Towards Tractable Reasoning on Temporal Projection Problems

Xing Tan
Semantic Technologies Laboratory
Department of Mechanical and Industrial Engineering
University of Toronto
Email: xtan@mie.utoronto.ca

Michael Gruninger
Semantic Technologies Laboratory
Department of Mechanical and Industrial Engineering
University of Toronto
Email: gruninger@mie.utoronto.ca

***Abstract*—It is well known that reasoning with AI temporal projection problems is difficult. Determining the Possible Truth problem, a basic temporal projection decision problem, in the so-called Simple Event System remains NP-complete. In this paper, two types of constraints, on the graph-theoretic representation of the cause-and-effect relationships between events and on the partial orders of events, are further considered upon this boundary NP-complete Possible Truth problem, in an attempt to gain tractability. In particular, we show that the problem is still NP-complete even if the cause-and-effect graphs maintain a strongly restricted topology, whereas it is tractable if the graph is closely associated with the set of pairwise disjoint partial orders, which in addition is hierarchically structured.**

I. Introduction

Temporal projection involves reasoning about consequences of events in dynamical systems. It has broad applications in a variety of different domains, from manufacturing process control in Operational Research to Semantic Web Services in AI Knowledge Representation and Reasoning.

In real world, major and direct causes and effects of events can often be observed without ambiguity. Hence, the cause-and-effect relationships between events in a dynamical system model are usually characterized with complete knowledge. However, due to the uncertainty or choices in various circumstances, the order in which events will occur can only be partially specified. Since a partial order could potentially involve a set of exponentially many total orders, one should expect that determining consequence of events is difficult.

The analysis of computational properties of temporal projection problems was initialized by Dean and Boddy [1], where events are partially ordered, and the framework for cause-and-effect relationships are based on the propositional STRIPS representation. The intractability of temporal projection is verified in [1]: determining the Possible Truth (PT) problem, a basic temporal projection decision problem, remains NP-complete even when several severe restrictions are presented; whereas only unrealistically trivial ones are polytime solvable.

A technical follow-up written by Neble and Bäckström [5] points out that one special case, the PT problem in Simple Event Systems (SES), conjectured to be polynomial time solvable in [1], is indeed NP-complete. Meanwhile, the NP-completeness proof for the first time relates the PT problem in SES with the Path Avoiding Forbidden Pairs of Edges (PAFP-E) problem in graph theory.

In order to receive tractability, two types of constraints, on the graph-theoretic representation of the cause-and-effect relationships between events and on the partial orders of events, are further considered in this paper on the PT problem in SESs, which insofar as we know is the most restricted version of an NP-complete PT problem. As a matter of fact, our results on both types of constraints are based on the results of the PAFP-E problem.

For the constraints regarding the cause-and-effect relationships between events, we first show that the problem of PAFP-E in a special class of graphs, 2-layered planar s-t-DAGs, is NP-complete, by transforming from the NP-complete One-in-Three 3SAT problem ([2], page 259, and also see appendix of this paper) into it. Using this result, we move on to show that the PT problem in a SES, where its cause-and-effect graph is a 2-layered planar s-t-DAG, is NP-complete, by a polynomial transformation construction in exactly the same way as the one presented in [5]. This work is summarized in Section III.

The work in Section IV contains three parts. First, motivated by a recent result [4], in which a polytime algorithm is proposed to solve the Path Avoiding Forbidden Pairs of Vertices (PAFP-V) problem where the forbidden pairs are with the so-called (Vertice) Hierarchical Structure (VHS), we revise the algorithm to show that PAFP-E with a similarly defined (Edges) Hierarchical Structure (EHS) can also be solved in polytime. Second, we show that, if for any $a_1 \prec a_2$ in its set of pairwise disjoint partial orders there exists a path in its cause-and-effect graph G from a_2 to a_1, a PT problem in SES can be transformed into a PAFP-E problem and the existence of solution in both directions is perserved by this transformation. Third, if the transformed PAFP-E is with EHS, it can be solved in polytime, which clearly means that the original special class of PT problem is also polytime solvable.

