Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling

New Encoding Methods for SAT-Based Temporal Planning

Masood Feyzbakhsh Rankooh and **Gholamreza Ghassem-Sani**
Sharif University of Technology
Tehran, Iran

Abstract

Although satisfiability checking is known to be an effective approach in classical planning, it has scarcely been investigated in the field of temporal planning. Most notably, the usage of $\exists$-step semantics for encoding the problem into a SAT formula, while being demonstrably quite efficient for decreasing the size of the encodings in classical planning, has not yet been employed to tackle temporal planning problems. In this paper, we define temporal versions of classical $\forall$-step and $\exists$-step plans. We show that when the casual and temporal reasoning phases of a SAT-based temporal planner are separated, these semantics can be used to translate a given temporal planning problem into a SAT formula. We introduce two different types of $\exists$-step encodings in temporal planning. The first encoding method is a temporal version of the classical $\exists$-step encoding. Like its classical counterpart, in the new encoding we suppose a few restrictive simplifying assumptions. On the other hand, by relaxing one of these assumptions, the second type of $\exists$-step encodings, which is often more compact than the first one, is introduced. However, if a temporal planning problem possesses the property that we call required causal simultaneity, neither of our proposed encodings will be expressive enough to represent a valid temporal plan. Nevertheless, we show that this property is rather rare and can be detected in polynomial time. Our experiments indicate that by embedding the proposed encodings into ITSAT, a SAT-based temporal planner based on the $\forall$-step encoding, a considerable improvement is achieved in terms of both speed and memory usage of the planner. The resulting planner significantly outperforms POPF, which is currently the state-of-the-art of temporally expressive planners.

Introduction

Previous research in the field of temporal planning has enormously benefited from employing well-developed classical planning strategies. In fact, many classical planning methods have already been used to tackle temporal planning problems, too. For instance, many successful temporal planners have utilized the ideas of partial order planning e.g., VHPOP (Younes and Simmons 2003) and CPT (Vidal and Geffner 2006). Planning graph analysis has also been adopted by temporal planners such as TGP (Smith and Weld 1999) and TPSYS (Garrido, Fox, and Long 2002). Some other temporal planners have embedded temporal reasoning into heuristic state space search. TFD (Eyerich, Matmuller, and Roger 2009) and POPF (Coles et al. 2010) are two successful instances of this latter approach.

Employing satisfiability checking is another important trend of classical planning research. In this approach, a given planning problem is encoded into a SAT formula. In order to make the corresponding SAT formula finite, the potential plan is assumed to have a finite number of steps. The formula is given as the input to an off-the-shelf SAT solver. The SAT solver tries to find a model for the formula. If such a model exists, the final plan is extracted from it. Otherwise, the number of steps is increased and the whole process is repeated.

SAT-based classical planning was first used to find optimal plans, i.e., plans with minimum number of actions (Kautz and Selman 1992). To guarantee the optimality of the output plan, the formulae must include certain clauses to ban each step from containing more than one action. However, if optimality is not the objective of the planner, forcing single-action steps is not necessary. In an alternative approach, which has been shown to be quite effective (Rintanen, Heljanko, and Niemelä 2006), the planning problem is encoded in such a way that each step of the final plan can have several parallel actions. The usage of multiple-action steps results in a smaller number of steps in SAT formulae, which in turn reduces the number of SAT variables. Since the speed of SAT solvers may exponentially decrease as the number of variables is increased, employing this idea often results in considerably faster planners. Several encoding methods have been introduced to take advantage of such a parallelism. The research in this area is mainly focused on the so-called $\forall$-step and $\exists$-step semantics of valid plans (Rintanen, Heljanko, and Niemelä 2006).

The $\forall$-step semantics allows each step of a plan to include a particular set of actions, only if those actions can be executed in every possible order without affecting the validity of the plan. On the other hand, the $\exists$-step semantics is based on some weaker requirements: for each step of a plan, there must exist at least one possible ordering in which the actions of that step can be arranged without falsifying the validity of

the plan. It should be clear that the ∃-step semantics potentially allows more parallelism than what is permitted by the ∀-step semantics. In fact, ∃-step encoding has been shown to be one of the most efficient methods for converting classical planning problems to SAT formulae (Rintanen, Heljanko, and Niemelä 2006).

Satisfiability checking has also been used to tackle temporal planning problems. STEP (Huang, Chen, and Zhang 2009) and T-SATPLAN (Mali and Liu 2006) are two SAT-based planners that handle temporal constraints by assigning explicit discrete time labels to each step of the encoding. TM-LPSAT (Shin and Davis 2005), which has been designed to solve planning problems defined by PDDL+ (Fox and Long 2002), is another SAT-based planner that can handle temporal planning problems. However, in TM-LPSAT, the steps of the SAT formula do not possess predefined time labels. Instead, the execution time of each step will be stored in a variable whose value is to be determined by an SMT solver (Armando and Giunchiglia 1993).

ITSAT (Rankooh, Mahjoob, and Ghassem-Sani 2012) is yet another example of SAT-based temporal planners. Like TM-LPSAT, ITSAT does not assign explicit time labels to the steps of its encoding. Besides, ITSAT first abstracts out the durations of actions. It then finds a plan that is only causally valid. The plan is then refined in such a way that satisfies temporal constraints imposed by the durations of actions.

