOSE procedure is $O(m^2)$ where m is the number of subintervals in a decomposition.

3.3. Closing all the decompositions in a hierarchical structure

A hierarchical structure often consists of more than one decomposition. Our strategy is to close a hierarchy in a post-order fashion, since higher-level intervals can be defined in terms of lower-level subintervals. In other words, we start the closing process from the bottom-level decompositions and work our way up until all the decompositions are closed in the hierarchy.

```
procedure CLOSE_ALL (k)
begin
    get a list S of subintervals for k
    if S is not empty then begin
        for each i ∈ S do
            CLOSE_ALL (i)
        CLOSE (k, S)
    end
end
```

Figure 6: Closing all the decompositions in a plan

Theorem 4. The time complexity of the CLOSE_ALL procedure is bounded below by O(n) and above by $O(n^2)$, where n is the number of intervals in a plan.

3.4. The Strengthened Algorithm

The CLOSE_ALL procedure closes all the decompositions in a hierarchical structure. To get stronger results for a temporal network, we first use an existing reasoning algorithm to compute the set of constraints to be as specific as possible. Then, for each hierarchical structure in the temporal network, we recursively close all the decompositions using the CLOSE_ALL procedure. After that, some of the decomposition edges may be updated, and we call the temporal reasoning algorithm again to propagate the effects of these new constraints. Thus, we generally need to call interactively an existing reasoning algorithm and our CLOSE_ALL procedure. Such a process will eventually terminate since every time we update a constraint, some of its basic relations will be eliminated and there are at most 13 basic relations in any constraint.

We now give the strengthened algorithm for temporal reasoning with hierarchical constraints:

```
algorithm STRENGTHENED
begin
  Q←{initial paths in a temporal network}
  H←{roots of all hierarchical structures}
  while Q is not empty do begin
      MODIFIED_TR
      foreach k ∈ H do
          CLOSE_ALL (k)
  end
end
```

Figure 7: The strengthened algorithm for temporal reasoning about plans

The set H contains the roots of all hierarchical structures, and CLOSE_ALL closes all the decompositions in a hierarchy. The set Q contains those paths whose effects need to be propagated, and MODIFIED_TR is the same as an existing algorithm for temporal reasoning except that the initialization of Q is removed from the procedure.

Theorem 5. The time complexity of our strengthened algorithm is at most $O(T \log_2 n)$, where n is the number of intervals in a temporal network and T is the time complexity of an existing reasoning algorithm (n^3 for the path-consistency procedure, n^4 for Van Beek's procedure, and exponential for some more exact procedures).

Proof: First, each iteration of our algorithm takes $O(T+n^2)$ time or O(T), since all the existing algorithms take time at least $O(n^3)$. Second, the worst case corresponds to a balanced binary tree, with maximum levels of decompositions and for which the effects of closed decompositions need to be propagated upwards. Thus, we have the factor of $\log_2 n$ for the number of iterations.

4. An application to plan recognition

Plan recognition is the process of inferring an agent's plan based on the observation of the agent's actions. A recognized plan is useful in that it helps to decide an agent's goal and predict the agent's next action. For example, if we observe that John has made the sauce and he is now boiling the noodles, then based on the plan shown in figure 1, we can decide that John's goal is to make a pasta dish and his next action is to put noodles and sauce together. Plan recognition has found applications in such areas as story understanding, psychological modeling, natural language pragmatics, and intelligent interfaces.

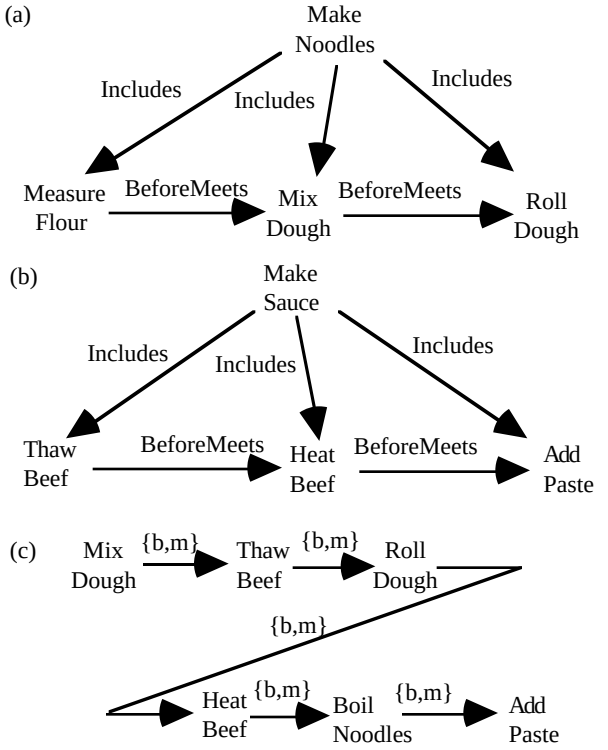

Figure 8: An extended plan for making a pasta dish

Most existing models for plan recognition assume a library of all possible plans that might occur in a particular domain. Then, through some kind of search and matching mechanism, one can find all the plans that contain the observed actions, called candidate plans. Since the observation of an agent's actions is often incomplete and some actions may appear in many different plans of the plan library, it is often difficult to determine the unique plan that an agent is pursuing. Kautz (1987) suggests that one way of reducing the number of candidate plans is to use various kinds of constraints, including the temporal relations explicitly reported in the observations, to further eliminate those inconsistent plans[2]. However, Kautz only adopted a subset of Allen's interval algebra and did not use fully the temporal constraints that correspond to the decomposition edges in a candidate plan.