Other sections of this paper serve the purposes as follows. Section II provides background materials: definitions and complexity results of the PT problem are presented in Section II.A; whereas Section II.B includes all graph-theoretic concepts related to the PAFP problems. Section V is a brief summary. The Appendix Section provides definitions of SAT problems, One-In-Three 3SAT problem in particular.

II. PRELIMINARIES

A. Temporal Projection

The formalization of a given domain as an Event System and the PT problem is equivalent to the one given by [1] and essential the same as the one given by [5].

Definition 1 (Causal Structure): A Causal Structure (CS) is defined as a 3-tuple $\Phi = \langle \mathcal{E}, \mathcal{P}, \mathcal{R} \rangle$ where

- $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ is a set of event types;
- $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ is a set of propositional conditions;
- $\mathcal{R} = \{r_1, r_2, \ldots, r_p\}$ is a set of causal rules in the form $r_i = \langle t_i, \varphi_i, \alpha_i, \delta_i \rangle$ where
  - $t_i \in \mathcal{T}$ is the event type that triggers r_i,
  - $\varphi_i \subseteq \mathcal{P}$ is the set of preconditions,
  - $\alpha_i \subseteq \mathcal{P}$ is the set of added conditions,
  - $\delta_i \subseteq \mathcal{P}$ is the set of deleted conditions.

The concept of CS is introduced to specify how a given state, which is a subset of $\mathcal{P}$, evolves over actual occurrences of events. Given a Φ and a state S, an actual event a with event type $t \in \mathcal{E}$ is applicable in S if and only if the preconditions for t are satisfied in S. If applicable, a changes S by adding some conditions to S and removing some others from it, in the way specified by the rule $\mathcal{R}$. However, if not applicable, a is effectless to S. Only those applicable ones in a sequence of events changes S in a sequential manner.

The notion of Event System is introduced to describe a set of actual events that are subject to temporal constraints in the form of partial orders:

Definition 2 (Event System): An Event System (ES) is a 6-tuple $\Theta = \langle \mathcal{E}, \mathcal{P}, \mathcal{R}, \mathcal{A}, \mathcal{O}, \mathcal{I} \rangle$ where

- $\mathcal{T}, \mathcal{P}, \mathcal{R}$ are the same as the ones defined in Θ,
- $\mathcal{A} = \{a_1, a_2, \ldots, a_p\}$ is a set of actual events, such that for each a_i, $type(a_i) \in \mathcal{T}$,
- $\mathcal{O}$ is a set of partial orders on $\mathcal{A}$,
- $\mathcal{I}$ is a set of initial conditions in the system.

Given a set of events $\mathcal{A}$ in an Event System Θ , there exists a set of completions, i.e., linear extensions, on $\mathcal{A}$ with their sizes equal to $|\mathcal{A}|$. One basic temporal projection decision problem is defined as follows.

Definition 3 (Possible Truth Problem): The Possible Truth (PT) problem involves deciding in an ES Θ whether there exists a sequence on partially ordered events $\mathcal{A}$ such that a given condition p holds in the state obtained from execution of the sequence.

In general, temporal projection problems involve evaluating the possible truth (PT) and the necessary truth (NT) of conditions, however, our primary concern in this paper is with the PT problems. It should also be noted that the PT problems, as formulated by [1] and [5], determine the value of p in the state right after certain event $a \in \mathcal{A}$, whereas we evaluate p in the state after the sequence of all events in $\mathcal{A}$ is executed. Regarding computational complexity, these two variants are equivalent.

As indicated by the following theorem, temporal projection is intractable.

Theorem 1: The PT problem is NP-complete ([1], Theorem 2.1).

