Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Models of Action Concurrency in Temporal Planning

Jussi Rintanen*
Department of Computer Science
Aalto University, Helsinki, Finland

Abstract

Models of temporal planning are complex, due to the possibility of multiple concurrent and mutually interacting actions. This work compares two modeling languages, one with a PDDL-style action exclusion mechanism, and another with an explicit notion of resources, and investigates their implications on constraint-based search. The first mechanism forces temporal gaps in action schedules and have a high performance penalty. The second mechanism avoids the gaps, with dramatically improved performance.

1 Introduction

Temporal planning has traditionally been viewed as action sequencing + scheduling. This close connection between planning and scheduling has been visible in the use of constraint-based methods in solving hard large-scale scheduling and resource allocation problems, and during the last decade also in solving temporal planning problems, for example with the satisfiability problem of the propositional logic (SAT) [Rankooh and Ghassem-Sani, 2013] and its extensions such as Satisfiability modulo Theories (SMT) [Shin and Davis, 2005; Maris and Régnier, 2008], or MILP [Dimopoulos and Gerevini, 2002]. Interestingly, in recent years researchers in temporal planning have been abandoning approaches that use one-off reductions to constraint or logic languages. A very recent exception to this is based on the NDL modeling language for which a discretization method for temporal planning was developed [Rintanen, 2015]. Most other recent works, using the standard PDDL modeling language [Fox and Long, 2003], split the temporal planning problem to first solving what is essentially a classical planning problem with all quantitative temporal information removed [Rankooh and Ghassem-Sani, 2013], followed by or interleaved with a separate scheduling phase to determine if all actions can be correctly scheduled. If the scheduling fails, the first phase is run again with additional constraints to rule out the same incorrect solution. No explanation has emerged why the more monolithic search methods that tightly integrate the action selection phase and the linear numeric constraint solving phase, such as MILP or SMT, have been mostly abandoned.

In this work, we investigate mechanisms for constraining actions' concurrency and co-occurrence employed in temporal modeling languages used by the AI planning community, and argue that they explain why MILP or SMT have seemed unattractive. Specifically, we observe that PDDL 2.1 [Fox and Long, 2003] induces temporal *gaps* between consecutive interdependent actions, and these gaps often induce twice the number of *steps* in the plans than what is necessary, with strong negative performance implications. The gaps are caused by the management of implicit resources as manipulation of state variables, for which the PDDL 2.1 definition introduces what is essentially a 0-duration *critical section* that can be utilized by at most one of conflicting actions at a time.

Our results provide new support to the idea that modeling languages and problem modelings have far more impact on the practical usability of planning technologies than what is widely recognized, and that some of the current modeling languages for temporal planning are not well suited for many of the leading methods for solving realistic planning and scheduling problems, including constraint programming, mixed integer linear programming (MILP), and SAT-based methods including SAT modulo Theories (SMT).

The structure of the work is as follows. Section 2 discusses representations of action exclusion in PDDL and an alternative language. Section 3 discusses the computational implications of the two models. Section 4 outlines encodings in SMT. Section 5 analyzes mappings between the two modeling languages. Section 6 experiments with them. We conclude by listing topics for future research.

2 Models

We use two temporal modeling languages that respectively use the action exclusion mechanism from PDDL 2.1 [Fox and Long, 2003] and explicit resources as formalized in the NDL language [Rintanen, 2015], with extensions to support our results in Section 5.

We assume a familiarity of the propositional logic. If x is a state variable, then x and $\neg x$ are literals. The *complement* $\overline{l}$ of literal l is defined by $\overline{x} = \neg x$ and $\overline{\neg x} = x$. We define states as valuations $v : X \to \{0, 1\}$ of the state variables. Executions of temporal plans are infinite state sequences on the

*Also affiliated with Griffith University, Brisbane, Australia, and the Helsinki Institute for Information Technology, Finland. This work was funded by the Academy of Finland (Finnish Centre of Excellence in Computational Inference Research COIN, 251170).

rational time line, represented by valuations $v(t, x)$ of state variables for rational numbers $t \geq 0$. As the definition of states coincides with the definition of models in the propositional logic, we denote by $v(t, \phi)$ the obvious generalization of the valuation to propositional formulas ϕ.

2.1 PDDL-Style Action Exclusions

An action consists of a *precondition* which determines whether an action can be taken at a given point of time, as far as the values of the state variables are concerned but ignoring other actions, and *effects* which determine how and when state variables change after the action has been taken.