In this paper, we generalize the concepts of single-action-step, ∀-step, and ∃-step plans to the temporal planning context. We show that according to our definition of parallel plans, STEP, T-SATPLAN, and TM-LPSAT are all using a temporal version of the single-action-step encoding. We also show that the separation of causal and temporal reasoning has enabled ITSAT to somehow use the temporal version of ∀-step semantics for its encoding. We also introduce a new temporal version of ∃-step semantics and use it to propose two novel encoding methods. The completeness of the proposed semantics is also analyzed in this paper. For testing the efficiency of the new encodings, we replace the ∀-step encoding of ITSAT by each of our proposed encodings and compare the performance of the resulting planner with that of original ITSAT and POPF which are currently the state-of-the-art of temporally expressive planners.

Preliminaries

In this section, we define basic concepts such as temporal states, actions, problems, and plans. Our definitions of these concepts here are consistent with the level 3 of PDDL2.1 (Fox and Long 2003), and have been inspired by the formalization used for TEMPO (Cushing et al. 2007). We assume that the reader is familiar with the definitions of states, actions, and problems of classical planning.

Definition 1 (temporal states) A temporal state, s, is a pair $(state(s), agenda(s))$, where $state(s)$ is a classical planning state and $agenda(s)$ contains all the actions that are started but not yet finished before reaching s.

Definition 2 (temporal actions and events) A temporal action, a, is a quadruple $(start(a), end(a), over(a), dur(a))$ where $start(a)$ and $end(a)$ are two classical planning actions denoting the starting and ending events of a, $over(a)$ is a set of classical preconditions representing the over-all conditions of a, and $dur(a)$ is a positive rational number specifying the duration of a.

Definition 3 (mutual exclusion) Two events, e_i and e_j, are mutually exclusive in the temporal sense if either of the following conditions holds:

- e_i and e_j are mutually exclusive in the classical sense (Blum and Furst 1997).
- e_i (or e_j) is the staring event of action a, and e_j (or e_i) deletes a member of $over(a)$.

Definition 4 (applicability) A set of events, $E = \{e_1, ..., e_n\}$, is applicable in state s, if all following conditions hold:

- For each i, e_i is applicable to $state(s)$ in the classical sense.
- If e_i is the starting event of action a, then $over(a) \subseteq \bigcup_{e \in E} add(e) \cup state(s)$
- If e_i is a starting event, then it does not delete an over-all condition of any member of $agenda(s)$.
- If e_i is the ending event of action a, then a is a member of $agenda(s)$ and e_i does not delete an over-all condition of any other member of $agenda(s)$.
- For all i and j, e_i and e_j are not mutually exclusive.

We say that members of E are simultaneously applied to state s.

Definition 5 (successors) If a set of events $E = \{e_1, ..., e_n\}$ is applicable to state s, it will change s to s' where $state(s')$ is the result of applying all members of E to $state(s)$ in the classical sense and in an arbitrary order, and $agenda(s')$ is determined by the following rule: $agenda(s') = agenda(s) \cup \{a | start(a) \in E\} - \{a | end(a) \in E\}$. s' is also denoted by $succ(s, E)$. Applying a sequence of sets of event to step s is defined by the following recursive rule: $succ(s, \langle E_1, ..., E_n \rangle) = succ(succ(s, E_1), \langle E_2, ..., E_n \rangle)$.

Definition 6 (temporal problems) A temporal problem, P, is a triple (I, G, A) where I is a temporal state such that $agenda(I) = \phi$ representing the initial state, G is a set of classical goal conditions, and A is the set of all possible temporal actions of P.

Definition 7 (temporal plans) A temporal plan π is a sequence $E_1, ..., E_n$, where each E_i is a set of simultaneously executed events representing a step of π. π is valid for problem $P = (I, G, A)$ if there exist a sequence $s_0, ..., s_n$ of temporal states, such that $s_0 = I$, $G \subseteq state(s_n)$, $agenda(s_n) = \phi$, and for every i, $s_i = succ(s_{i-1}, E_i)$. Moreover, there must exist a scheduling function $\tau : \{1, ..., n\} \rightarrow$ Q with the following properties:

- For all i, $\tau(i) < \tau(i + 1)$.

- For each $a \in A$, if $start(a) \in E_i$ and $end(a) \in E_j$, then $\tau(j) = \tau(i) + dur(a)$.

It should be noted that by Definition 7, a valid temporal plan is a sequence of steps where each step includes several simultaneously executed events. In other words, all events of any particular step must be executed at the same time. We call this semantics, 1-step semantics for temporal plans. This is in fact a generalization of classical single-action-step semantics.

Simultaneity of events is necessary for solving some temporal problems. For instance, consider the plan shown in Figure 1(a). In this plan, we have two temporal actions a and b, where the starting event of each is providing the over-all conditions of the other. Consequently, if the goal state is reached by either a or b, both actions have to be started simultaneously. Another example of situations where simultaneity of events is necessary is depicted in Figure 1(b). In this example, the over-all condition of a is added and deleted respectively by the starting and ending events of b. The fact that a and b have equal durations necessitates the simultaneous execution of these two actions.

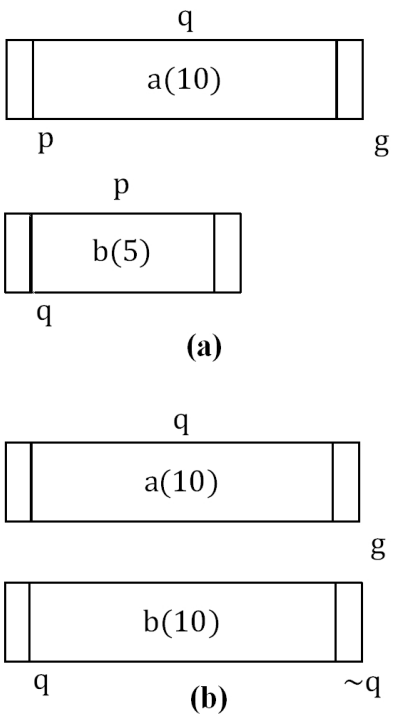

Figure 1. Plans with simulataneous events

ITSAT Planning System

In this section, the ITSAT planning system is briefly described. ITSAT was the first SAT-based temporal planner in which the causal and temporal reasoning tasks were performed in two separate phases. Such a separation is critical for the feasibility of our ∃-step encoding methods discussed later.

