Reasoning about Events with Time-interval Information

Shieu-Hong Lin
Department of Mathematics and Computer Science
Biola University
13800 Biola Avenue
La Mirada, California 90639
shieu-hong.lin@bubbs.biola.edu

Abstract

It is a challenging task to reason about a dynamically evolving system with events triggering state transitions and uncertainty about the ordering of events. In previous research, uncertainty of the event ordering is mainly modelled by partial orders over the events. In this paper, we investigate uncertainty of the event ordering modelled by time intervals in which the events may occur. Each event will occur at exactly one point of time, but it could be any point within the time interval associated with the event. We present both positive and negative results that provide insight into the complexity of temporal reasoning about events with time-interval information.

1 Introduction

Reasoning about dynamically evolving systems with events triggering state transitions has important applications in plan validation (Nebel & Bäckström 1992) (Dean & Boddy 1988) and system verification (Dean & Wellman 1991) (Manna & Pnueli 1995) (Lin & Dean 1996). The dynamics of the system is often modelled by a set of boolean state variables, each of which has a value, $true$ or $false$, at a given point of time. The values of the state variables at a point of time determine the system state at that time. The evolution of the system depends on a sequence of state transitions triggered by a set of coming events. Often there is uncertainty about in what order the events will occur as a sequence. In general, there may be many possible event sequences that end in very different trajectories of system evolution. In previous research (Dean & Boddy 1988) (Nebel & Bäckström 1992) (Lin & Dean 1996), uncertainty of the event ordering is mainly modelled by partial orders over the events. In this paper, we investigate uncertainty of the event ordering modelled by time intervals in which the events may occur. Each event will occur at exactly one point of time, but it could be any point within the time interval associated with the event.

When an event occurs, it instantly changes the system state according to a set of causal rules associated with the event. A causal rule is a STRIPS-like operator as used in the planning community (Russell & Norvig 2003) (Ghallab, Nau, & Traverso 2004). A causal rule is applicable in a state if the values of the state variables satisfy the antecedent requirements of the rule. When the rule is applied in an applicable state, the values of the state variables are updated as described in the rule. Events could be deterministic or nondeterministic. An event is a nondeterministic event if there are two or more associated causal rules applicable in a state. In that situation, exactly one of the applicable rules is nondeterministically picked and applied to precipitate a state transition. In general, reasoning about nondeterministic events is more challenging than about deterministic events due to such nondeterminism.

In this paper, we focus on the temporal projection problem (Dean & Boddy 1988) (Nebel & Bäckström 1992) (Lin & Dean 1996) over events in dynamical systems as described above. Given an initial state and a goal, the task of temporal projection is to determine the existence of a possible event sequence that ends in a system state that satisfies the goal, and provide information of such an event sequence if one exists. Temporal projection can be considered as a special kind of planning (Lin & Dean 1996) (Ghallab, Nau, & Traverso 2004) since events are associated with operators (causal rules) and we are looking for an event sequence to achieve the goal. However, for temporal projection, we do not have full control over the events. Instead, we have to achieve the goal under the restrictions that the number of events is fixed, that the event ordering must be consistent with the time interval information, and that a fixed number of operators (causal rules) are associated with each event.

Temporal projection is a challenging computational task. In addition to the number of events involved, there are several factors affecting the complexity of temporal projection, such as the number state variables of the system, whether the events are deterministic or nondeterministic, and how tightly the associated time intervals overlap with one another. In this research, we develop an algorithm that transform the temporal projection problem to an AI search problem and the algorithm allows us to exploit structure embedded within the time-interval information to improve performance. We

also present both positive and negative results that provide insight into the complexity of temporal projection involving events with time-interval information.

The remainder of the paper is organized as follows. Section 2 provides definitions of terms and notations to formally describe the temporal projection problem involving events with time-interval information. Section 3 describes the algorithm to exploit the event-chain structure within the time-interval information, which transform the temporal projection problem to an AI search problem. In section 4, we investigate the complexity trade-offs involving various factors mentioned above.

2 Temporal Projection with Time-interval Information

In the following, we introduce definitions of terms and notations to formally describe the temporal projection problem with time-interval information associated with the events. These terms and notations are also used in the results and the proofs appearing in the later sections. Using the notations introduced in this section, Figure 1 provides a concrete problem instance of six events together with their causal rules and time-interval information.

