Reasoning about Discrete Event Sources

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

Abstract

We investigate the modelling of workflows, plans, and other event-generating processes as discrete event sources and reason about the possibility of having event sequences ending in undesirable states. In previous research, the problem is shown to be NP-Complete even if the number of events to occur is fixed in advance. In this paper, we consider possible events sequences of indefinite length and show that many interesting cases of such reasoning task are solvable in polynomial time.

1 Introduction

Events precipitate changes in the world. Both the deliberate actions of workflows, plans, and the dynamics of uncontrollable processes in the operational environment may generate events or initiate other event-generating processes. Individual workflows and plans may work fine when executed separately, but fail when executed concurrently or in the presence of other event-generating processes not accounted in the original planning stage.

It is a challenging task to validate workflows and plans in the presence of exogeneous events. Incorporating the capacity of reasoning about processes that change the world continuously into planners is computationally costly (McDermott 2003). Subtle semantic issues also arise and are not fully resolved (Fox, Howey, & Long 2005) when considering the modelling of (i) processes that change the world continuously throughout some time intervals and (ii) the cause-and-effect relationship between deliberate actions and the event-generating processes they initiate at the same time point.

For semantic simplicity, in this paper we focus on event-generating processes that generate events to change the world only at discrete time points. We view workflows, plans, and other event-generating processes as event sources, and consider the simple plan validation task of verifying that none of the event sequences possibly generated by the event sources can end in undesirable states. We model the concurrency of the processes via the interleaving of event occurrences (Aceto & Murphy 1996) (Reiter 2001), assuming that each event occurs instantaneously at a time point and that no two events would occur at the same time point. The state is modelled by the values of a set of boolean state variables, each modelling a condition that is either true or false at a given time point. When an event occurs, it precipitates a state transition according to a set of causal rules (Dean & Boddy 1988) (Lin & Dean 1996) associated with the event.

We are particularly interested in validating production plans (Joshi & Smith 1994) (Lin & Dean 1995) and workflow procedures (van der Aalst & van Hee 2002) (Vajpayee 1998) in the presence of other discrete event sources. In these domains, the underlying event-generating processes often have series of stages associated with different types of events that may occur. Since the processes may iterate through these stages for an indefinite number of times in different ways, there is uncertainty about the types, the order, and the effects of the events that may occur. Different event sequences may lead to very different trajectories of the system's evolution. It is therefore important to verify none of the possible event sequence can lead to undesirable states.

Unlike the classical STRIPS planning, which is solvable in polynomial time as a graph reachability problem when the underlying state space is polynomial in size (Ghallab, Nau, & Traverso 2004), reasoning about the reachability of system states over all possible event sequences is shown to be NP-complete even if the exact number of events to occur is fixed in advance and the underlying state space is polynomial in size (Lin & Dean 1996) (Dean & Boddy 1988). In this paper, event sources can generate event sequences of indefinite length. This makes the task even more challenging.

The paper is organized as follows. Section 2 describes the modelling of event sources. Section 3 models the dynamics of events. Section 4 defines the validation task and presents the polynomial-time solvable cases from the perspective of automata theory, which indicates model checking techniques for verifying automata may be applied here to deal with cases involving state spaces of exponential sizes.

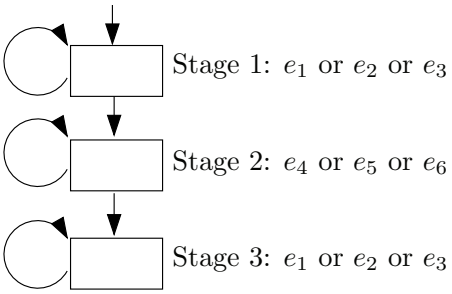

Figure 1: A production-control planworkflow with three stages and six types of operations

2. Modelling Discrete Event Sources

The physical world, the workflows, the plans being executed, and other processes in the operational environment generate events of various types such as storms and earthquakes, operational actions, power failure, or equipment malfunction.