To solve a temporal planning problem, ITSAT first abstracts out the durations of actions. In other words, it is assumed that actions can have arbitrary durations. It then encodes the abstract problem into a SAT formula. This abstraction causes the encoding to be very similar to that of classical SAT-based planners. However, beside ordinary clauses used by classical SAT-based planners, ITSAT needs a number of extra clauses to satisfy the over-all conditions of actions. Moreover, there are several clauses to appropriately manipulate the agendas of states before and after each step.

By using the encoding method explained above, ITSAT may find plans that are not temporally valid. However, all the obtained plans are guaranteed to be what we call *causally valid*.

Definition 8 (causally valid temporal plans) A temporal plan π is causally valid for temporal problem P, if it admits all requirements of definition 7, except for the existence of the scheduling function τ, which is optional.

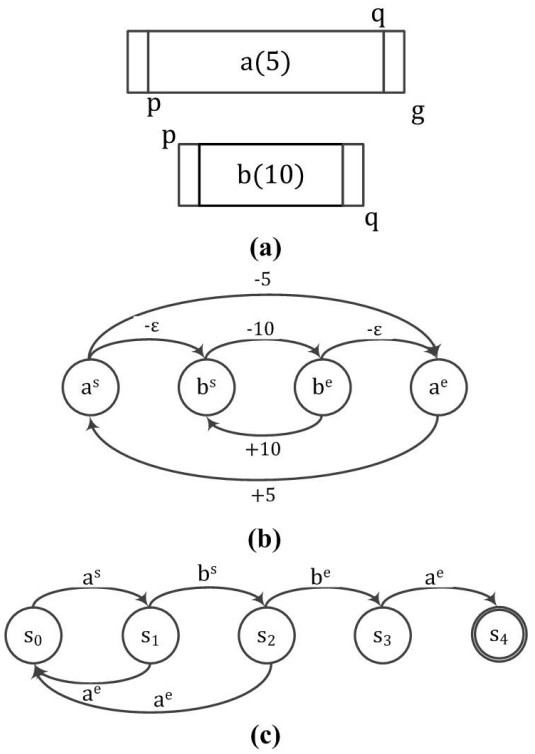

Figure 2. Negative cycle detection in ITSAT

Figure 2(a) represents a causally valid temporal plan that is not temporally valid. That is because action b is to be executed during the execution of action a, while the duration of b is greater than that of a.

To find a temporally valid plan, ITSAT tries to schedule the events of the obtained causally valid plan by solving a particular Simple Temporal Problem (STP) (Dechter, Meiri, and Pearl 1991) that enforces the temporal constraints of the

planning problem. If the STP is inconsistent, there must exist a negative cycle in the corresponding Simple Temporal Network (STN). The STN of the plan in Figure 2(a) is shown in Figure 2(b), where the cycle $a^s b^s b^e a^e a^s$ is a negative cycle. The sequence of events that lead to such negative cycles can be detected by simple Finite State Machines (FSMs). The transition of these FSMs can then be turned into appropriate caluses that collectively prevent such negative cycles from reoccurring. The FSM that detects the cycle $a^s b^s b^e a^e a^s$ is depicted in Figure 2(c). It has been shown that ITSAT is capable of solving problems with the required concurrency property (Cushing et al. 2007), and is competitive with the state-of-the-art temporally expressive planners.

New Semantics for Causally Valid Temporal Plans

According to definition 7, although planners such as STEP, T-SATPLAN, and TM-LPSAT allow parallel execution of actions, they are in fact using the 1-step encoding. That is because these planners assume simultaneous execution of all events in each step.

As we mentioned before, the classical ∀-step semantics permits the execution of more than one action in each step, only if the validity of the plan is not dependent on the execution order of those actions. This can simply be guaranteed by adding a particular clause for each pair of mutually exclusive actions to ensure that those actions will not be included in the same step. However, such a strategy does not work for temporal planning. In temporal planning, because of the temporal constraints between the starting and ending events of actions, the validity of a particular ordering of events of a step, also depends on the ordering of events of other steps. Nevertheless, in ITSAT this problem has been tackled by separating the causal and temporal reasoning phases. In general, if we focus on finding causally valid plans, and postpone the scheduling phase, the mentioned problem about checking the feasibility of imposing different orderings of events in each step will no longer exist.

We now introduce our semantics for causally valid ∀-step and ∃-step temporal plans.

Definition 9 (temporal ∀-steps and ∃-steps) Let $S = \{E_1, ..., E_n\}$ be a set of sets of events, and s_1 and s_2 be two temporal states. S is a temporal ∀-step from s_1 to s_2 if for all one-to-one ordering functions $O : \{1, ..., n\} \rightarrow \{1, ..., n\}$, we have: $s_2 = succ(s_1, \langle E_{O(1)}, ..., E_{O(n)} \rangle)$.
S is a temporal ∃-step from s_1 to s_2 if for some one-to-one ordering functions $O : \{1, ..., n\} \rightarrow \{1, ..., n\}$, we have: $s_2 = succ(s_1, \langle E_{O(1)}, ..., E_{O(n)} \rangle)$

Definition 10 (causally valid ∀-step and ∃-step temporal plans) Let $P = (I, G, A)$ be a temporal planning problem. Suppose $s_0, ..., s_n$ is a sequence of temporal states such that $s_0 = I$, $G \subseteq state(s_n)$, and $agenda(s_n) = \phi$. If for each $1 \leq i \leq n$, $Step_i$ is a ∀-step (∃-step) from s_{i-1} to s_i, then we call the sequence $\langle Step_1, ..., Step_n \rangle$, a causally valid ∀-step (∃-step) temporal plan for P.