In order to obtain tractability, several types of constraints on CS of Θ are considered in [1]. For example:

- R1 each $t \in \mathcal{T}$ has but one applicable rule in $\mathcal{R}$;
- R2 $|\varphi| = |\alpha| = |\delta| = 1$ for all rules in $\mathcal{R}$, and the size of the initial condition is also 1, $|\iota| = 1$;
- R3 $\delta \subseteq \varphi$;
- R4 $|\mathcal{R}| = 2$.

The following two theorems outline the complexity bounds for the PT problems.

Definition 4: An Event System Θ with restrictions of R1, R2, R3, is a Simple Event System (SES), written as Θ_{simple}.

Theorem 2: The PT problem with Θ_{simple} is NP-complete ([5], Theorem 3.3; conjectured to be polytime solvable in [1]).

Theorem 3: The PT problem with Θ_{simple}, plus R4, is polytime solvable (see the Algorithm on page 398 of [1]).

B. Graph-theoretic Concepts

A *digraph* is a pair $G = \langle N, E \rangle$ where N is a finite set of nodes and E consists of edges, i.e., ordered pairs of nodes on N. A direct graph is *acyclic* (DAG) if there does not exist a sequence $\langle v_1, v_1, \ldots, v_{k-1}, v_k \rangle$ of nodes such that $v_1 = v_k$ and $(v_i, v_{i+1}) \in E$ for $i = 1, \ldots, k$.

A DAG is *layered* if N can be partitioned into a sequence of p disjoint subsets (layers), $l_1, \ldots, l_p$, such that all edges in E are between consecutive layers. A DAG is *planner* if it can be drawn without crossing edges.

Definition 5: A *2-Layered Planar s-t-DAG* is a layered planner DAG with exactly one source node s and one sink node t, and the size of each layer is at most 2.

Definition 6 (PAFP of Vertices): Given a digraph $G = (V, E)$, specified vertices $s, t \in V$, a set $F_{vertices}$ of pairs of vertices from V, the problem of finding a path avoiding forbidden pairs of vertices (PAFP-V) involves finding a *s-t* path such that at most one vertex from any pair in $F_{vertices}$ is contained in the path.

Theorem 4: PAFP-V, and its variant PAFP-E, in which the the set of forbidden pairs F_{edges} consists of edges, are NP-complete ([3], [2]).

Definition 7 (Vertices Hierarchical Structure): Consider two nodes $n_1, n_2 \in N$, if there exists a path from n_1 to n_2 we write $n_1 \prec_N n_2$. The set $F_{vertices}$ in PAFP-V, is with the Vertices Hierarchical Structure (VHS) if for any two pairs, say $(n_1, n_2), (n_3, n_4) \in F_{vertices}$, there does not exit a path in G such that $n_1 \prec_N n_3 \prec_N n_2 \prec_N n_4$.

Definition 8 (Edges Hierarchical Structure): Consider two edges $e_1, e_2 \in E$, if there exists a path from e_1 to e_2 we write $e_1 \prec_E e_2$. The set F_{edges} in PAFP-E is with the Edges Hierarchical Structure (EHS) if for any two pairs, say $[e_1, e_2], [e_3, e_4] \in F_{edges}$, there does not exit a path in G such that $e_1 \prec_E e_3 \prec_E e_2 \prec_E e_4$.

Theorem 5: PAFP-V with its $F_{vertices}$ satisfying VHS can be solved in polytime ([4]).

III. 2-Layered planar DAG and Temporal Projection

Definition 9 ($PAFP2LP_{edges}$): $PAFP2LP_{edges}$ involves finding a path from s to t with forbidden pairs of edges in a 2-Layered Planar s-t-DAG.

Theorem 6: $PAFP2LP_{edges}$ is NP-complete.

Proof: It is easy to see that $PAFP2LP_{edges}$ is in NP: for any *s-t* path, it can be checked in polynomial time if the path satisfies the F_{edges}.