Our approach to plan recognition is to represent a plan as a temporal network and perform temporal reasoning to eliminate those candidate plans that are inconsistent with the temporal constraints explicitly given in the observations. Such a reasoning process can have two useful effects: the given constraints can be used to reduce the number of candidate plans (an example of this effect can be found in Song and Cohen (1991)) and the given constraints can be made more specific by combining them with the prestored constraints in a candidate plan.

To illustrate the second effect, we extend the plan for making a pasta dish in section 1 by adding the decompositions for MakeNoodles and MakeSauce, shown in figure 8(a) and (b). Suppose that the observation of an agent's actions is given in figure 8(c). We can then use Allen's algorithm or our strengthened algorithm to make some of the constraints in the plan more specific. Using Allen's algorithm, we can make some of the constraints more specific, as shown in figure 9(a). Using our strengthened algorithm, we can make these constraints even more specific, as shown in figure 9(b).

MakePastaDish {si, di} MakeNoodles
MakePastaDish {si, di} MakeSauce
MakePastaDish {di, fi} PutTogether

MakeNoodles {si, di} MeasureFlour
MakeNoodles {di, fi} RollDough

MakeSauce {si, di} ThawBeef
MakeSauce {di, fi} AddTomatoPaste

MakeNoodles {o, s, d} MakeSauce
MakeNoodles {b, m} BoilNoodles

Figure 9(a): Results from Allen's Algorithms

[2] Other solutions include the use of preference heuristics (Allen, 1983b; Litman, 1985; Carberry, 1986) and probabilities (Goldman and Charniak, 1988).

MakePastaDish {si} MakeNoodles
MakePastaDish {di} MakeSauce
MakePastaDish {fi} PutTogether

MakeNoodles {si} MeasureFlour
MakeNoodles {fi} RollDough

MakeSauce {si} ThawBeef
MakeSauce {fi} AddTomatoPaste

MakeNoodles {o} MakeSauce
MakeNoodles {b} BoilNoodles

Figure 9(b): Results from our strengthened algorithm

5. Conclusion

We presented a strengthened algorithm for temporal reasoning about plans, which improves on straightforward applications of the existing reasoning algorithms for Allen's interval algebra. We view plans as both temporal networks and hierarchical structures. Such a dual view allows us to design a closing procedure which makes as specific as possible the temporal constraints between abstract actions and their subactions. The procedure is then used interactively with an existing reasoning algorithm to help obtain the strengthened results. We applied our algorithm to the problem of plan recognition and showed that such an application can both reduce the number of candidate plans make the constraints in the remaining plans more specific.

One possible area for future work is to improve the efficiency of our algorithm, which calls interactively an existing reasoning algorithm and our closing procedure. Although the strengthened algorithm only adds a factor of $\log_2 n$ to the time complexity of an existing reasoning algorithm, it is worth investigating whether such interactions can be localized and reduced. Some results on localizing the propagation of temporal constraints in Allen's interval algebra have been reported (Koomen 1989). This would form a useful starting point for our future research.

Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.

ALLEN, J. F. 1983a. Maintaining knowledge about temporal intervals. Communications of the ACM, **26**: 832-843.

_____1983b. Recognizing intentions from natural language utterances. *In* Computational models of discourse. *Edited by* M. Brady and R. Berwick. The MIT press, Cambridge, Mass. pp. 107-166.

ALLEN, J. F., and KOOMEN, J. A. 1983. Planning using a temporal world model. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, pp. 741-747.

CARBERRY, S. 1986. Pragmatic modeling in information system interfaces. Ph.D. Dissertation, University of Deleware.

GOLDMAN, R., and CHARNIAK, E. 1988. A probabilistic ATMS for plan recognition. Proceedings of the AAAI Workshop on Plan Recognition.

KAUTZ, H. A. 1987. A formal theory of plan recognition. Ph.D. Dissertation, University of Rochester, Rochester, N.Y.

KOOMEN, J. A. 1989. Localizing temporal constraint propagation. Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning, Toronto, Ontario, Canada, pp. 198-202.

LITMAN, D. 1985. Plan recognition and discourse analysis: an integrated approach for understanding dialogues. Ph.D. Dissertation, University of Rochester.

SONG, F. 1991. A processing model for temporal analysis and its application to plan recognition. Ph.D. Dissertation, University of Waterloo, Waterloo, Ontario, Canada.

SONG, F., and COHEN, R. 1991. Temporal reasoning during plan recognition. Proceedings of the Ninth National Conference on Artificial Intelligence, Anaheim, CA, pp. 247-252.

VALDÉS-PÉREZ, R. E. 1987. The satisfiability of temporal constraint networks. Proceedings of the Sixth National Conference on Artificial Intelligence, pp. 256-260.

VAN BEEK, P. 1989. Approximation algorithms for temporal reasoning. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pp. 1291-1296.

VILAIN, M., and KAUTZ, H. 1986. Constraint propagation algorithms for temporal reasoning. Proceedings of the Fifth National Conference on Artificial Intelligence, pp. 377-382.

VILAIN, M., KAUTZ, H., and VAN BEEK, P. 1989. Constraint propagation algorithms for temporal reasoning: a revised report. *In* Readings in qualitative reasoning about physical systems. *Edited by* D.S. Weld and J. de Kleer. Morgan Kaufman, San Mateo, CA, pp. 373-381.