∃-step Encodings for Causally Valid Temporal Plans

Classical ∃-step encoding, which has been introduced in (Rintanen, Heljanko, and Niemelä 2006), is based on the ∃-step semantics for classical valid plans. However, for the sake of improving the efficiency of the planner, the following restrictive rules have been also enforced on it.

- Rule 1: Instead of accepting all possible orderings among the actions of each step, only a fixed arbitrary ordering was allowed. Executing a step means executing its actions according to this fixed ordering.
- Rule 2: Preconditions and effects of all actions of each step must be consistent with the states before and after that step, respectively.

The second rule causes an action a to be excluded from a step if there is a contradiction between its effects and that of any other action in that step. Also, a is prevented from being in a step if its precondition is deleted in that step by any other action that according to the predefined fixed ordering is located before a.

In this section, we present two ∃-step encodings for temporal planning. Both proposed encodings are based on the ∃-step semantics for causally valid temporal plans (definition 10). By considering events, instead of actions, both rules mentioned above can be applied to temporal planning, too. While in our first encoding, we respect both rules, our second encoding relaxes the second one. Besides, we also use a third restrictive rule in both proposed encodings. We will later discuss the benefits of the third rule.

- Rule 3: The ending event of each action must be located next to the starting event of that action in the fixed ordering mentioned in Rule 1.

It should be noted that since we are using a total order between all events, our encodings are not completely coherent with the ∃-step semantics defined by definition 10. In fact, for the sake of simplicity, we have assumed that no pair of actions can happen simultaneously in a causally valid plan. This assumption does not render our encodings incomplete unless the problem has a certain property that we call *required causal simultaneity*.

Definition 11 (required causal simultaneity) We say a temporal plan $\pi = \langle E_1, ..., E_n \rangle$ has simultaneity if for some i, we have $|E_i| > 1$. If every causally valid plan of a temporal problem P has simultaneity, we say that P requires causal simultaneity.

Note that while the plan in Figure 1(a) requires causal simultaneity, this is not the case in the plan presented in Figure 1(b). Moreover, while required simultaneity entails required concurrency (Cushing et al. 2007), the reverse is not true. In other words, required simultaneity is more specific than required concurrency. We now show that the existence of the required causal simultaneity has some necessary (but not sufficient) conditions that can be detected in polynomial time.

Let P be a temporal planning problem. Associated with P, we construct a precedence graph $G(P) =< V, E >$ as follows:

- For each event e_i of P, there is a vertex $v_i \in V$.
- If e_i is the starting event of action a, and e_j adds an over-all condition of a, we add a directed edge (v_j, v_i) to E.
- If e_i is the ending event of action a, and e_j deletes an over-all condition of a, we add a directed edge (v_i, v_j) to E.

The precedence graphs of the problems corresponding to the plans of Figure 1(a) and Figure 1(b) are presented in Figure 3(a) and Figure 3(b), respectively.

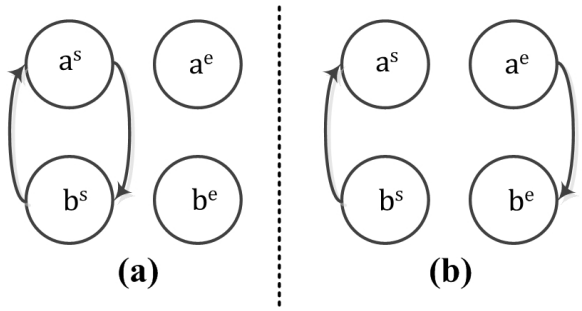

Figure 3. Precedence graphs

Theorem 1. Let P be a temporal planning problem for which there exist a causally valid temporal plan. If P requires causal simultaneity, then $G(P)$ must have a cycle.

Proof sketch. The proof is given by contradiction. Suppose that $G(P)$ is acyclic. By applying the topological sort algorithm to $G(P)$, we obtain a total ordering on the vertices of (and consequently on the events of P). Let π be a causally valid temporal plan for P. We construct a new plan π', which is the same as π except for the previously simultaneous events that are now totally ordered by the topological ordering. The ordering we imposed on the events of π' prevents it from becoming causally invalid (details are omitted here). This contradicts our assumption that P requires causal simultaneity. □

Since topological sort is a polynomial time algorithm, we conclude that detecting the necessary conditions of required causal simultaneity, stated in theorem 1, can be done in polynomial time. Our investigations show that from all domains used in different International Planning Competitions, only the rovers domain has this necessary condition. Therefore, preventing the occurrence of simultaneous events in the causally valid plans will not seriously damage the generality of our method.

We now describe the clauses that are to be included in both new encodings. These are clauses needed for appropriate manipulation of the agendas of states and preventing the over-all conditions of each action from being deleted during the execution of that action.

Clauses Shared by Both Proposed Encodings

Assume that the encoding is to represent a $\exists$-step temporal plan, $\langle Step_1, ..., Step_n \rangle$, where members of each $Step_t$ are applied to the temporal state s_{t-1}, according to a predefined fixed ordering, and map it to state s_t. Suppose that the index of each event represents the location of that event in the fixed ordering. From now on, if we say that an event e_i is earlier (later) than an event e_j, we mean that j is greater (less) than i. We denote the existence of an event e in a step t by the SAT variable Y_e^t. We also use the SAT variable O_a^t to denote that action a is a member of $agenda(s_t)$. The SAT variable X_p^t is used to denote the existance of proposition p in state s_t.