State variables and states: We model the dynamics of the system in terms of a set of m boolean state variables $\mathbf{X} = \{X_1, X_2, \ldots, X_m\}$. The value of a state variable X_i is either $true$ or $false$ at a point of time. The values of the state variables in $\mathbf{X}$ all together at a point of time determine the system state at that time. We represent a state as $(x_1, x_2, \ldots, x_m)$ where x_i is the value of X_i in the state u. For notational convenience, we use the notation $S_{\mathbf{X}}$ to refer to the state space $\{true, false\}^m$.

State expressions: A state expression is a set of pairs of the form $\langle X_i, value\rangle$ where X_i is a state variable and $value$ is either $true$ or $false$. A state expression α is true in a state u if and only if for each pair $\langle X_i, true\rangle$ in α, the value of X_i in u is $true$ and for each pair $\langle X_j, false\rangle$ in α, the value of X_j in u is $false$. An empty state expression is true in every state.

Causal rules: A causal rule r is represented as $\alpha \rightarrow \beta$ where α and β are state expressions. α is the antecedent requirement of rule r and β is the causal effect of rule r. A rule r of the form $\alpha \rightarrow \beta$ is applicable in state u if and only if α is true in state u. Rule r can cause a state transition from state u to state v if rule r is applicable in state u, β is true in state v, and the values of the state variables not appearing in β remain the same in v as they are in u.

Remark: A rule $r = \alpha \rightarrow \beta$ is actually a STRIPS-like operator where the state variables appearing in α are the preconditions of the operator while those appearing in β are the postconditions of the operator.

Events and state transitions: When an event occurs, it instantly triggers a state transition at a point of time. No two events can occur at the same point of time. Each event is associated with a set of causal rules. Given the state u immediately before an event e occurs, event e can trigger a state transition from state u to another state v according to one of the causal rules applicable in u. If two or more rules of event e are applicable in state u, event e triggers state transition nondeterministically according to exactly one of the applicable rules. If no rule explicitly associated with an event is applicable to a state, nothing is changed by the event and the system remains in the same state. In other words, there is an implicit causal rule that will cause a state transition back to the same state when none of the explicit rules is applicable.

Remark: If an event e cause a state transition from state u to state v due to a causal rule r, we say that the rule r is applied by the event e in state u. An event precipitates change of system state. The effects of an event is conditional on the state immediately before the event occurs, and the set of causal rules associated with the event. In this way, an event determines a state-transition relation over $S_{\mathbf{X}}$.

Nondeterministic events: An event e is a nondeterministic event if there exists a state in which two or more explicit causal rules associated with e are applicable; otherwise the event is a deterministic event. A deterministic event encodes a complete state-transition function over the state space determined by the set of state variables while a nondeterministic event encodes a complete state-transition relation over the state space.

Time-interval information of events: The time line is modelled as the line of real numbers with every point of time uniquely encoded as a real number. Each event e is associated with an open time interval $I(e) = (t, t')$ where $t < t'$. For notational convenience, we refer to the two ends (t and t') of this open interval as $AT(e)$ and $BT(e)$ respectively. Event e will occur at exactly one point of time, and it could be any point of time after $AT(e)$ and before $BT(e)$.

Event sequences, tails, and prefixes: An event sequence q of length h over a set of events E, where $h \leq |E|$, is a sequence $\langle e_1, e_2, \ldots, e_h\rangle$ where $e_i \in E$ for $1 \leq i \leq h$, and $e_i \neq e_j$ if $i \neq j$. The event e_h is the tail of q. A prefix of an event sequence $q = \langle e_1, e_2, \ldots, e_h\rangle$ is an event sequence $q' = \langle e_1, e_2, \ldots, e_{h'}\rangle$ where $h' \leq h$.

Trajectory: A state sequence $\langle s_0, s_1, \ldots, s_h\rangle$ is a trajectory of an event sequence $q = \langle e_1, e_2, \ldots, e_h\rangle$ if and only if each event e_i, can trigger a state transition from state s_{i-l} to state s_i, for $1 \leq i \leq h$. A trajectory of an event sequence is simply a possible system evolution in the state space triggered by the event sequence.