To show that $PAFP2LP_{edges}$ is NP-hard, we will transform One-In-Three 3SAT (see [2] page 259, and the appendix) to $PAFP2LP_{edges}$. From an arbitrary instance of One-In-Three 3SAT $S = \langle U, C\rangle$, where $U = \{u_1, u_2, ..., u_n\}$ is a set of variables and $C = \{c_1, c_2, ..., c_m\}$ such that $|c_j| = 3$ for $1 \leq j \leq m$ is the set of clauses, we constructed a 2-Layered planar s-t-DAG $G = \langle N, E\rangle$. The node set N consists of a source s, a sink t, $|n|-1$ connectors $\{cntr_i | 1 \leq i \leq |n|-1\}$, and two nodes, v_{ij} and d_{ij} (d stands for "dummy"), for each literal p_{ij} in C. The edge set is

$$\begin{gathered} E = \{(s, v_{11}), (s, d_{11})\} \\ \cup\{(v_{i1}, d_{i2}), (d_{i1}, d_{i2}), (d_{i1}, v_{i2}) | 1 \leq i \leq m\} \\ \cup\{(v_{i2}, d_{i3}), (d_{i2}, d_{i3}), (d_{i2}, v_{i3}) | 1 \leq i \leq m\} \\ \cup\{(v_{i3}, cntr_i), (d_{i3}, cntr_i) | 1 \leq i < m\} \\ \cup\{(v_{i3}, t), (d_{i3}, t) | i = m\}. \end{gathered}$$

The set of forbidden pairs F_{edges} is a union of F_1 and F_2, where
$F_1 = \{[(v_{i1}, d_{i2})(d_{i2}, v_{i3})], [(d_{i1}, d_{i2})(d_{i2}, d_{i3})] | 1 \leq i \leq m\}$ ensures a path will pass through one and only one literal node for each clause component, and F_2 contains forbidden pairs in which the two edges of each forbidden pair enter, respectively, two literal nodes that are negation of each other:
$F_2 = \{[node_1, v_{ij})(node_2, v_{kl})] | v_{ij} = \neg v_{kl}\}$,
where $node_1$ and $node_2$ represent the unique processor nodes to v_{ij} and v_{kl}, respectively, .

This is obviously a polynomial time transformation. Now consider a constrained path P from s to t in G, F_1 on P makes one and only one corresponding literal in each C_i of S true whereas F_2 on P ensures that the assignment is consistent. Hence, the existence of P implies the One-In-Three satisfiability of S. The converse direction can be proved similarly. ■

An illustration of this construction is given in Figure 1. From a One-In-Three 3SAT instance $S = \{[a, b, c], [b, \neg c, d]\}$, we construct $G = \langle N, E\rangle$ and F_{edges}, where

$$\begin{gathered} N = \{v_{11}, v_{12}, v_{13}, v_{21}, v_{22}, v_{23}\} \\ \cup\{d_{11}, d_{12}, d_{13}, d_{21}, d_{22}, d_{23}\} \\ \cup\{S, T, Cntr1\}, \end{gathered}$$

$$\text{and } F_2 = \{[(d_{12}, v_{13}), (d_{21}, v_{22})]\}.$$

Elements of E is shown in Figure 1 and the construction of forbidden pairs in F_1 can also be easily derived.

It is worth noting that 1) Theorem 6 holds even if the forbidden pairs in F_{edges} are pairwise disjoint; 2) the general transformation in [3] produces a DAG that is not planar; and 3) their restricted *in-and-out-degree-at-most-two* can only be further transformed into a planar DAG that is at least 3-layered.