Definition 1 (Actions) *Let X be a finite set of state variables. An action is a pair $\langle p, e\rangle$ where*

- *the precondition p is a propositional formula over X,*
- *the effect e is a set of pairs (t, l) where $t \geq 0$ is a rational number and l is a literal over X.*

The precondition has to be true when the action is taken, and effects (t, l) cause literals l to be true a duration t after the action was taken. Define *prec*$(\langle p, e\rangle) = p$ and *eff*$(\langle p, e\rangle) = e$.

We next give a formal definition of a simple modeling language that incorporates PDDL's action exclusion mechanism.

Definition 2 (Problem instance) *A problem instance in temporal planning is a quadruple $\langle X, I, A, G\rangle$ where*

- *X is a set of state variables,*
- *$I : X \to \{0, 1\}$ is the initial state, describing the initial values of state variables,*
- *A is a set of actions over X, and*
- *G is the* goal*, a propositional formula over X.*

Definition 3 (Plans and Executions) *Given a problem instance $\langle X, I, A, G\rangle$, a* plan *$\pi$ is a finite set of pairs (a, t), where $t \geq 0$ is a rational number and $a \in A$ is an action such that the following holds.*

1. *There is no $\{(a_1, t_1), (a_2, t_2)\} \subseteq \pi$ such that*
   (a) *x occurs in prec(a_1),*
   (b) *$(t, l) \in$ eff(a_2) for $l \in \{x, \neg x\}$, and*
   (c) *$t_1 = t_2 + t$.*
2. *There is an* execution *$v : \mathbb{Q} \times X \to \{0, 1\}$ which is a mapping from non-negative rational time points and state variables to 0 and 1 such that*
   (a) *$v(0, x) = I(x)$ for all $x \in X$,*
   (b) *if $(a, t) \in \pi$, then for some $\epsilon > 0$ and for all t' such that $t - \epsilon < t' < t$ we have $v(t', prec(a)) = 1$,*
   (c) *if $(a, t) \in \pi$ and $(t', l) \in$ eff(a), then $v(t + t', l) = 1$,*
   (d) *state variables not changed by actions retain their values: for any t_l and t_u such that $t_l < t_u$, if*
   - *$v(t_l, x) = 1$, and*
   - *there is no $(a, t') \in \pi$ such that $(t'', \neg x) \in$ eff(a) and $t_l \leq t' + t'' \leq t_u$*

   then $v(t_i, x) = 1$ for all t_i such that $t_l < t_i \leq t_u$. (Analogously for $v(t_l, x) = 0$.)
3. *There is t such that $v(t', G) = 1$ for all $t' > t$.*

Condition 1 is the essence of action exclusions in PDDL: two actions cannot be taken simultaneously if the precondition of one is falsified by the *start effect* of the other.

Prevention of Actions' Overlap

In this modeling, the only mechanism for explicitly expressing constraints on the co-occurrence of actions is the ban on two actions starting at *exactly the same time point* according to Condition 1. Not being *exactly* simultaneous may seem insufficient for modeling non-overlapping, but this condition should only be viewed as a mechanism for creating a *critical section* in which state variables can be manipulated in ways that allow expressing more complex conditions on the overlap of actions. In particular, the critical sections allow the exclusive allocation of an implicit *resource* so that the overlap of actions is prevented: to make two actions exclusive, both actions have the same precondition x and the same effect $(0, \neg x)$. Now the actions cannot be taken simultaneously, and since each action falsifies the precondition of the other, the second action cannot be taken before x is later made true.

Limits

The technically tricky part of the definition is the seeming simultaneity of the precondition x and the effect $(0, \neg x)$. When an action is taken at some time point t_0, the precondition x should be true at t_0 and the effect $(0, \neg x)$ should make x false at t_0. Since a state variable cannot really be true and false simultaneously, we must interpret the truth of x in the precondition at t_0 as the *limit* of truth of x as time approaches t_0: x is "true" at t, as far as the precondition at t is concerned, if x is true at every time point t' over some (possibly very short) interval $]t - \epsilon, t[$, where $\epsilon > 0$. This is what our formalization of the truth of the precondition in (2b) of Definition 3 says. In terms of finite sets of actions taking place at different time points, this means that x becomes true at or before $t - \epsilon$, and remains true until t.

2.2 Action Exclusions Based on Resources in NDL

Another model of exclusiveness of actions is *resources* [Le Pape, 1994], as used in NDL [Rintanen, 2015].