Encoding the initial state and goal conditions of the problem is quite standard. The following clauses are introduced to guarantee that the agendas of states are changed appropriately. We give both a verbal description, and a formal representation of each clause.

- If e_i is the starting event of action a, the presence of e_i in step t implies that a is not a member of $agenda(s_{t-1})$. Besides, if a is a member of $agenda(s_t)$ but not $agenda(s_{t-1})$, then e_i must be present in step t: $(Y_{e_i}^t \rightarrow \sim {O_a}^{t-1})$ and $(\sim O_a^{t-1} \wedge O_a^t \rightarrow Y_{e_i}^t)$. Furthermore, if e_i is present in step t, but e_{i+1}, which according to our third restrictive rule must be the ending event of a, is not present in step t, then a has to be a member of $agenda(s_t)$: $Y_{e_i}^t \wedge \sim Y_{e_{i+1}}^t \rightarrow O_a^t$.
- A description analogous to what is given above also applies to the ending event of a: $(Y_{e_{i+1}}^t \rightarrow \sim {O_a}^t)$, $(O_a^{t-1} \wedge \sim O_a^t \rightarrow Y_{e_{i+1}}^t)$, and $(\sim Y_{e_i}^t \wedge Y_{e_{i+1}}^t \rightarrow O_a^{t-1})$.
- The agenda of the initial and final state of the plan must be empty: for each a, $(\sim O_a^0 \wedge \sim O_a^n)$.

The following clauses are added to the encoding for preventing the over-all conditions of each action from being deleted during the execution of that action. These clauses are representing a schematic message passing strategy, which is inspired by the chaining method used in (Rintanen, Heljanko, and Niemelä 2006). For each step t, proposition p, and event e_i whose corresponding action has p as an over-all condition, the SAT variable $B_{p,i}^t$ denotes whether or not p is deleted in step t, by an event whose index is less than i (i.e., an earlier event). Similarly, variable $A_{p,i}^t$ represents that whether or not p is deleted in step t by an event whose index is greater than i (a later event). Finally, variable D_p^t shows whether or not p is deleted by any event in step t.

- Assume that event e_i deletes proposition p, and e_j is the first event after e_i with the property that its corresponding action has p as an over-all condition. If e_i is present in step t, then a message must be sent to e_j to indicate that p has been deleted earlier in step t: $(Y_{e_i}^t \rightarrow B_{p,j}^t)$ and $(Y_{e_i}^t \rightarrow D_p^t)$.
- Assume that event e_i deletes proposition p, and e_j is the last event before e_i with the property that its corresponding action has p as an over-all condition. If e_i is present in step t, then a message must be sent to e_j to indicate that p will be deleted later in step t: $(Y_{e_i}^t \rightarrow A_{p,j}^t)$.
- Assume that p is an over-all condition of action a, e_i is the ending event of a, and e_j is the first ending event after e_i with the property that its corresponding action has p as an over-all condition. If e_i receives a message implying

that p has been deleted earlier in step t, then it must pass this message to e_j: $(B^t_{p,i} \rightarrow B^t_{p,j})$.

- Assume that p is an over-all condition of action a, e_i is the starting event of a, and e_j is the last starting event before e_i with the property that its corresponding action has p as an over-all condition. If e_i receives a message implying that p is to be deleted later in step t, then it must pass this message to e_j : $(A^t_{p,i} \rightarrow A^t_{p,j})$.
- Assume that p is an over-all condition of action a. If p is deleted in step t, then a cannot be a member of both $agenda(s_{t-1})$ and $agenda(s_t)$: $(D^t_p \wedge O^t_a \rightarrow\sim O^{t-1}_a)$. In other words, if a is started before and ended after step t, its overall conditions cannot be deleted in step t.
- Assume that p is an over-all condition of action a, and e_i is the starting event of a. If e_i is present in step t, and p has been deleted by an event later than e_i in step t, then e_{i+1} (the ending event of a) must be present in step t, too: $(Y^t_{e_i} \wedge A^t_{p,i} \rightarrow Y^t_{e_{i+1}})$. This implies that if a is started but not ended in step t, its over-all condition cannot be deleted later in step t. This is where we are taking advantage of our third restrictive rule: if a is both started and ended in the same step, because its starting and ending events are next to each other, no other event can delete its over-all conditions while a is being executed.
- Assume that p is an over-all condition of action a, and e_i is the ending event of a. If e_i is present in step t, and p is deleted by an event earlier than e_i in step t, then e_{i-1} (i.e., the starting event of a) must be present in step t, too: $(Y^t_{e_i} \wedge B^t_{p,i} \rightarrow Y^t_{e_{i-1}})$. This implies that if a is ended but not started in step t, its over-all condition cannot be deleted in step t by an earlier event.

In addition to the shared clauses stated above, there are other necessary clauses exclusive to each of our new encodings. We present these clauses in their corresponding subsection. We say that an event e requires a proposition p if p is a precondition of e, or e is the starting event of an action that has p as an over-all condition. We say that the a proposition p is relevant to an event e if e requires, adds, or deletes p.

A Natural Extension to the Classical ∃-step Encoding

Our first proposed encoding is a natural extension to the classical ∃-step encoding, as it uses all three restrictive rules stated above. According to the second rule, the preconditions and effects of events of each step must be consistent with the states before and after that step, respectively. Therefore, this part of the encoding, which also includes explanatory frame axioms, is very similar to its corresponding part in the standard classical encodings. However, when we are dealing with temporal planning problems, the over-all conditions of actions must be encoded, too:

- Assume that e is the starting event of an action a, p is an over-all condition of a, and e does not add p. If e is present in any step t, then p must be true in s_{t-1}: $(Y^t_e \rightarrow X^{t-1}_p)$.