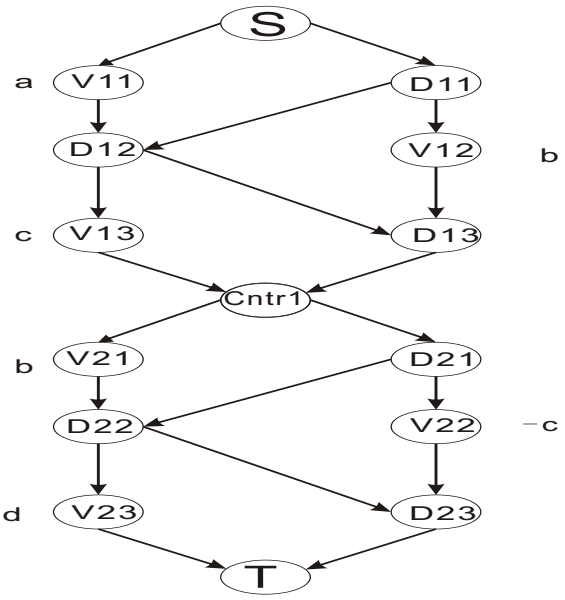

Fig. 1. Transformation from an One-In-Three 3SAT Instance to a PAFP Problem

In the proof of Theorem 3.3 of [5] (Theorem 2 in this paper), NP-complete PAFP-E in general DAGs with disjoint F_{edges} is transformed into the PT problem in Θ_{simple} to show that the problem is NP-hard. Given Theorem 6, we can obtain the main result of this section as follows, by transforming from this NP-complete $PAFP2LP_{edges}$ into the current problem.

Definition 10: A $\Theta_{simple\_2lp\_stdag}$ is a Θ_{simple} and its cause-and-effect graph is a 2-layered planar s-t-DAG.

Theorem 7: The PT problem in $\Theta_{simple\_2lp\_stdag}$ is NP-complete.

IV. Partial Orders and Temporal Projection

As indicated in [4] and Theorem 5 of this paper, PAFP-V in DAGs, with $F_{vertices}$ satisfying VHS, can be solved in polytime. The theorem as follows shows that the polytime solvability can also be achieved on PAFP-E in DAGs, with F_{edges} satisfying the EHS.

Theorem 8: PAFP-E in DAGs, with F_{edges} satisfying EHS can be solved in polytime.

Proof: The algorithm, which is a modification to the Algorithm in Section 4 of [4], is presented in Algorithm 1 of next page.

The proof on the correctness of the algorithm consists of Corollary 9, which shows that applying Step1, Step2, or Step3 will not affect the result on the existence of a forbidden path, and Corollary 10, which indicates that when the algorithm finishes, a new problem with trivial solution is reduced from the original one.

Algorithm 1 The Algorithm to Find a Path in DAGs With Forbidden Pairs of Edges

Input: $\langle N, E, F_{edges}, s, t\rangle$, where s is the source node and t is the sink node in N:
while $|V| > 2$ **do**
Step1: find a free edge say $e_1 = (x, y)$ (i.e., there does not exist another edge say e_2 such that $[e_1 e_2] \in F_{edges}$ or $[e_2 e_1] \in F_{edges}$); replace $x, y \in N$ by a new node $u \in N$ such that, in E, any edge (v, x) or (v, y) is replaced by (v, u) and any edge (x, w) or (y, w) is replaced by (u, w); remove (x, y) from E;

Step2: find an edge pair $[(x, y)(y, z)] \in F_{edges}$ and for any edge (y, u) such that $u \neq z$ add an edge (x, u); for any pair in F_{edges} that involves (x, y), add a new pair where (x, y) is replaced by (x, u); remove (x, y) from E, remove any pairs that contains (x, y) from F_{edges};

Step3: remove an edge pair $[(x, y)(u, v)] \in F_{edges}$ such that there does not exist a path from (x, y) to (u, v), $(x, y) \not\prec (u, v)$.
end while
if $E = \phi$ **then**
print: Does Not Exist A Path
else
print: Exist A Path
end if

Corollary 9: Given $G = \langle N, E, F_{edges}, s, t\rangle$, and $G' = \langle N', E', F'_{edges}, s, t\rangle$, which is obtained by applying either Step 1, Step2, or Step3, there exists a forbidden path P in $G \Longleftrightarrow$ there exists a forbidden path P' in G'.