Remark. When a set of events occur as a sequence, it causes a sequence of state transitions. This corresponds to the evolution of system, which critically depends on the ordering of

$\mathbf{X} = \{X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8\}$
$s_0 = (true, true, true, true, true, true, true, true)$
$G = (\ \langle X_1, false\rangle\langle X_2, false\rangle\langle X_4, false\rangle$
$\langle X_5, false\rangle\langle X_7, false\rangle\langle X_8, false\rangle\ )$
$E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$

$e_1 : I(e_1) = (21, 28)$
$\{(\langle X_1, true\rangle\langle X_2, true\rangle) \rightarrow (\langle X_1, false\rangle\langle X_2, false\rangle\langle X_3, true\rangle)$
$(\langle X_1, true\rangle\langle X_3, true\rangle) \rightarrow (\langle X_1, false\rangle\langle X_3, false\rangle\langle X_4, true\rangle)$
$\}$
$e_2 : I(e_2) = (25, 32)$
$\{(\langle X_1, true\rangle\langle X_3, true\rangle) \rightarrow (\langle X_1, false\rangle\langle X_3, false\rangle\langle X_2, true\rangle)$
$(\langle X_1, true\rangle\langle X_2, true\rangle) \rightarrow (\langle X_1, false\rangle\langle X_3, false\rangle\langle X_4, true\rangle)$
$\}$
$e_3 : I(e_3) = (3, 9)$
$\{(\langle X_4, true\rangle\langle e, true\rangle) \rightarrow (\langle X_4, false\rangle\langle X_5, false\rangle\langle X_6, true\rangle)$
$\}$
$e_4 : I(e_4) = (1, 12)$
$\{(\langle X_4, true\rangle\langle X_6, true\rangle) \rightarrow (\langle X_4, false\rangle\langle X_6, falae\rangle\langle e, true\rangle)$
$\}$
$e_5 : I(e_5) = (16, 23)$
$\{(\langle X_1, false\rangle\langle X_4, true\rangle) \rightarrow (\langle X_1, true\rangle\langle X_4, false\rangle\langle X_7, false\rangle)$
$\}$
$e_6 : I(e_6) = (13, 18)$
$\{(\langle X_1, true\rangle\langle X_4, false\rangle) \rightarrow (\langle X_1, false\rangle\langle X_4, true\rangle\langle X_8, false\rangle)$
$\}$

Figure 1: A problem instance $(\mathbf{X}, s_0, G, E)$ with six events

the events and the choices of applicable rules for individual events.

Sets of event sequences: Given a set of events E together with their time-interval information, Q^E is the set of all event sequences of length $|E|$ over E that are consistent with the time-interval information.

Temporal projection problem instances: An instance of the temporal projection problem is a four-tuple $(\mathbf{X}, s_0, G, E)$, where $\mathbf{X}$ is a set of state variables that determine the states of the system, s_0 is the initial state, G is a state expression to implicitly specify a set of goal states in which G is true, and E is a set of events together with the causal rules and the time-interval information associated with the events.

Temporal Projection: Given an instance of the temporal projection problem, the PRJ task of temporal projection is to determine the existence of event sequences in Q^E immediately following which the system is in one of the goal states. If such event sequences do exist, we would like to generate one such sequence together with (1) the causal rules applied by the individual events and (2) a suggested list of specific points of time in which the events may occur.

Example: In Figure 1, we have a set X of eight state variables, each of which models whether the concentration of certain chemical is low or not in a production process. Initially, all chemicals are low in concentration, and thus in the initial state s_0 all the variables have the same value, $true$. Six events in the production process will change the values of state variables according to the causal rules. Each event will occur at a point within the specified time interval. The task is to determine, as specified in the goal G whether it is possible that in the end of the production process the concentrations of the first, the second, the fourth, the fifth, the seventh, and the eighth of the chemicals are all high, and thus would damage the quality of the final product. In this particular example, the event sequence $\langle e_3, e_4, e_6, e_5, e_1, e_2\rangle$ with the first causal rules of these events applied will end in a state satisfying the goal. Such an event sequence may come with events occurring in that order at the specific points of time 5, 10, 15, 20, 25, and 30 respectively.