Similar to the classical ∃-step encoding, for any $i < j$, if e_i deletes p and e_j requires p, then e_i and e_j cannot be both present in any step. To ensure this, a schematic message passing strategy very similar to the one mentioned before, is employed.

For each step t, proposition p, and event e_i that requires p, $Z^t_{p,i}$ denotes if p is deleted in step t by an event earlier than e_i.

- Assume that e_i deletes p, and e_j is the first event after e_i with the property of requiring p. If e_i is present in step t, then a message must be sent to e_j to indicate that p has been deleted earlier in step t: $(Y^t_{e_i} \rightarrow Z^t_{p,j})$.
- Assume that e_i requires proposition p, and e_j is the first event after e_i with the property of requiring p. If e_i receives a message implying that p has been deleted earlier in step t, then e_i cannot be present in step t, and it must pass this message to e_j: $(Z^t_{p,i} \rightarrow\sim Y^t_{e_i} \wedge Z^t_{p,j})$.

Relaxed ∃-step Encoding

The second restrictive rule presented before, prevents a proposition from being both produced and used in the same step of the final plan. It also does not allow the deletion and production of any particular proposition to happen in the same step. By relaxing these restrictions the encodings can be further compressed, i.e., the relaxation permits more events in each steps. In classical planning, a less relaxed form of Rule 2 has been used in (Wehrle and Rintanen 2007), where the effects of actions in each step can be used by other actions in that step. However, here we totally relax Rule 2 and allow each proposition to be required, added, and deleted in each step as many times as is needed. In the first encoding, explained in the previous subsection, we inform events if any of their requirements is deleted earlier in the same step. In the relaxed encoding, however, the events are informed about the very last change in the truth value of their requirements. No event can occur in the final plan unless the last change in the truth value of any of its requirements has caused the requirement to become true. For each step t, event e_i, and proposition p that is relevant to e_i, $V^t_{p,i}$ represents the truth value of p just before the hypothetical execution of e_i.

- Assume that e_i deletes p, and e_j is the first event after e_i with the property of having p as a relevant proposition. If e_i is present in step t, then a message must be sent to inform e_j that the last change in p has been performed to delete it: $(Y^t_{e_i} \rightarrow\sim V^t_{p,j})$. An analogous discussion is valid when e_i adds p: $(Y^t_{e_i} \rightarrow V^t_{p,j})$. Moreover, if e_i is not present in step t, it must pass any received information regarding the value of p to e_j: $(\sim Y^t_{e_i} \wedge V^t_{p,i} \rightarrow V^t_{p,j})$ and $(\sim Y^t_{e_i} \wedge \sim V^t_{p,i} \rightarrow\sim V^t_{p,j})$.
- Assume that e_i is the last event in the ordering with the property of having p as a relevant preposition. If e_i is not present in step t, the value of p just before e_i must be transferred to the next step: $(\sim Y^t_{e_i} \wedge V^t_{p,i} \rightarrow X^t_p)$ and $(\sim Y^t_{e_i} \wedge \sim V^t_{p,i} \rightarrow\sim X^t_p)$. Moreover, If e_i deletes p we add the clause $(Y^t_{e_i} \rightarrow\sim X^t_p)$ to ensure that the presence of e_i in step t, implies that p is not true in s_t. Again,

an analogous discussion is valid when e_i adds p: $(Y^t_{e_i} \rightarrow X^t_p)$.

- Assume that e_i is the first event in the ordering with the property of having p as a relevant proposition. e_i must be informed of the value of p in s_{t-1}: $(X^{t-1}_p \leftrightarrow V^t_{p,i})$.
- Assume that e_i requires p. If e_i is present in step t, then p must be true just before the execution of e_i: $(Y^t_{e_i} \rightarrow V^t_{p,i})$.

Implementation Details and Empirical Results

We have incorporated our new encoding methods into ITSAT. ITSAT has been slightly modified so that our new encodings can coherently work with it. These modifications have been applied to mutual exclusion analysis and negative cycle detection parts of ITSAT.

It is known that SAT-based planners can significantly benefit from inference about mutually exclusive propositions (Kautz, Selman, and Hoffmann 2006). Any two propositions that remain mutually exclusive after the planning graph has been leveled off (Blum and Furst 1997), cannot be both true in the same state of a valid plan. These are the only mutual exclusion relations that are included in our new encodings.

The second minor modification of ITSAT is related to its FSMs that detect certain negative cycles. As we mentioned before, the negative cycles are prevented by encoding the transitions of a particular FSM to the SAT formula. In our $\exists$-step encodings, the events of each step are assumed to be executed according to a predefined ordering. We have slightly modified the encoding of ITSAT to impose the corresponding order on the transitions of each FSM in each step.

For evaluating the proposed encoding methods, we have tested three versions (i.e., the original $\forall$-step, the $\exists$-step, and the relaxed $\exists$-step version) of ITSAT on the problem sets of previous International Planning Competitions. The experiments have been conducted on a 3.1GHz corei5 CPU with 4GB main memory. Precosat (Biere 2009), which is a free off-the-shelf SAT solver, has been used for satisfying SAT formulae in all three versions of ITSAT. For each problem and each version of ITSAT, several SAT formulae with increasing number of steps were produced. Some of the results are shown in Table 1.