"$\Longrightarrow$" direction: Given $P \subseteq G$ there are three cases to consider

1) $P \subseteq G'$;
2) P contains a free edge $e_{free} = (x, y)$ that is removed in G' by applying Step 1;
3) P contains $(x, y)(y, z)$, but by applying Step 2, (x, y) is removed and (x, z) is added in G'.

For case 1, P is a forbidden path for G'. For case 2, remove e_{free} in P and concatenate the two segments to obtain P', which is a forbidden path for G'. For case 3, replacing $(x, y)(y, z)$ in P with x, z to obtain P'. Note that applying Step 3 does not affect the solution. The proof for the "$\Longleftarrow$" direction is similar.

Corollary 10: Given $G = \langle N, E, F_{edges}, s, t\rangle$, if no Step 1, Step2, or Step3 can be applied to G, then $V = \{s, t\}$.

We prove this by contradiction. Suppose, on the contrary, there exists an edge e_1, e_1 with some other edge e_2 must appear in F_{edges} otherwise Step 1 can be applied. The forbidden pair $[e_1, e_2]$ must be separated by some e_3 otherwise Step 2 can be applied. Similarly, e_3 must also be in F_{edges} with some e_4. But all of the following cases

1) $e_1 \prec_E e_3 \prec_E e_4 \prec_E e_2$;
2) $e_1 \prec_E e_3 \prec_E e_2$, $e_3 \prec_E e_4$ and $e_4 \not\prec_E e_2$;
3) $e_1 \prec_E e_4 \prec_E e_2$, $e_3 \prec_E e_4$ and $e_1 \not\prec_E e_3$

will not be allowed, otherwise we can start with $e_3 \prec_E e_4$ with two extra edges e_1 and e_2. The two remaining cases

1) $e_3 \prec_E e_1 \prec_E e_4 \prec_E e_2$;
2) $e_1 \prec_E e_3 \prec_E e_2 \prec_E e_4$

violate EHS.

Given $|N| = n$ and $|E| = m$ (assume that $n < m$), we have $|F_{edges}| \leq m^2$. In Step 1, searching for a free edge in F_{edges} and then removing the edge if one is found, require $O(m)$ time if appropriate data structure such as an array of linked lists is applied to store the forbidden pairs in F_{edges}, concatenating the two ends of the removed edge into a new node requires $O(n)$ time, thus Step 1 is bounded by $O(m)$. In Step 2, searching for a forbidden pair connected by a node requires $O(m^2)$ by brute force, the subsequent operations if one such pair is found require another $O(n)$, making Step 2 $O(m^2)$ bounded . As for Step3, since the redundancy check for each forbidden pair requires an $O(m)$ time breadth-first or depth-first search, Step 3 is bounded by $O(m^3)$. Observe that each application of Step 2 or Step 3 will reduce the size of F_{edges} by one, the number of iterations is bounded by $O(m^2)$. The overall time complexity is thus bounded by $O(m^4)$. Note that it is likely that a more sophisticated implementation results in a reduced complexity.

Finally, it should be noted that the algorithm can be easily modified to print out the forbidden path, if one exists in the original graph. ■

Next we show the implication of Theorem 8 on PT problems.

Definition 11: A Θ_{simple_dag} is a simple event system and its cause-and-effect graph is an acyclic digraph.

Definition 12: A $\Theta^{orders}_{simple\_dag}$ is a Θ_{simple_dag}. In addition, if $a_1 \prec a_2$ is in its set of pairwise disjoint partial orders $\mathcal{O}$, there exists a path in the cause-and-effect graph G from a_2 to a_1 (i.e., $a_2 \prec_E a_1$).

It should be noted that, from the above definition, we have in $\Theta^{orders}_{simple\_dag}$ 1) $a_2 \prec_E a_1$ implies $a_1 \not\prec_E a_2$, as G is a DAG; 2) if $a_1 \prec a_2$, there does not exist any permutation of $\mathcal{A}$, in which both a_1 and a_2 are applicable, because we have $a_2 \prec_E a_1$ in G.