Representing event sequences as strings of event types: Given the set of event types E, we represent an event sequence q of h events as a string $e_{q_1}e_{q_2}\ldots e_{q_h}$ in E^* where e_{q_i} is the event type associated with the ith event in the sequence and $e_{q_i} \in E$ for $1 \leq i \leq h$. For simplicity, we refer to the ith event by its event type as e_{q_i}. We say that $q' = e_{q_1}e_{q_2}\ldots e_{q_{h'}}$ where $h' \leq h$ is a prefix of the event sequence q.

Event sources and possible event sequences: An event source O is associated with a set of event types E, and generates events of such event types. We use the notation L_O^E to denote the set of event sequences that O may possibly generate from the time O is activated to the time it becomes inactive. We use $\overrightarrow{L}_O^E$ to denote the set of event sequences $\{q'|q' \text{ is a prefix of } q, q \in L_O^E\}$. $\overrightarrow{L}_O^E$ is the set of possible event sequences that O may have generated by some point in the time line.

Example: Consider the production-control workflow depicted in Figure 1. The production workflow has three stages involving six kinds of production operations. The production process starts with stage 1, followed by stage 2 and then stage 3. Each stage may be repeated for an indefinite number of iterations. For each iteration in stage 1 and stage 3, either an operation of e_1 type, or an operation of e_2 type, or an operation modelled of e_3 type must be performed to initiate or wrap up the entire production process. For each iteration in stage 2, either an operation of e_4 type, or an operation of e_5 type, or an operation of e_6 type must be performed. We can consider the production workflow as event source O and $E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$ as the set of event types associated with O. The string $e_3e_4e_6e_5e_1e_2$ represents an event sequence in L_O^E with six events of the event types e_3, e_4, e_6, e_5, e_1, and e_2 respectively, one by one in that exact order.

$O = (N, E, R, P)$
$N = \{P, U, S_1, S_2, S_3\}$
$E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$
$R = \{$
$\quad P \rightarrow S_1PS_3$
$\quad P \rightarrow U$
$\quad U \rightarrow S_2|S_2U$
$\quad S_1 \rightarrow e_1|e_2|e_3$
$\quad S_2 \rightarrow e_4|e_5|e_6$
$\quad S_3 \rightarrow e_1|e_2|e_3$
$\}$

Figure 2: A context-free event source represented as a context-free grammar $O = (N, E, R, P)$

The string e_3e_4 represents a possible event sequence in $\overrightarrow{L}_O^E$ with two events of the event types e_3 and e_4 respectively.

Regular event sources: For event-generating processes related to production control and workflow procedure, often they transition between a number of stages triggering events of types associated with these stages like the production control domain in Figure 1. We can view them as finite-automata generating events sequences that can be described as regular expressions over the event types, and model them as regular event sources. A regular event source O is symbolically represented as a regular expression in terms of the union operator $|$, the concatenation operator $\cdot$, the closure operator *, and the set of event types E as the alphabet. The set of possible complete event sequences L_O^E (as strings in E^*) that O may generate is simply the set of strings in E^* specified by the regular expression O.
Example: The production workflow in Figure 1 can be modelled as a regular event source $O = (e_1|e_2|e_3)^* \cdot (e_4|e_5|e_6)^* \cdot (e_1|e_2|e_3)^*$. The set of possible complete event sequences L_O^E is simply any event sequences with (i) an indefinite number of events of event type e_1 or e_2 or e_3 generated by the iterations in stage 1, followed by (ii) an indefinite number of events of event type e_4 or e_5 or e_6 generated by the iterations in stage 2, and then followed by (iii) an indefinite number of events of event type e_1 or e_2 or e_3 generated by the iterations in stage 3