The columns of Table 1 represent: the name of the domain, the problem number, the used encoding method, the number of steps in the encoding, the result of precosat in terms of satisfiability or unsatisfiability of the formula, the number of clauses and variables divided by 1000, the amount of time taken by precosat to determine the result, and the amount of memory needed for saving the formula. For each problem and each encoding method, the results are presented for two cases: unsatisfiable formula with the highest number of steps, and satisfiable formula with the lowest number of steps. We have used symbol $\exists^*$ as an abbreviation for the relaxed $\exists$-step encoding. Symbol ∞ is used in the time column for cases in which precosat has not determined the satisfiability of the formula in 1800 seconds. Please note that in the sokoban domain, an extra comparison between $\exists^*$ and $\exists$ configurations has been presented in Table 1 for problem no. 1 in which the $\forall$ configuration could not find a plan.

As it is shown in Table 1, using our proposed encodings causes a considerable improvement in terms of both speed and memory usage of the planner. Furthermore, the relaxed $\exists$-step encoding is faster than the other two in almost all domains. Although the results presented in table 1, does not cover all the domains used in previous IPCs, we should mention that the same pattern has been observed in nearly all of those domains, too.

domain	prob	enc	steps	res	$\frac{C}{1000}$	$\frac{V}{1000}$	time (s)	mem (MB)
sokoban (2011)	4	$\forall$	29	F	1907	60	∞	134
		$\exists$	13	F	349	86	7	45
		$\exists^*$	6	F	219	101	11	62
		$\forall$	30	T	1973	63	465	138
		$\exists$	14	T	378	93	12	47
		$\exists^*$	7	T	256	118	11	64
	1	$\exists$	12	F	765	187	233	95
		$\exists^*$	4	F	346	157	7	70
		$\exists$	13	T	835	203	120	99
		$\exists^*$	5	T	435	197	48	124
parcprint. (2011)	12	$\forall$	52	F	8080	153	10	509
		$\exists$	31	F	1442	220	1.4	134
		$\exists^*$	13	F	689	74	1.4	139
		$\forall$	53	T	8233	156	8	518
		$\exists$	32	T	1929	293	2.1	137
		$\exists^*$	14	T	744	83	1.7	142
floortile (2011)	10	$\forall$	30	F	250	22	139	21
		$\exists$	11	F	67	18	0.5	10
		$\exists^*$	8	F	69	27	1.3	16
		$\forall$	31	T	257	22	169	21
		$\exists$	12	T	74	20	0.7	11
		$\exists^*$	9	T	78	31	1.1	17
crewplan. (2011)	1	$\forall$	41	F	80	10	∞	8
		$\exists$	41	F	59	17	∞	10
		$\exists^*$	9	F	241	74	0.7	39
		$\forall$	42	T	83	10	2.9	8
		$\exists$	42	T	60	18	2.2	10
		$\exists^*$	10	T	269	83	0.7	45
pegsol (2011)	20	$\forall$	26	F	73	6	∞	7
		$\exists$	12	F	23	6	∞	4
		$\exists^*$	5	F	15	7	0	4
		$\forall$	27	T	77	6	5	8
		$\exists$	13	T	25	7	11	6
		$\exists^*$	6	T	19	8	0.4	4
depots (2004)	10	$\forall$	9	F	25155	145	6	1533
		$\exists$	5	F	864	215	1.3	97
		$\exists^*$	4	F	977	483	2	250
		$\forall$	10	T	28741	163	11	1745
		$\exists$	6	T	1058	260	4.7	103
		$\exists^*$	5	T	1237	607	4	266
driverlog (2004)	15	$\forall$	17	F	15365	217	15	939
		$\exists$	8	F	739	199	2.3	97
		$\exists^*$	6	F	647	246	1	187
		$\forall$	18	T	16606	232	9	1058
		$\exists$	9	T	841	225	8	104
		$\exists^*$	7	T	763	266	2	240

Table 1. Comparing Different Encoding Methods

We have also compared the most efficient version of ITSAT (i.e., ITSAT-$\exists^*$), which uses the relaxed $\exists$-step encoding, with POPF2, the state-of-the-art of temporally expressive planners. For each problem, a time limit of 30 minutes has been imposed on both planners. Starting with a formula with one step, ITSAT-$\exists^*$ uses precosat to satisfy the formula within a time limit of 5 minutes. If the formula is unsatisfiable, 5 more steps will be added to the encoding. Otherwise, ITSAT-$\exists^*$ tries to schedule the achieved causally valid temporal plan, in order to find a valid plan. If such a scheduling is impossible, ITSAT-$\exists^*$ adds the encoding of an FSM, as it was discussed before, to the formula. The whole process is repeated until the 30 minutes time limit is reached.

Although parallel solving of formulae with different number of steps was shown to be more effective than our naïve sequential approach Rintanen, Heljanko, and Niemelä 2006), the empirical results show that even this simple method is sufficient to outperform current temporally expressive planners. We leave the investigation regarding the effect of using such parallelism for our future research.

We have compared ITSAT-$\exists^*$ with POPF2 based on the number of problems they can solve in each domain and also by the total score given to each planner using the scoring strategy of recent IPCs: if a planner cannot solve a problem, it will get score 0 for it; Otherwise, its score is equal to the makespan of the best plan found by either of planners divided by the makespan of the plan found by the planner. In order to achieve a better assessment of ITSAT-$\exists^*$ in problems with required concurrency, we have also tested both planners in two extra domains (driverlogshift and matchlift), which were introduced by Strathclyde Planning Group and have this property.