We formalize the resource-based action exclusion mechanism in NDL in order to compare it to the syntactic condition 1 in Definition 3. First we extend Definition 1 of actions by including an additional component, *resource requirements*.

Definition 4 (Actions with Resources) *Let X be a finite set of state variables and R a finite set of resources. An action is a triple $\langle p, q, e\rangle$ where*

- *the precondition p is a propositional formula over X,*
- *the* resource requirement *q is a set of*
  - *$(t^s, t^e, r) \subseteq \mathbb{Q} \times \mathbb{Q} \times R$ such that $t^s < t^e$, and*
  - *$(t^s, t^e, r, n) \subseteq \mathbb{Q} \times \mathbb{Q} \times R \times \mathbb{N}$ such that $t^s < t^e$, and*

- *the effect e is a set of pairs (t, l) where $t \geq 0$ is a rational number and l is a literal over X.*

We consider two types of resource requirements, *unary resources*, expressed by the triples (t^s, t^e, r), which model absolute exclusion inside a group of actions: at most one action at a time can allocate the resource r, as well as *state resources*, expressed by 4-tuples (t^s, t^e, r, n), where n is the state. Multiple actions can use the same state resource as long as the required state n is the same. Hence state resources induce exclusions between sets of actions.

Definition of plans and executions (Definition 3) mostly remain unchanged for our second model. The action exclusion condition is changed. Additionally, preconditions are evaluated at the starting point of actions, not in the limit. Define *prec*$(\langle p, q, e\rangle) = p$, *eff*$(\langle p, q, e\rangle) = e$ and *rreq*$(\langle p, q, e\rangle) = q$.

Definition 5 (Plans and Executions with Resources) *Given a problem instance $\langle X, R, I, A, G\rangle$, a* plan π *is a finite set of pairs (a, t), where $t \geq 0$ is a rational number and $a \in A$ is an action such that*

1. *for all $\{(t_1, a_1), (t_2, a_2)\} \subseteq \pi$ such that for some $r \in R$*
   - *$(t_1^s, t_1^e, r) \in$ rreq(a_1) and $(t_2^s, t_2^e, r) \in$ rreq(a_2), or*
   - *$(t_1^s, t_1^e, r, n_1) \in$ rreq(a_1) and $(t_2^s, t_2^e, r, n_2) \in$ rreq(a_2) such that $n_1 \neq n_2$*

   either

   (a) $t_1 + t_1^e \leq t_2 + t_2^s$, or

   (b) $t_2 + t_2^e \leq t_1 + t_1^s$.
2. *Exactly as in Definition 3, except for*

   (2b) $v(t, prec(a)) = 1$ for all $(a, t) \in \pi$

We do not allow a precondition ϕ to be simultaneous with effects that make ϕ true.

Resource requirements (t_0, t_1, v) and (t_0, t_1, v, n) with $t_0 < t_1$ are interpreted as open intervals $]t_0, t_1[$, and with $t_0 = t_1$ they are interpreted as closed intervals $[t_0, t_0]$. If we limited to durations > 0 only, it would suffice – from the point of view of allowing consecutive actions with a zero-duration gap between them – to have the intervals half-open. However, intervals $[t_0, t_0]$ have important uses, and it sometimes is desirable to allow them *between* two open intervals.

3 Implications to Scheduling of Actions

Now we discuss the pragmatics of the temporal models formalized in Definitions 1, 2 and 3, and in Definitions 4 and 5. Our primary interest is in the implications of these models on *constraint-based* methods for temporal planning.

3.1 Timed PDDL

The core issue with PDDL 2.1 is that the *start* effect $\neg x$ cannot possibly be *simultaneous* with the effect x of a preceding action, and hence there must be a gap of some non-zero duration between the actions.

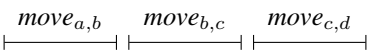

Figure 1: Gaps in Action Schedule

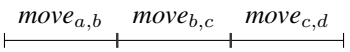

Figure 2: Action Schedule without Gaps

Example 1 *Consider the actions*

$$\begin{aligned} move_{a,b} &= \langle a, \{(0, \neg a), (2, b)\}\rangle \\ move_{b,c} &= \langle b, \{(0, \neg b), (2, c)\}\rangle \\ move_{c,d} &= \langle c, \{(0, \neg c), (2, d)\}\rangle \\ move_{a,e} &= \langle a, \{(0, \neg a), (2, e)\}\rangle \end{aligned}$$

which represent the movement of some (unnamed) object from location to location. Actions move$_{a,b}$ and move$_{a,e}$ are exclusive of each other because they both falsify the other's precondition as their first effect.