Context-free event sources: Sometimes the numbers of events triggered in individual or blocks of the stages are related and the possible event sequences need to be described by a context-free grammar. For this purpose, we represent a context-free event source O symbolically as a context-free grammar $O = (N, E, R, P)$ where N is a set of nonterminal symbols, E is the set of event types serving as the set of terminals in the grammar, R is the set of grammar rules, while P is a nonterminal symbol in N serving as the start symbol. The set of possible complete event sequences L_O^E (as strings in E^*) is simply the set of strings that can be produced by the context-free grammar O.
Example: A variant production workflow with the same stage structure like the one in Figure 1 has strong correla-

tion between stage 1 and stage 3. Each iteration in stage 1 attaches a component to the main frame of a product while there must be a corresponding iteration in stage 3 to fasten the attachment of that component. This imposes an additional constraint requiring the number of iterations in stage 1 to equal the number of iterations in stage 3. No regular event source can model this since no regular expression can meet this constraint according to the pumping lemma of automata theory (Hopcroft, Motwani, & Ullman 2001). Instead, the context-free event source O depicted in Figure 2 can model it. The set of possible complete event sequences L_O^E is simply the collection of every event sequence with (i) an indefinite number of events of event type e_1 or e_2 or e_3 generated by the iterations in stage 1, followed by (ii) an indefinite number of events of event type e_4 or e_5 or e_6 generated by the iterations in stage 2, and then followed by (iii) the same number of events as in stage 1 of event type e_1 or e_2 or e_3 generated by the iterations in stage 3

Forming compound event sources: Multiple event sources $O_1, O_2, \ldots, O_k$ together form a compound event source O'. (The individual event source O_i itself may be a compound event source.) They may coexist to generate events independently (i.e. by interleaving), or one event source may be constrained to activate only sometime after another one is deactivated (i.e. by partial ordering), or one event source must be activated immediately after another event source becomes inactive (i.e. by concatenation), or only exactly one of event sources is nondeterministically selected for activation (i.e. by union), or the same kind of event source may be initiated and finished for an indefinite number of times one after another (i.e. by closure). Let $E_1, E_2, \ldots, E_k$ be the sets of event types associated with event sources $O_1, O_2, \ldots, O_k$ accordingly. The set of event type associated with O' is simply $E' = \bigcup_{1 \leq i \leq k} E_i$. In the following, we more formally describe the kinds of formation of compound event sources mentioned above.

Forming compound event sources by interleaving: When multiple event sources $O_1, O_2, \ldots, O_k$ coexist and independently generate events, they form a compound event source $O' = Interleave(O_1, O_2, \ldots, O_k)$. The set of possible event sequences generated by O' is $L_{O'}^{E'} = \{q | q$ is the interleaving of some event sequences $q_1, q_2, \ldots, q_k$ where $q_i \in L_{O_i}^{E_i}$ for $1 \leq i \leq k\}$.

Forming compound event sources by partial ordering: When a set of event sources $O_1, O_2, \ldots, O_k$ coexist with a partial order $\prec$ over them, they form a compound event source $O' = PartialOrder_{\prec}(O_1, O_2, \ldots, O_k)$. The set of possible event sequences generated by O' is $L_{O'}^{E'} = \{q | q = q_{i_1} q_{i_2}, \ldots, q_{i_k}$ where $q_{i_l} \in L_{O_{i_l}}^{E_{i_l}}$ for $1 \leq l \leq k$, $\langle O_{i_1}, O_{i_2}, \ldots, O_{i_k} \rangle$ is a permutation of $\{O_1, O_2, \ldots, O_k\}$, and $\langle O_{i_1}, O_{i_2}, \ldots, O_{i_k} \rangle$ is consistent with the partial order $\prec$ over $\{O_1, O_2, \ldots, O_k\}$.

Forming compound event sources by concatenation: The concatenation of two event sources O_1 and O_2 is a compound event source $O' = O_1 \cdot O_2$. The set of event sequences possibly generated by O' is $L_{O'}^{E'} = \{q | q = q_1 q_2$ where $q_1 \in L_{O_1}^{E_1}$ and $q_2 \in L_{O_2}^{E_2}\}$. This is actually a spacial case of forming compound event sources by partial order with two event source one is ordered before the other.