Theorem 11: The PT problem in $\Theta^{orders}_{simple\_dag}$ has a solution if and only if the PAFP-E in $\langle G, F_{edges}\rangle$, derived by the rule $a_1 \prec a_2 \in \mathcal{O} \Longleftrightarrow [a_2, a_1] \in F_{edges}$, has a forbidden path.

Proof: Suppose in a PT problem, there exists a sequence seq, which is a permutation $\mathcal{A}$ and achieves the goal condition t from the initial condition s. Those applicable events in seq construct a subsequence seq_{app} in the form $[s, \ldots, t]$, which has a one-to-one correspondence to a path in G in the form $path_{app} = [n(s) \ldots, n(t)]$ in G. By the above rule, and the fact that seq_{app} also satisfies $\mathcal{O}$, any two edges constructing a forbidden pair cannot both appear in $path_{app}$. Hence, $path_{app}$ is a forbidden path in G.

Suppose the derived $\langle G, F_{edges}\rangle$ has a forbidden path $path_{forbidden} = [n(s) \ldots, n(t)]$. The path $path_{forbidden}$ corresponds to a sequence of applicable events in $\Theta^{orders}_{simple\_dag}$,

written as seq_{app}. Now, we can partition $\mathcal{A}$ into $\mathcal{A}_1 \cup \mathcal{A}_2$, and $\mathcal{O}$ into $\mathcal{O}_1 \cup \mathcal{O}_2$. Assume that no event in seq_{app} ever appears in $\mathcal{A}_1$ or $\mathcal{O}_1$, we can obtain a sequence seq_{irvt}, which is irrelevant to seq_{app}, by running topological sorting on $\mathcal{O}_1$. Given that, 1) partial orders in $\mathcal{O}$ are pairwise disjoint and 2) for any $(a_1 \prec a_2) \in \mathcal{O}_2$, exactly one of a_1 or a_2 will appear in seq_{app}, it is safe to extend seq_{app} to seq_{ext} by inserting a_1 before a_2 if a_2 is in seq_{app}, and vice versa. The concatenation $seq_{ext} + seq_{irrt}$ is a permutation of $\mathcal{A}$ that can achieve t from s. ■

Definition 13: A $\Theta^{orders\_ehs}_{simple\_dag}$ is a $\Theta^{orders}_{simple\_dag}$. In addition, If an F_{edges} is constructed such that $a_1 \prec a_2 \in \mathcal{O} \iff [a_2, a_1] \in F_{edges}$, $\langle G, F_{edges} \rangle$ is with EHS.

The main result of this section is stated as follows.

Theorem 12: The PT problem in $\Theta^{orders\_ehs}_{simple\_dag}$ can be solved in polytime.

Proof: By Theorem 11, it is safe to transform the PT problem in $\Theta^{orders\_ehs}_{simple\_dag}$ into $\langle G, F_{edges} \rangle$ satisfying EHS, for solution. By Theorem 8, $\langle G, F_{edges} \rangle$ satisfying with EHS can be solved in polynomial time. ■

V. Concluding Remarks

The most restricted PT problem known to be NP-complete is the one in Θ_{simple}. Starting from this problem, we pushed the tractability boundary by adding additional constraints on the topology of the graph indicating the cause-and-effect relationships and on the structure of the set of partial orders. In particular, we showed in this paper that the PT problem in Θ_{simple} remains NP-complete if its cause-and-effect graph is a 2-layered planner s-t-DAG, whereas it is tractable if the graph is associated in a special way with the set of pairwise disjoint partial orders, and the set is in EHS.