To move from a to d we need three actions. First move$_{a,b}$ is taken, starting at step 0 with effects at step 1. Then move$_{b,c}$ follows, starting at step 2 with effects at step 3, and the final action is move$_{c,d}$, starting at step 4 with effects at step 5. This is a total of 6 steps or explicitly represented states in the execution of the plan. These steps are for time points $t_0, \ldots, t_5$, satisfying $t_1 = t_0 + 2$, $t_2 > t_1$, $t_3 = t_2 + 2$, $t_4 > t_3$, $t_5 = t_4 + 2$. The gaps (see Figure 1) between t_1 and t_2, and t_3 and t_4 can be arbitrarily small.

3.2 Explicit Resources and NDL

When using the NDL modeling language (Definitions 4 and 5), the *at start* effects that lead to the gaps shown in Figure 1 can be postponed to the end of the actions, and the evaluation of the preconditions at the start of the action – and not through the limit construction – allows action schedules without gaps.

Example 2 *Consider the actions*

$$\begin{aligned} move_{a,b} &= \langle a, \{(0, 2, x)\}, \{(2, \neg a), (2, b)\}\rangle \\ move_{b,c} &= \langle b, \{(0, 2, x)\}, \{(2, \neg b), (2, c)\}\rangle \\ move_{c,d} &= \langle c, \{(0, 2, x)\}, \{(2, \neg c), (2, d)\}\rangle \\ move_{a,e} &= \langle a, \{(0, 2, x)\}, \{(2, \neg a), (2, e)\}\rangle \end{aligned}$$

Here x is a resource that could be intuitively associated with the object to be moved, allowing at most one move at a time.

With the NDL semantics, the action can be taken at the same time point where its precondition becomes true. Without the gaps the sequence of actions move$_{a,b}$, move$_{b,c}$, move$_{c,d}$ only requires 4 steps (Figure 2): the starting points of each of the three actions (with the end point of the action coinciding with the starting point of the next) followed by the end point of the last action.

In general, when n consecutive actions each have a precondition that depends on the previous action, the PDDL modeling requires $2n$ steps, whereas with explicit resources $n + 1$ steps suffice. As is well known from Planning as SAT [Kautz and Selman, 1996], the number of steps is one of the most critical factors in the scalability of SAT solving. Halving the number of steps can improve performance exponentially.

4 Encodings of Timed Actions in SMT

Temporal planning can be reduced [Shin and Davis, 2005] to constraint-based formalisms such as SAT modulo Theories (SMT) [Wolfman and Weld, 1999].

In both SAT and SMT encodings the propositional variables encode the values of the state variables in a finite number of steps, numbered $0, \ldots, T$. Here T is the maximum number of consecutive actions (steps) in the plan. To represent temporal planning finitely, only those time points and states are represented in which (non-continuous) change takes place. This includes changes in discrete (Boolean) state variables [Shin and Davis, 2005]. Hence the explicit states are a sequence $s_0, \ldots, s_T$ for time points $\tau_0 < \tau_1 < \cdots < \tau_T$.

The SMT representation for the sequence $s_0, \ldots, s_T$ consists of state variables x encoded as propositional variables $x@k$ for every $k \in \{0, \ldots, T\}$, together with numeric variables $\Delta_i = \tau_i - \tau_{i-1}, i \in \{1, \ldots, T\}$ representing the differences of the absolute time points of consecutive steps.

4.1 PDDL-Style Encodings with Limits

We will be using a simplified variant of Shin and Davis' [2005] encoding of temporal PDDL planning in SMT.[1]

In the following, we write $\phi@i$ for formulas ϕ with each state variable x replaced by $x@i$.