$\mathbf{X} = \{X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8\}$
$s_0 = (true, true, true, true, true, true, true, true)$
$G = (\ \langle X_1, false\rangle\langle X_2, false\rangle\langle X_4, false\rangle$
$\quad \langle X_5, false\rangle\langle X_7, false\rangle\langle X_8, false\rangle\ )$
$E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$
$O = Interleave(O_1, O_2) \prec O_3$ where
$O_1 = (e_1|e_2|e_3)^* \cdot (e_4|e_5|e_6)^* \cdot (e_1|e_2|e_3)^*$
$O_2 = (e_1|e_5)^*$ and $O_3 = (e_2|e_6)^*$

The causal rules associated with the event types in E:
$e_1 =$
$\{(\langle X_1, true\rangle\langle X_2, true\rangle) \rightarrow (\langle X_1, false\rangle\langle X_2, false\rangle\langle X_3, true\rangle)$
$\ (\langle X_1, true\rangle\langle X_3, false\rangle) \rightarrow (\langle X_1, false\rangle\langle X_3, true\rangle\langle X_4, true\rangle)$
$\}$
$e_2 =$
$\{(\langle X_1, true\rangle\langle X_3, true\rangle) \rightarrow (\langle X_1, false\rangle\langle X_3, false\rangle\langle X_2, true\rangle)$
$\ (\langle X_1, true\rangle\langle X_2, false\rangle) \rightarrow (\langle X_1, false\rangle\langle X_3, true\rangle\langle X_4, true\rangle)$
$\}$
$e_3 =$
$\{(\langle X_4, true\rangle\langle e, true\rangle) \rightarrow (\langle X_4, false\rangle\langle X_5, false\rangle\langle X_6, true\rangle)$
$\}$
$e_4 =$
$\{(\langle X_4, true\rangle\langle X_6, true\rangle) \rightarrow (\langle X_4, false\rangle\langle X_6, falae\rangle\langle e, true\rangle)$
$\}$
$e_5 =$
$\{(\langle X_1, false\rangle\langle X_4, true\rangle) \rightarrow (\langle X_1, true\rangle\langle X_4, false\rangle\langle X_7, false\rangle)$
$\}$
$e_6 =$
$\{(\langle X_1, true\rangle\langle X_4, false\rangle) \rightarrow (\langle X_1, false\rangle\langle X_4, true\rangle\langle X_8, false\rangle)$
$\}$

Figure 3: Modelling the system dynamics of the production-control domain with the five-tuple $(\mathbf{X}, s_0, G, E, O)$

Forming compound event sources by union: The union of two event sources O_1 and O_2 is a compound event source $O' = O_1|O_2$. The set of event sequences possibly generated by O' is $L_{O'}^{E'} = L_{O_1}^{E_1} \bigcup L_{O_2}^{E_2}$.

Forming compound event sources by closure: The closure of an event source O associated with the event types E is a compound event source O^*. The set of event sequences possibly generated by O^* is $L_{O^*}^{E} = \{q | q = \epsilon$ or $q = q_1 q_2 \ldots q_k$ where $1 \leq k$ and $q_i \in L_O^E$ for $1 \leq i \leq k\}$.

3 Modelling the Causes and Effects of Events

In the following, we model the causes and effects of events in terms of simple *if-then* causal rules and model dynamics of the entire discrete event system in terms of a five-tuple $(\mathbf{X}, s_0, G, E, O)$, where (i) $\mathbf{X}$ is a set of state variables that determine the states of the system, (ii) s_0 is the initial state, (iii) G is a state expression to implicitly specify a set of goal states, (iv) E is a set of event types together with their associated *if-then* rules, and (v) O is an event source (compound or not) that generates events of the types in E. Figure 3 depicts the five-tuple $(\mathbf{X}, s_0, G, E, O)$ describing the causes

and effects of the events and modelling the system dynamics of a variant of the production workflow in Figure 1. In addition to the production workflow modelled as the event source O_1, there are two other event sources O_2 and O_3. O_1 and O_2 generate event concurrently while O_3 is activated immediately after both O_1 and O_2 are finished.