3 Temporal Projection as Search

In the following, we describe an algorithm to exploit the event-chain structure within the time-interval information and reduce the temporal projection problem to an AI search problem. First of all, we need to define (1) the degree of uncertainty over the ordering of events and (2) the event-chain structure as follows.

Definition 1 (Degree of uncertainty) *Given a set of events, we say the set of events has a degree d of uncertainty over the ordering of events if and only if (1) there exists one point of time t such that t falls inside the associated time intervals of $d + 1$ events and (2) at any other point of time t', t' falls inside the associated time intervals of no more than $d + 1$ events. In other words, $d + 1$ is the maximal number of events within this set that can occur in any order relative to one another. When d is zero, there is no overlapping between the time intervals and thus no uncertainty over the ordering of events. When d is zero, events in the set can only occur in a fixed order.*

Definition 2 (Event chains) *A set of events forms an event chain if and only if their associated time intervals do not overlap. Such a set of events has 0 degree of uncertainty over the ordering of events, and they can only occur in a fixed order relative to one another. Conceptually, we can view these events as a chain one following another through the time line.*

The following procedure can determine d, the degree of uncertainty over the ordering of events and partition the set of events into $d + 1$ event chains.

Procedure DecompositionIntoChains
Input: an problem instance $(\mathbf{X}, s_0, G, E)$
Output: (1) d: the degree of uncertainty over the ordering of events and (2) a partition of events in E into $d + 1$ event chains
1. Set up a priority queue to maintain the information of

available chains and the time they are created. Initially there is no event chain yet and none of the events is assigned to any chain. Sort the collection of all the $AT(e)$ and $BT(e)$ end points of the time intervals associated with every event e in E. [1]

2. Repeatedly scan the end points from the earliest one to the latest one along the time line. For each end point, do either step 3 or step 4 accordingly.

3. For an $AT(e)$ end point, if there is no available chain at the moment, create a new chain, assign e as the first event in the chain, and mark this chain as unavailable. Otherwise, remove among the available chains in the priority queue the one that was created earlier than any one else, assign e as the next event in this available chain, and mark that chain as unavailable.

4. For an $BT(e)$ end point, mark the chain that event e is assigned to as an available chain again and put it back to the priority queue.

5. Output the event chains according to the assignment of events to chains and report the degree of uncertainty d as the total number of chains allocated minus one.

Lemma 1 *Given a set of n events with the time-interval information, the procedure **DecompositionIntoChains** can determine the degree of uncertainty over the ordering of events d in $O(n \log n)$ time and partition the set of events into $d+1$ event chains. Any possible event sequence over the n events is derived from interleaving these $d+1$ event chains into a sequence.*

Proof sketch. It takes $O(n \log n)$ time to sort the end points, and then spends $O(n)$ time to process the end points. This procedure basically scan the time line from left to right to assign every pair of events with overlapping intervals into different chains, and it does so by using the minimum number of chains needed. This minimum number of chains needed to separate the intervals also tells us the maximum number of the intervals that any point of time t can fall into.

When the degree of uncertainty over the event ordering is a constant, we can view the set of events as several event chains and reason about the interleaving of these event chains into a possible event sequence. In the following, we reduce the task of temporal projection to graph search by applying the standard graph-search algorithms to derive the answers. It turns out the event-chain structure can be exploited in the procedure to guarantee polynomial-time computational efficiency when dealing with a state space of polynomial size .

Procedure TemporalSearch
Input: an problem instance $(\mathbf{X}, s_0, G, E)$.
Output: (1) determine the existence of a goal state s_G that the system may end in after some event sequence in Q^E and s_G satisfies the goal G; (2) an event sequence q in Q^E that ends in the goal state s_G if one exists; (3) the rules applied by individual events in q to enter the goal state; and (4) a suggested list of specific points of time in which the events may occur

1. Call procedure **DecompositionIntoChains** to determine the degree of uncertainty d and partition the set of events E into d event chains as stated in Lemma 1.