When an action is taken at step i, its precondition has to hold in the limit when approaching i. For discrete variables this limit equals the value of precondition in the preceding step. Hence preconditions p are encoded by

$$a@i \rightarrow p@(i-1). \tag{1}$$

If action a has an effect l at time t (relative to the starting time of the action), then there must be a time point with a time difference t to the starting time.

$$a@i \rightarrow \bigvee_{j=i}^{T} (\tau_j - \tau_i = t) \tag{2}$$

If action a can make a literal l true at time t, then a contributes to the possible causes of l at step i by including

$$\bigvee_{j=1}^{i} (a@j \wedge (\tau_i - \tau_j = t)) \tag{3}$$

as one disjunct in the formula $\mathit{causes}_i(l)$ (with other disjuncts contributed by other actions.) Empty disjunctions are taken as constants *false*. The formula $\mathit{causes}_i(l)$ is used for expressing when a literal becomes true.

$$\mathit{causes}_i(l) \rightarrow l@i \tag{4}$$

The formulas $\mathit{causes}_i(l)$ are also used in *frame axioms* to indicate the conditions under which the value of a state variable is the same in two consecutive steps.

$$(\bar{l}@(i-1) \wedge l@i) \rightarrow \mathit{causes}_i(l) \tag{5}$$

The time that passes at each step has to be positive,

$$\Delta_i > 0 \tag{6}$$

[1] Since we do not use continuous variables, we can identify Shin and Davis' $Q[T_{i-1}^{+}]$ with $Q[T_i^{+}]$.

and the Δ and τ variables are related in the obvious way.

$$\tau_i - \tau_{i-1} = \Delta_i \tag{7}$$

For every pair of actions a_1 and a_2 that should be exclusive because one falsifies the precondition of the other as its time 0 effect, we have

$$\neg a_1@i \vee \neg a_2@i. \tag{8}$$

4.2 Limit-Free NDL Encodings with Resources

With the resource-based NDL language we can avoid the expression of the critical sections and limits that are needed for allocation of implicit resources in PDDL.

The encoding of the resource-based model is as above with the following modifications [Rintanen, 2015]. First, formula (8) is removed, as action exclusions are not based on the critical sections any more. Second, preconditions are evaluated in the starting step of an action instead of in the limit (represented by the preceding step). Hence we have

$$a@i \rightarrow p@i \tag{9}$$

replacing (1). Third, we have additional formulas for representing resources, as described next.

Let actions a_1 and a_2 have conflicting resource needs respectively for intervals $]t_0, t_1[$ and $]t_2, t_3[$. If time t has passed since taking the first action, then for the second action to be possible at time 0, the following must hold.

$$t + t_1 \leq t_2 \text{ or } t_3 \leq t + t_0 \tag{10}$$

If intervals overlap, the actions cannot be started at the same step, encoded as the obvious binary mutexes

$$\neg a_1@i \vee \neg a_2@i. \tag{11}$$

When action a_1 has been taken earlier, we use (10) to derive a constraint to prevent a_2. The constraint is obtained as a disjunction that iterates over all past steps and tests whether the time t in (10) has passed since taking a_1.

$$\begin{array}{l} a_2@i \rightarrow \bigvee_{j=1}^{i-1} (a_1@j \rightarrow (t_2 \geq t_1 + (\tau@i - \tau@j) \\ \qquad\qquad \vee t_0 + (\tau@i - \tau@j) \geq t_3)) \end{array} \tag{12}$$

This formula says that if an action is taken, then its need for the resource starts after the need by another action ends, or its need ends before the need by another action starts. [2]

5 Translating PDDL into NDL

We have developed a semi-automated translation process from PDDL into NDL. Theorem 1 illustrates many of the core issues, and allows partial translations for many of the standard benchmark problems, fully automated translation for Sokoban, and translations of most actions for Crewplanning and Elevators. Actions with PDDL's *over all* conditions require the use of state resources: the *over all* condition is replaced by the allocation of a state resource in state 1 for the action's duration, and any effect in other actions violating

[2] Note that there may be $\mathcal{O}(n^2)$ formulas (12) for n actions. In many important cases $\mathcal{O}(n)$ size encodings of resource constraints are possible, but these encodings are out of the scope of this work.

the condition is extended with the allocation of the same state resource in state 0 for a single time point.

Theorem 1 has been expressed in terms of two NDL models that differ in some effects having been delayed from time point 0 to the last time point of the action. This is how PDDL's *at start* effects can be turned into *at end* effects if resource constraints can be used to make sure the change does not invalidate existing plans, provided that no new plans are enabled that incorrectly exploit the changed action definition.

We outline the process next. Let the action have an effect e at time 0. Let $a_1, \ldots, a_n$ be all actions that depend on e, that is, their preconditions share a state variable with e. If all of $a_1, \ldots, a_n$ are mutually exclusive (mutex) with a, then the effect e can be moved from 0 to the action's end time t^a_{end}. By two actions being mutex we mean that resources or system properties [Rintanen, 2014] guarantee that the interval between the starting and ending points of the actions cannot overlap.