State variables, states, and the initial state: The value of a state variable X_i is either $true$ or $false$ at a time point. The values of the state variables in a set of m boolean state variables $\mathbf{X} = \{X_1, X_2, \ldots, X_m\}$ all together at a time point determine the system state at that time. The initial state is the system state in the very beginning. We can represent a state as $(x_1, x_2, \ldots, x_m)$ where x_i is the value of X_i in the state u. We say the underlying state space $S_{\mathbf{X}} = \{true, false\}^m$ is a *polynomial state space* if its size is polynomial in the size of the given five-tuple $(\mathbf{X}, s_0, G, E, O)$.

Example: For the production-control domain in Figure 3, different production-control conditions are modelled by the state variables in $\{X_1, X_2, \ldots, X_8\}$. Variables X_1, X_2, and X_3 indicate whether the concentration of three chemicals are within their threshold levels respectively. Variables X_4 and X_5 indicate whether the temperature and the air pressure within a specific compartment are within their threshold. Variables X_6, X_7, and X_8 indicate whether the insulation, the automatic pressure adjustment device, and the temperature sensor are fully functional respectively. The values of the eight state variables in $\mathbf{X} = \{X_1, X_2, \ldots, X_8\}$ all together at a point of time determine the system state at that time. In the beginning of the production process, all the conditions represented by the state variables $X_1, X_2, \ldots, X_8$ are true. This is represented by the initial state s_0.

State expressions and goal states: 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. The goal expression is a state expression. The goal states is the set of states in which the goal expression is true.
Example: For the production-control domain in figure 3, whenever the production-control conditions represented by the variables X_1, X_2, X_4, X_5, X_7, and X_8 are all false at the same time, it would severely damage the quality of the final product. This is represented by the state expression G, and we would like to avoid ending in a state where G is true.

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 causal rule $r = \alpha \rightarrow \beta$ can be viewed as 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: Each event type is associated with a set of causal rules. When an event occurs, it instantly triggers a state transition at a point of time. We assume no two events can occur at the same point of time. 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 (associated with e's event type) 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 the 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. $S_{\mathbf{X}}$.
Example: Each of the six event types in E in Figure 3 models the corresponding kind of production operations of the same name in Figure 1. The causal rules associated with the event type model how an operation of that kind may change the conditions of the production environment based on the system state right before the operation. For example, in Figure 3 there are two causal rules associated with e_1. For the first causal rule associated e_1, it is applicable in every state where both the conditions represented by variables X_1 and X_2 are true. This rule models the possibility that after a production operation of e_1 type occurs, both the conditions represented by variables X_1 and X_2 become false and the condition represented by variable X_3 is guaranteed to be true while the other conditions remain unchanged.

Modelling nondeterminism in system dynamics: An event e is a nondeterministic event if there exists a state s in which two or more explicit causal rules associated with e's event type are applicable. The system state after the nondeterministic event e occurs in the state s could be any one of several possible next states. Such nondeterminism models uncertainty of possible outcomes.

Example: When an event of the event type e_1 occurs in in the initial state s_0, both rules associated with event type e_1 are applicable in the initial state s_0. They model two different possible outcomes in that situation. In this situation, exactly one of the applicable rules is picked nondeterministically. In case the first rule is picked, both the conditions represented by variables X_1 and X_2 become false while the other

conditions remain true. In case the second rule is picked, both the conditions represented by variables X_1 and X_3 become false while the other conditions remain true. In the production-control example, only events of the event types e_1 and e_2 are nondeterministic events.

4 Workflow and Plan Validation by Temporal Projection

Given a five-tuple $(\mathbf{X}, s_0, G, E, O)$, as described in Section 2 and Section 3, where (i) O is a compound event source associated with a set of event types E that models workflows, plans, and other event-generating processes to validate, (ii) $\mathbf{X}$ is a set of state variables modelling the factors involved in the causes and effects of the events, (iii) s_0 is the initial system state, and (iv) G specifies a set of undesirable system states to avoid, we can either detect a potential fault or ensure no possible event sequences can end in any undesirable situation by conducting the following temporal projection task.

The temporal projection task: Given a five-tuple $(\mathbf{X}, s_0, G, E, O)$, the task of temporal projection is to determine the existence of possible event sequences in $\overrightarrow{L}_O^E$ immediately following which the system may end in one of the goal states. If such event sequences do exist, we would like to generate one of such sequences as an evidence.