2. Conduct a graph search as described in step 5 through the following directed acyclic graph $T^E = (V, A)$ as described in step 3 and step 4 to find the information we need. [2]

3. The set of vertices V of the graph $T^E = (V, A)$ models all the possibilities of the system state after some numbers of events in the individual event chains have already occurred. A vertex v in V is of the form $(s, c_1, c_2, \ldots, c_{d+1})$ where $s \in S_{\mathbf{X}}$ indicates the corresponding system state in vertex v while c_i $(1 \leq i \leq d+1)$ is an event counter for the ith event chain indicating that the first c_i events in the ith chain have occurred while the remaining events in the ith chain have not occurred yet.[3]

4. The set of arcs A of the graph $T^E = (V, A)$ models all possible state transitions precipitated by the events. There is an arc from vertex $(s, c_1, c_2, \ldots, c_k - 1, \ldots, c_{d+1})$ to vertex $(s, c_1, c_2, \ldots, c_k, \ldots, c_{d+1})$ if and only if (i) the c_kth event in the kth chain can trigger a state transition from s to t by applying a causal rule r, and (ii) without violating the time-interval information this event can be the next event to occur exactly after the first $c_k - 1$ events in the kth chain and the first c_i events in the ith chain $(1 \leq i \leq d+1, i \neq k)$ have all occurred . We attach the event-rule tag (e, r) to the arc to indicates that event e can trigger such a state transition by applying causal rule r.

5. Let l_i be the the total number of events in the ith event chain. Search the graph T^E starting from the source vertex $(s_0, 0, \ldots, 0)$ to determine, for each goal state s_G that satisfies the goal G, whether $(s_G, l_i, l_2, \ldots, l_{d+1})$ is reachable from $(s_0, 0, \ldots, 0)$. [4] If such a vertex $(s_G, l_i, l_2, \ldots, l_{d+1})$ is reachable, do step 6. If no such vertex is reachable, reports that the goal can never be reached in the system.

6. Find a directed path from the root vertex $(s_0, 0, \ldots, 0)$ to the goal vertex $(s_G, l_i, l_2, \ldots, l_{d+1})$. The event-rule tags of the arcs along this directed path them immediately give us an event sequence q in Q^E that ends in the goal state s_G, together with the information of the rules applied by the events in q to end in s_G. Scan the events in the event sequence q one by one, and assign each event the earliest possible time point that is (1) within its own time interval and (2) later than every time points already assigned to the preceding events in q.

[1] The sorting is conducted by comparing the numerical values of the end points. If an $AT(e)$ end point of an event e and an $BT(e')$ end point of an event e' turns out to have the same value, we consider $BT(e')$ to be smaller than $AT(e)$.

[2] We do not need to construct the entire graph T^E explicitly before the search. Instead, we only progressively expand portions of the graph as needed when we conduct the search.

[3] Let l_i be the the total number of events in the ith event chain. Each state s in $S_{\mathbf{X}}$ is mapped to $\prod_{1 \leq i \leq d+1}(1 + l_i)$ vertices in V

[4] See Lemma 2.

Lemma 2 *According to the formation of the graph T^E as described in the **TemporalSearch** procedure, a goal state s_G satisfying the goal can be reached immediately after an event sequence in Q^E if and only if there exists a vertex $(s_G, l_i, l_2, \ldots, l_{d+1})$ that is reachable from the root vertex $(s_0, 0, \ldots, 0)$ in T^E.*

Theorem 1. *Temporal projection with n nondeterministic events, $O(1)$ variables, and $O(1)$ degree of uncertainty over the ordering of events is solvable in $O(n \log n)$ time.*
Proof sketch. The *TemporalSearch* procedure has to search through vertices of the form $(s, c_1, c_2, \ldots, c_d)$ where $s \in S_{\mathbf{X}}$ indicates a system state while c_i ($1 \leq i \leq d$) is an event counter for the ith event chain. Consider the event counter for the first chain. For each fixed c_1 value for that counter, it turns out there are only $O(1)$ possible values for the event counters of any other chains. Let e be the event corresponding to the fixed c_1 value for the first chain. Because of the $O(1)$ degree of uncertainty over the ordering of events, there are only $O(1)$ possibilities about the what events in any other chain can occur after the c_ith event in the first chain has occurred and before the $(c_i + 1)$th event in the first chain occurs. Since the first chain is at most of length n, there is only an $O(n)$ number of possible combinations of c_i values, $1 \leq i \leq d$. Therefore the number of vertices reachable by the *TemporalSearch* procedure from the source vertex is no more than $O(n)$ times the size of the state space S_X, which is again $O(n)$ given the $O(1)$ variables. Since there is only $O(1)$ events chains given the $O(1)$ degree of uncertainty, the maximal out degree of the vertices in the graph is $O(1)$. So there is only $O(n)$ arcs explored. Since graph search can be done efficiently in time linear in the number of vertices and arcs explored (Corman & Leiserson 2001), we can conclude the *TemporalSearch* procedure takes $O(n))$ to search the graph T^E. The extra $O(n \log n)$ time needed in step 1 to sort the end points of n time intervals then becomes the dominant factor of the overall computational time. Therefore the procedure has $O(n \log n)$ time complexity.