As it is shown in Table 2, ITSAT-$\exists^*$ significantly outperforms POPF2 in both the total number of solved problems and the total score. In fact, ITSAT-$\exists^*$ solves 53 more problems than POPF2. Moreover, the total score of ITSAT is 32 percent higher than that of POPF2. These results show a major improvement in temporally expressive temporal planning. The only domains in which POPF2 has a considerable lead over ITSAT-$\exists^*$ are parking and elevators. These two domains are inherently difficult for SAT-based planners, as the number of actions that provide each preposition is relevantly high in their problems.

			solved		score	
domain	IPC	prob	ITSAT	POPF2	ITSAT	POPF2
zenotravel		20	13	13	**12.41**	11.46
driverlog	2004	20	15	15	**13.70**	11.84
rovers		20	**20**	19	**20**	13.91
depots		22	**13**	7	**12.84**	5.53
airport	2006	50	**37**	15	**37**	13.40
satellite		36	14	14	**13.35**	8.96
pegsol		20	**20**	19	**20**	18.62
crewplanning		20	20	20	19.11	**20**
openstacks		20	8	**20**	6.47	**20**
parking		20	1	**20**	0.53	**20**
elevators		20	0	**2**	0	**2**
floortile	2011	20	**20**	5	**18.78**	5
storage		20	**6**	0	**6**	0
matchcellar		20	20	20	20	20
sokoban		20	**4**	3	**4**	2.89
parcprinter		20	**20**	0	**20**	0
turnandopen		20	6	**9**	6	**8.52**
tms		20	**20**	4	**20**	4
driverlogshift	—	10	10	10	**10**	8.99
matchlift		14	**14**	13	**14**	12.06
total		482	**281**	228	**274.19**	207.18

Table 2. Comparing ITSAT-$\exists^*$ with POPF2

Conclusion

In this paper, we formally defined temporal versions of classical $\forall$-step and $\exists$-step semantics. We also showed that by separating the casual and temporal reasoning phases of a SAT-based temporal planner, one can employ these semantics to construct effective encodings. Two different types of $\exists$-step encodings were introduced for temporal planning. We embedded our new $\exists$-step encodings into ITSAT and empirically showed the new encodings to be more efficient than the $\forall$-step encoding employed previously in ITSAT. The resulting planner, ITSAT-$\exists^*$, significantly outperforms POPF2, which has been so far the state-of-the-art in temporally expressive planning.

References

Armando, A.; and Giunchiglia E. 1993. Embedding Complex Decision Procedures inside an Interactive Theorem Prover. *Annals of Mathematics and Articial Intelligence*, 8(34), 475502.

Biere, A. 2009. P{re,i}coSAT@SC'09. Solver description for SAT Competition 2009. *In SAT 2009 Competitive Event Booklet*.

Blum, A.; and Furst, M. 1997. Fast planning through planning graph analysis. *Artificial intelligence*. 90:281-300.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. *Proceedings of 20th International Conference on Automated Planning and Scheduling*, 42-49, AAAI press.

Cushing, W.; Kambhampati, S.; Mausam; and Weld, D. S. 2007. When is temporal planning really temporal? *Proceedings of 20th International Joint Conference on Artificial Intelligence*, 1852-1859, AAAI press.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal Constraint Networks. *Artificial Intelligence* 49(1-3): 61-95.

Eyerich, P.; Mattmuller, R.; and Roger, G. 2009. Unifying Context-Enhanced Additive Heuristic for Temporal and Nu-

meric Planning. *Proceedings of 19th International Conference on Automated Planning and Scheduling*, AAAI press.

Fox, M.; and Long, D. 2002. PDDL+: Modelling Continuous Time-dependent Effects. *In Proceedings of the Third International NASA Workshop on Planning and Scheduling for Space*.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20: 61-124.

Garrido, A.; Fox, M.; and Long, D. 2002. A temporal planning system for durative actions of PDDL2.1. *Proceedings of 15th European Conference on Artificial Intelligence*, 586-590, IOS press.

Huang, R.; Chen, Y.; and Zhang, W. 2009. An optimal temporally expressive planner: Initial results and application to P2P network optimization. *Proceedings of 19th International Conference on Automated Planning and Scheduling*, AAAI press.

Kautz H.; and Selman, B. 1992. Planning as Satisfiability. *Proceedings of 10th European Conference on Artificial Intelligence*, 359-363, IOS press.

Kautz, H.; Selman, B.; and Hoffmann, J. 2006. SatPlan: Planning as Satisfiability. *International Planning Competition*.

Mali, A. D.; and Liu, Y. 2006. T-SATPLAN: A SAT-based Temporal Planner. *International Journal of Artificial Intelligence Tools* 15(5): 779-802.

Rankooh, M. F.; Mahjoob, A.; and Ghassem-Sani, G. 2012. Using Satisfiability for Non-Optimal Temporal Planning. *Proceedings of the 13th European Conference on Logics in Artificial Intelligence*, Springer.

Rintanen, J.; Heljanko, K.; and Niemelä. 2006. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence*, 170(12-13): 1031-1080.

Shin, J.; and Davis, E. 2005. Processes and continuous change in a SAT-based planner. *Artificial Intelligence*, 166(1-2): 194-253.

Smith, D. E.; and Weld D. S. 1999. Temporal planning with mutual exclusion reasoning. *Proceedings of 16th International Joint Conference on Artificial Intelligence*, 326-337, AAAI press.

Vidal, V.; and Geffner, H. 2006. Branching and pruning: An optimal temporal POCL planner based on constraint programming. *Artificial Intelligence*, 170(3): 298-335.

Wehrle, M.; and Rintanen, J. 2007. Planning as Satisfiability with Relaxed $\exists$-step Plans. *Proceedings of 20th Australian Joint Conference on Artificial Intelligence*, 244-253, Springer-Verlag.

Younes, H. L. S.; and Simmons, R. G. 2003. VHPOP: Versatile Heuristic Partial Order Planner. *Journal of Artificial Intelligence Research*, 20: 405-430.