The exclusivity test is split into two parts. First, check that no action violates the exclusivity condition. Second, when the exclusivity condition otherwise holds, but would be violated by moving the effect, then the exclusivity has to be restored. We will discuss these next.

Consider an action a with effect $\neg x$ at time 0. Hence x is (possibly) true until 0 and false after 0. Moving $\neg x$ from 0 to t^a_{end} could affect possible plans in three ways.

1. Actions depending on x could possibly be taken between 0 and t^a_{end} where they otherwise could not be taken.
2. Actions depending on $\neg x$ cannot be taken between 0 and t^a_{end} where they otherwise could be taken.
3. Actions with effect x scheduled between the start and end of a could have their effect overridden if $\neg x$ is moved from 0 to t^a_{end}.

For the move to not produce new plans that are not valid according to the original problem description, taking actions of type 1 must be prevented. For the move to not invalidate existing plans, actions of types 2 and 3 must not exist.

The theorem presents a test for detecting actions of type 2 and 3 (preventing the move), and derives new constraints for forbidding actions of type 1 in the modified problem instance.

The theorem is given with $\neg x$ as the effect to be moved, but it obviously holds when the roles of x and $\neg x$ are reversed.

Theorem 1 *Let P be a NDL problem instance with actions A. Let P' be an NDL problem instance obtained from P by delaying effects $(0, l)$ to the end of the action by the following procedure. Let a be any action such that*

1. *$\neg x$ is an effect of a at 0,*
2. *x is not an effect of a at any $t' > 0$,*
3. *the last effects of a are at t^a_{end},*
4. *a is mutex with every action that makes x true (and consequently, no other action can make x true between 0 and t^a_{end}),*
5. *a is mutex with every action with precondition $\neg x$ (and consequently, no action depends on $\neg x$ between 0 and t^a_{end}).*[3]

Action a is replaced by a modified action obtained by

1. *moving effect $\neg x$ from 0 to t^a_{end},*
2. *adding resource requirement $(0, t^a_{end}, R^x, 0)$,*
3. *adding resource requirement $(0, 0, R^x, 1)$ to any action in $A - \{a\}$ with x in the precondition, and*
4. *adding resource requirement $(t, t, R^x, 1)$ to any action in $A - \{a\}$ with effect (t, x).*

Then any plan for P is also a plan for P', and any plan for P' is also a plan for P.

Proof: Let π be any plan for P. Define $P_0 = P$, and let $P_, \ldots, P_n$ be NDL instances such that each $P_i, i > 0$ is obtained from P_{i-1} by modifying one of the actions. Our proof shows that each P_i preserved the plans of P_{i-1}. We obtain P' as the last instance P_n in the sequence generated by modifying n actions.

Let π be any plan for P_{i-1} with an occurrence of the action a that is modified. We show that π is also a plan for P_i.

- We show that same states will be generated. The difference between the two cases is that the effect $\neg x$ of a is moved from 0 to t^a_{end}. Until the starting point of a and starting from the end point of a the execution is the same in both cases. The changes possibly caused simultaneously by other actions are not affected by the postponement of $\neg x$. In particular, since no other action that could overlap with a can make x true (assumption 4), no effect of an overlapping action is overridden by the postponed $\neg x$.
- We show that all preconditions are satisfied. Again, the only change is postponing $\neg x$ from 0 to t^a_{end} in action a. Hence by assumption 5 there are no actions in π with a precondition that turns false when $\neg x$ is delayed to t^a_{end}.
- We show that the additional resource requirements in P_i are not violated. Since in π no action with x in the precondition starts (relative to the starting point of a) between 0 and t^a_{end} (because x is false there), the additional resource requirements in P_i (which only concern this case) is satisfied.

For the proof in the other direction, assume that π is a plan for P_i. We show that π is also a plan for P_{i-1}.

- We show that same states will be generated. The difference between the two cases is that the effect $\neg x$ of a is moved from t^a_{end} to 0. Until 0 and starting from t^a_{end} the execution is the same in both cases.

  Resource requirement 4 guarantees that no action makes x true between 0 and t^a_{end}. Hence the state at the end of action a with P_{i-1} and P_i is the same.
- We show that all preconditions are satisfied. Again, the only change is move of $\neg x$ from t^a_{end} to 0 in action a. The

[3] For many of the standard benchmarks a weaker condition suffices: $\neg x$ is not in the precondition of any action in $A - \{a\}$.