Theorem 2. *Temporal projection with n nondeterministic events, $O(\log n)$ variables, and $O(1)$ degree of uncertainty over the event ordering is solvable in time polynomial in n.*
Proof sketch. The analysis in the proof sketch of Theorem 1 still works here except that now the size of the state space $S_{\mathbf{X}}$ is polynomial in n. Therefore the number of vertices reachable by the *TemporalSearch* procedure from the source vertex is no more than $O(n)$ times the size of the state space S_X, which is polynomial in n. Since graph search can be done efficiently in time polynomial in the number of vertices explored, we can conclude the procedure is a polynomial time algorithm in this case.

4 Complexity Tradeoffs

Though the *TemporalSearch* procedure in the previous section provides polynomial-time performance guarantee given the $O(1)$ overlapping of time intervals and a polynomial-size state space. Temporal projection with time-interval information in general is hard. The results in this section show that (1) the number state variables of the system, (2) whether the events are deterministic or nondeterministic, and (3) the degree of uncertainty in event ordering all contribute to the computational complexity of temporal projection with time-interval information. For convenience of comparison, we depict the computational complexity results in Table 1 (based on Theorem 3, Theorem 4, and Theorem 5), Table 2 (based on Theorem 1 and Theorem 2), and Table 3 (based on Theorem 6 and Theorem 7) respectively according to different levels of uncertainty over the event ordering.

Theorem 3. *Temporal projection with n nondeterministic events, $O(n)$ variables, and no uncertainty over the ordering of events is NP-Complete.*
Proof sketch.
This is established by a reduction from the 3SAT problem (Garey & Johnson 1979). Given an instance of the 3SAT problem, for each of the n clauses in that instance, we can create a nondeterministic event which has three causal rules to simulate the nondeterminism in choosing one of the three literals in that clause and fix it as true. The time-interval information set these events to occur in a fixed order as the clauses are ordered. For each literal, we create two state variables to record whether its logic value is fixed yet and what its truth true is if its value is fixed already. In the initial state, we set the values of these variables to reflect that the truth values of none of these literals are fixed yet. For each clause, we create a variable to record whether we have already successfully set the logical value of one of the three literals to make satisfy the clause. Let's refer to the set of these variables as X. Initially, none of the clauses are satisfied yet, and in the initial state all the variables in X initially are set to false. To find a solution to the 3SAT instance, we simply have to look at the n corresponding events in the fixed order one by one and decide the rule to apply each time. The 3SAT instance is satisfiable if and only if it is possible that in the end of the event sequence all the variables in X are set to true, which means all the clauses are satisfied.

Theorem 4. *Temporal projection with n nondeterministic events, $O(\log n)$ variables, and no uncertainty over the ordering of events is solvable in time polynomial in n.*
Proof sketch.
Starting from the initial state as the only reachable state before any event occurs, we can record what states are reachable after one more event occurs. Since the size of state space is polynomial in n and the events occur in a fixed or-

	Deterministic	Nondeterministic
$O(1)$ variables	$O(n \log n)$ time	$O(n \log n)$ time
$O(\log n)$ variables	$O(n \log n)$ time	polynomial in n
$O(n)$ variables	$O(n \log n)$ time	NP-Complete

Table 1: Complexity of temporal projection without uncertainty over the ordering of events