Layered planar digraphs are special DAGs in which efficient parallel algorithms exist for computing the shortest path [7]. From this perspective, the NP-completeness result for PAFP of edges in 2-layered planar s-t-DAGs should have significance in its own right. One interesting observation is that, 2-layered planar s-t-DAGs are not series-parallel decomposable (see Theorem 1 in [8] for recognition of series-parallel decomposable DAGs with respect to N graph as induced subgraph); meanwhile, it can be derived from the construction in [3] that PAFP of edges in series-parallel-decomposable DAGs is NP-complete.

The paper [4] also introduces a halving structure that leads to NP-complete subclass of PAFP of vertices; the structure is of little use for the purposes of this paper since tractability is obtained from the EHS. Nevertheless, it seems that other structures on partial orders could also lead to tractable PT problems: in [9], for example, Yinnone proposes a so-called skew symmetry for forbidden pairs of vertices and shows that PAFP under this restriction can be solved in polytime. We believe this result can be adopted to obtain another class of tractable PT problems, using the same method as the one applied in Section VI of this paper.

The two main results presented in this paper, particularly the NP-completeness one, is in favor of Nebel and Bäckström's claim in page 135 of [5], that is, it seems that the structure of cause-and-effects does not contribute much to the intractability of temporal projection problems and the set of partial orders is almost exclusively the source of complexity. However, given that the tractability of the other result in this paper depends on the close association between the set of orders and the cause-and-effect graph (see Theorem 11), the complete validity of this claim demands further account of comprehensive investigations. In particular, it would be interesting to see, by keeping the EHS, but relaxing other constraints (for example, allowing multiple preconditions and effects; multiple rules for events), whether intractability could possibly be restored.

References

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

[2] Garey, M.R., Johson, D.S.: Computers and intractability - a guide to NP-completeness. W.H. Freeman and Company, Cambridge, MA, USA (1979)

[3] Gabow, H. N., Maheshwari, S. N., Osterweil L. J.: On two problems in the generation of program test paths. IEEE transactions on software engineering **SE-2 No.3** (1976) 227–231

[4] P. Kolman and Pangrac, O.: On the complexity of paths avoiding forbidden pairs. Discrete Applied Mathematics **157** (2009) 2871–2876

[5] Nebel, B., Bäckström, C.: On the computational complexity of temporal projection, planning, and plan validation. Artificial Intelligence **66** (1994) 125–160

[6] Sipser, M.: Introduction to the theory of computation, Second Edition. PWS Publishing Company, Boston, MA, USA (2006)

[7] Subramanian, S., Tamassia, R., Vitter, J.: An efficient parallel algorithm for shortest paths in planar layered digraphs. Algorithmica **14** (1995) 332–339

[8] Valdes, J., Tarjan, R., Lawler, E.: The recognition of series parallel digraphs. SIAM J. Comput. **Vol. 11, No.2** (May 1982) 298–312

[9] Yinnone, H. On paths avoiding forbidden pairs of vertices in a graph. Discrete Applied Mathematics **74** (1997) 85–92

Appendix

A. Satisfiability Problems

A *Boolean formula* is an expression involving

1) Boolean variables, i.e., the variables having either 1 or 0 as their values,
2) Binary operators $\wedge$, $\vee$, or $\neg$, standing for the logical AND and OR or Negation,
3) Parentheses that can alter the default precedence of operators: $\neg$ is higher than $\wedge$, which is higher than $\vee$.

A *literal* is a Boolean variable or a negated variable. A *clause* is a disjunction of one or more literals. A Boolean formula is in *conjunctive normal form* (*CNF*) if it is a conjunction of one or more clauses. It is in *3CNF* if all the clauses have three literals.

A Boolean formula is *satisfiable* if some assignment to its variables makes the formula evaluate to 1. The *SAT Problem* is to test whether a Boolean formula is satisfiable. The *3SAT Problem* is to test whether a Boolean formula in 3CNF is satisfiable. The *One-In-Three 3SAT Problem* is to test whether a Boolean formula in 3CNF is satisfiable and each clause contains one and only one literal that is assigned with value 1. It is well known that all these SAT problems are NP-complete.