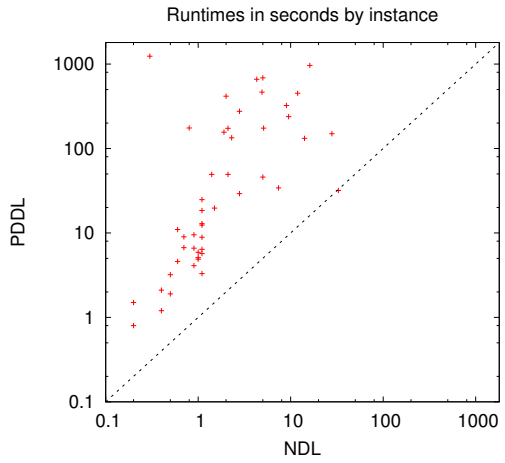

Figure 3: Runtime comparison by instance

resource requirements 2 and 3 guarantee that no action with precondition x is taken between 0 and t^a_{end}. Hence every precondition is satisfied for P_{i-1} if every precondition is satisfied for P_i.

- Resource requirements for P_{i-1} are not violated because the requirements for P_{i-1} is a subset of those of P_i, and the latter are satisfied by assumption. □

Conditions of Theorem Theorem 1 explicitly rule out problems with *required concurrency* [Cushing *et al.*, 2007]. Coles et al. [2009] show that *compression-safe* temporal actions can be replaced by classical ones. The conditions for compression-safety are distantly related to the conditions of Theorem 1, but our modeling language is different, our conditions more general, and our theorem is about mapping temporal actions to equivalent temporal actions, not to classical ones.

6 Experiments

We compare PDDL models to their NDL translations with problems from the 2008 and 2011 planning competitions.[4] We translated all PDDL models into NDL manually, although some of these translations, or parts of them, would have been possible to automate with Theorem 1.

We implemented fully automatic translations from PDDL 2.1 and NDL into SMT, and solved the benchmark problems with the Z3 SMT solver on a workstation cluster mostly consisting of mid-range Intel Xeon CPUs from about 2010 to 2012. The runtime per instance was limited to 30 minutes. For each instance, we generated SMT encodings for $3n$ steps, for all $n > 0$, and solved them one by one until a satisfiable SMT instance (corresponding to a plan) was encountered.

Figure 3 illustrates the performance improvements of NDL over PDDL, with a plot of runtimes with both models for instances solved in under 30 minutes. The typical performance gain is between one and three orders of magnitude. The numbers of instances solved in 30 minutes are given in Table 1. The weighted score is the sum of the fractions of instances solved for each domain.

		NDL	PDDL
2008-crewplanning	30	10	4
2008-elevators	30	4	0
2008-openstacks-adl	31	2	0
2008-pegsol	30	30	28
2008-sokoban	12	5	1
2008-transport	7	0	0
2011-floortile	20	5	0
2011-matchcellar	10	5	3
2011-parking	17	7	3
2011-storage	11	0	0
2011-tms	20	8	7
2011-turnandopen	20	10	4
total	238	86	50
weighted score	13	4.01	2.17

Table 1: Number of solved instances

Rintanen [2015] reports further performance gains for NDL models obtained by discretizing the rational time line to integers. Interestingly, reformulating PDDL models into NDL improves performance roughly equally as the discretization of NDL models.

7 Related Work

Earlier research on temporal planning has not identified PDDL modeling to be a performance issue. This has probably been because of the general focus of the research area on explicit state-space search (forward-chaining) rather than constraint-based methods [Kautz and Selman, 1996] or other search methods including backward-chaining [Rintanen, 2008], and due to the research area taking existing problem modelings and modeling languages as given.

Approaches based on specialized algorithms for search with temporal states [Do and Kambhampati, 2003] and action schedules [Gerevini *et al.*, 2010] do not suffer from the gaps the way all-encompassing reductions to SAT, SMT, MILP or CP with the leading encodings (as in Section 4) do. In these approaches, the requirement of having at least an ϵ gap preceding a precondition falsified by the same action's *at start* effects is merely an additional temporal constraint for action scheduling, with little or no performance implications. In contrast, in the state-of-the-art constraint based models of temporal planning, the gaps often double the number of steps in the planner, resulting in an exponential performance penalty, as demonstrated in Section 6.