	Deterministic	Nondeterministic
$O(1)$ variables	$O(n \log n)$ time	$O(n \log n)$ time
$O(\log n)$ variables	polynomial in n	polynomial in n
$O(n)$ variables	unknown	unknown

Table 2: Complexity of temporal projection with $O(1)$ degree of uncertainty over the ordering of events

	Deterministic	Nondeterministic
$O(1)$ variables	polynomial in n	polynomial in n
$O(\log n)$ variables	NP-Complete	NP-Complete
$O(n)$ variables	NP-Complete	NP-Complete

Table 3: Complexity of temporal projection with $O(n)$ degree of uncertainty over the ordering of events

der, we can process the n events one by one to trace the reachable states after one more event occurs. Each time it takes time polynomial in n to determine whether any state satisfying the goal is reachable after one more event occurs.

Theorem 5. *Temporal projection with n nondeterministic events, $O(1)$ variables, and no uncertainty over the ordering of events is solvable in $O(n \log n)$ time.*
Proof sketch.
It follows the same analysis as described in the proof of the previous theorem except that it is a constant-size the state space. So it takes only constant time to trace the reachable states after one event. It takes $O(n \log n)$ time to sort the events according to the $AT(e)$ of each event e. After that it takes $O(n)$ time to process the n events one by one to trace the reachable states.

Theorem 6. *Temporal projection with n deterministic events, $O(\log n)$ variables, and $O(n)$ degree of uncertainty over the ordering of events is NP-complete.*
Proof sketch.
This is established by a reduction from the Forbidden-pairs-of-arcs problem (Garey & Johnson 1979). An instance of this problem provides a directed graph and a number of forbidden pairs of arcs and the task is to determine whether there is a path from a source vertex to a target vertex in that graph without using both arcs in any of the forbidden pairs. The problem is NP-complete even if the arcs are disjoint. We can use $O(\log n)$ variables to encode every vertex in the directed graph. The source vertex and the target vertex are then encoded as the initial state and a goal state. Arcs in the directed then corresponds to state transitions. For each forbidden pairs, we can create a deterministic event with two causal rules, each of which can simulate state transitions alone the forbidden pairs. For each arc not in any of the forbidden pairs, we also create a deterministic event with one rule that can trigger a state transition corresponding to the arc. Finding a path in the forbidden-pair problem is then reduced to finding an event sequence to reach the goal state from the initial state.

Theorem 7. *Temporal projection with n nondeterministic events, $O(1)$ variables, and $O(n)$ degree of uncertainty over the ordering of events is solvable in time polynomial in n.*
Proof sketch.
This can be established by adapting the proof of Theorem 7 in (Lin & Dean 1996). The idea is to enumerate the constant number of state trajectories that reach a goal state from the initial state. For each enumerated trajectory, it can be determined in polynomial time whether there is an event sequence that can actually generate such a trajectory.

5 Conclusion

In this paper, we formally define the temporal projection problem to reason about events with time-interval information. We propose an algorithm to exploit the event-chain structure embedded in the time-interval information. The algorithm transforms the temporal projection problem into a search problem to improve the computational efficiency. We also present both positive and negative complexity results that show how (1) the number state variables of the system, (2) whether the events are deterministic or nondeterministic, and (3) the degree of uncertainty in event ordering all contribute to the tradeoffs of computational complexity.

References

Corman, T. H., and Leiserson, C. E. 2001. *Introduction to Algorithms*. 2nd edition.

Dean, T., and Boddy, M. 1988. Reasoning about partially ordered events. *Artificial Intelligence* 36(3):375–399.

Dean, T., and Wellman, M. 1991. *Planning and Control*. Morgan Kaufmann.

Garey, M. R., and Johnson, D. R. 1979. *Computers and Intractability*. W. H. Freeman and Company.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning*. Morgan Kaufmann.

Lin, S., and Dean, T. 1996. Localized temporal reasoning using subgoals and abstract events. *Computational Intelligence* 12(3):423–449.

Manna, Z., and Pnueli, A. 1995. *Temporal verification of reactive systems*. Springer-Verlag.

Nebel, B., and Bäckström, C. 1992. On the computational complexity of temporal projection and plan validation. In *Proceedings AAAI-92*, 748–753.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach,*. Prentice Hall, 2nd edition.