Example: Given the five-tuple $(\mathbf{X}, s_0, G, E, O)$ in the production-control example in Figure 3, the temporal projection task is to determine whether the compound event source $O = Interleave(O_1, O_2) \prec O_3$ could possibly generate an event sequence in $\overrightarrow{L}_O^E$ that may end in a state where the goal G is true and thus would severely damaged the quality of the final product. In this particular example, the event sequence $e_3e_4e_6e_5e_1e_2$ with the first causal rules of these events applied would end in that undesirable situation.

Positive complexity results: Temporal projection is shown to be NP-complete when the exact number of events to occur is fixed in advance (Dean & Boddy 1988) even if the underlying state space is polynomial in size (Lin & Dean 1996). However, in the following, we are able to show that temporal projection can be accomplished in polynomial time in many interesting cases involving possible event sequences of indefinite length generated by compound event sources. Note that Proposition 1 to Proposition 4 follow from the theory of automata and formal languages (Hopcroft, Motwani, & Ullman 2001). Based on them, we can relate the task of temporal projection to the task of finding the intersection of languages represented by automata (Hopcroft, Motwani, & Ullman 2001) (Garey & Johnson 1979) and this perspective sheds light on interesting polynomial-time solvable cases described in Theorem 1 to Theorem 6.

Proposition 1. *Let $L^E_{X,s_0,G}$ denote the set of all event sequences in E^* that may end in any of the goal states. Both $L^E_{X,s_0,G}$ and $\overrightarrow{L}_O^E$ are languages over the alphabet E.*

Proposition 2. *The task of temporal projection is equivalent to finding the intersection language of the languages $L^E_{X,s_0,G}$ and $\overrightarrow{L}_O^E$. If the intersection is an empty set, no possible event sequence generated by O can end in a goal state; otherwise any event sequence belonging to the intersection language may end in some goal state.*

Proposition 3. *We can construct a finite automata $M^E_{X,s_0,G}$ that accepts the language $L^E_{X,s_0,G}$. This can be done in time polynomial in the size of the underlying state space $S_{\mathbf{X}}$ and the size of the problem instance $(\mathbf{X}, s_0, G, E, O)$.*

Proposition 4. *We can construct a finite automata M^E_O that accepts the language $\overrightarrow{L}_O^E$ if O is a regular event source. Similarly, we can construct a pushdown automata M^E_O that accepts the language $\overrightarrow{L}_O^E$ if O is a context-free event source. Both can be done in time polynomial in the size of the underlying state space $S_{\mathbf{X}}$ and the size of the problem instance $(\mathbf{X}, s_0, G, E, O)$.*

Theorem 1. *Temporal projection regarding a polynomial state space and a regular event source is solvable in polynomial time.*
Proof sketch: For the case of a regular event source, according to the propositions above, we can construct two finite automata $M^E_{X,s_0,G}$ and M^E_O in polynomial time. It is well known that (i) constructing a finite automata from $M^E_{X,s_0,G}$ and M^E_O that accepts the intersection of the languages $L^E_{X,s_0,G}$ and L^E_O can be done in polynomial time (Garey & Johnson 1979), and (ii) testing whether the automata represents an empty language can be done in polynomial time (Hopcroft, Motwani, & Ullman 2001).

Theorem 2. *Temporal projection regarding a polynomial state space and the composition of a polynomial number of regular event sources by concatenation, union, and closure is solvable in polynomial time.*
Proof sketch: Because regular expressions are closed under the concatenation operation, the union operation, and the closure operation, the set of complete event sequences generated by the compound event source can still be represented as a regular expression. Therefore the compound event source is still a regular event source and thus by Theorem 1 we have this result.

Theorem 3. *Temporal projection regarding a polynomial state space and the composition of a constant number of regular event sources by interleaving is solvable in polynomial time.*
Proof sketch. By Proposition 4, for each regular event source, we can construct a finite automaton accepting exactly the event sequences the source may generate. Then we can construct in polynomial time from these automata, a finite automaton M^E_O that accepts exactly the interleaving

of the event sequences possibly generated by these regular event sources. By Propositions 2 to 4, temporal projection can be done in polynomial time by finding the intersection language of L_O^E and $L_{X,s_0,G}^E$ and testing its emptiness.

Theorem 4. *For every problem instance of any of the polynomial-time solvable cases in Theorem 1 to Theorem 3, we are also able to construct in polynomial time a finite automata that can accept exactly all the possible event sequences that may end in the goal states.*
Proof sketch: All these cases can be reduced to finding the intersection of two finite automata, and as described in the proof of Theorem 1 we can construct an automaton to represent the intersection in polynomial time.

Theorem 5. *Temporal projection regarding a polynomial state space and a context-free event source is solvable in polynomial time. We are also able to construct in polynomial time a push-down automata that can accept exactly all the possible event sequences that may end in the goal states.*
Proof sketch:
For the case of a context-free event source, it follows from almost the same lines of proof development for Theorem 1, except that M_O^E now is a pushdown automaton for the context-free event source. It is well known that (i) constructing a pushdown automata from $M_{X,s_0,G}^E$ and M_O^E that accepts the intersection of the languages $L_{X,s_0,G}^E$ and L_O^E can be done in polynomial time, and (ii) testing whether the pushdown automata represents an empty language can be done in polynomial time.

Theorem 6. *Temporal projection regarding a polynomial state space and a compound event source formed by partially ordering a polynomial number of event sources is solvable in polynomial time if (i) the partial order embeds a constant number of of chains in it and (ii) temporal projection regarding each individual component event source is solvable in polynomial time.*
Proof sketch: By temporal projection and the encoding of extra rules, for each component event source O and for each possible state immediately before O is activated, we can reason about (i) whether undesirable states are reachable while O is active and (ii) what state may be reached immediately after O is deactivated. This allows us to encode causal rules to abstract each event source as an abstract event, consider each chain of event sources as a regular event source, and examine the compound event source as the interleaving of a constant number of regular event sources with additional constraints all in polynomial time.

5 Conclusion

We investigate the modelling of workflows, plans, and other event-generating processes as discrete event sources and reason about the possibility of having event sequences ending in undesirable states. We show that many interesting cases of temporal projection for workflow and plan validation are actually solvable in time polynomial in the size of the state space.

In recent years, binary decision diagrams and other model checking techniques have been successfully used to deal with automata of exponential sizes for hardware and system verification (Clarke, Grumberg, & Peled 1999) and also for planning (Ghallab, Nau, & Traverso 2004). Since we establish the positive results mostly by relating the task of temporal projection to that of finding the intersection of automata, it indicates that by applying similar model checking techniques to the underlying automata for temporal projection, we may tackle exponential state spaces involved in workflow and plan validation as well.

References

Aceto, L., and Murphy, D. 1996. Timing and causality in process algebra. *Acta Informatica* 33:317–350.

Clarke, E. M.; Grumberg, O.; and Peled, D. A. 1999. *Model Checking*. MIT Press.

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

Fox, M.; Howey, R.; and Long, D. 2005. Validating plans in the context of processes and exogenous events. In *Proceedings of The 20th National Conference on Artificial Intelligence (AAAI-05)*.

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.

Hopcroft, J. E.; Motwani, R.; and Ullman, J. D. 2001. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2nd edition.

Joshi, S., and Smith, J., eds. 1994. *Computer Control of Flexible Manufacturing Systems : Research and Development*. Springer.

Lin, S., and Dean, T. 1995. Generating optimal policies for high-level plans with conditional branches and loops. In *Proc. Third European Workshop on Planning*. IOS Press.

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

McDermott, D. 2003. Reasoning about autonomous processes in an estimated-regression planner. In *Proceedings of ICAPS'03*.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Vajpayee, S. K. 1998. *Principles of Computer Integrated Manufacturing*. Prentice Hall.

van der Aalst, W., and van Hee, K. 2002. *Workflow Management: Models, Methods, and Systems*. MIT Press.