Some recent constraint-based planners have avoided the issue with gaps by abstracting the temporal planning problem to a classical one, and performing action scheduling separately. The work on ITSAT is a prominent example of this category [Rankooh and Ghassem-Sani, 2013]. ITSAT, however, is incapable of effectively minimizing plan duration, as temporal information is ignored during search.

[4] Except for the rather complicated PARC Printer problem which we did not understand deeply enough to model it in NDL.

8 Conclusion

We have investigated the impact of problem modelings on the scalability of constraint-based planning methods. Our main finding is that problem modelings based on explicit manipulation of state variables for resource management lead to SMT encodings that are harder to solve than those based on an explicit notion of resources.

Promising directions for future research include partial-order methods which have earlier been very successful for classical planning [Rintanen *et al.*, 2004; Wehrle and Rintanen, 2007; Robinson *et al.*, 2009]. Planning-specific implementation technologies for SAT-solving [Rintanen, 2012] are obviously applicable also to SMT-based temporal planning.

References

[Coles *et al.*, 2009] Amanda Jane Coles, Andrew Coles, Maria Fox, and Derek Long. Extending the use of inference in temporal planning as forwards search. In *ICAPS 2009. Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling*, pages 66–73. AAAI Press, 2009.

[Cushing *et al.*, 2007] William Cushing, Subbarao Kambhampati, Mausam, and Daniel S. Weld. When is temporal planning really temporal. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1852–1859. AAAI Press, 2007.

[Dimopoulos and Gerevini, 2002] Yannis Dimopoulos and Alfonso Gerevini. Temporal planning through mixed integer programming: A preliminary report. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, volume 2470 of *Lecture Notes in Computer Science*, pages 47–62. Springer-Verlag, 2002.

[Do and Kambhampati, 2003] Minh Binh Do and Subbarao Kambhampati. Sapa: A multi-objective metric temporal planner. *Journal of Artificial Intelligence Research*, 20:155–194, 2003.

[Fox and Long, 2003] Maria Fox and Derek Long. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[Gerevini *et al.*, 2010] Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. Temporal planning with problems requiring concurrency through action graphs and local search. In *ICAPS 2010. Proceedings of the Twentieth International Conference on Automated Planning and Scheduling*, pages 226–229. AAAI Press, 2010.

[Kautz and Selman, 1996] Henry Kautz and Bart Selman. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of the 13th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference*, pages 1194–1201. AAAI Press, 1996.

[Le Pape, 1994] Claude Le Pape. Implementation of resource constraints in ILOG Schedule: A library for the development of constraint-based scheduling systems. *Intelligent Systems Engineering*, 3(2):55–66, 1994.

[Maris and Régnier, 2008] Frederic Maris and Pierre Régnier. TLP-GP: New results on temporally-expressive planning benchmarks. In *Tools with Artificial Intelligence, 2008. ICTAI'08. 20th IEEE International Conference on*, volume 1, pages 507–514. IEEE, 2008.

[Rankooh and Ghassem-Sani, 2013] Masood Feyzbakhsh Rankooh and Gholamreza Ghassem-Sani. New encoding methods for SAT-based temporal planning. In *ICAPS 2013. Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling*, pages 73–81. AAAI Press, 2013.

[Rintanen *et al.*, 2004] Jussi Rintanen, Keijo Heljanko, and Ilkka Niemelä. Parallel encodings of classical planning as satisfiability. In *Logics in Artificial Intelligence: 9th European Conference, JELIA 2004*, number 3229 in Lecture Notes in Computer Science, pages 307–319. Springer-Verlag, 2004.

[Rintanen, 2008] Jussi Rintanen. Regression for classical and nondeterministic planning. In *ECAI 2008. Proceedings of the 18th European Conference on Artificial Intelligence*, pages 568–571. IOS Press, 2008.

[Rintanen, 2012] Jussi Rintanen. Engineering efficient planners with SAT. In *ECAI 2014. Proceedings of the 21st European Conference on Artificial Intelligence*, pages 684–689. IOS Press, 2012.

[Rintanen, 2014] Jussi Rintanen. Constraint-based algorithm for computing temporal invariants. In *Logics in Artificial Intelligence, 14th European Conference, JELIA 2014*, volume 8761 of *Lecture Notes in Computer Science*, pages 665–673. Springer-Verlag, 2014.

[Rintanen, 2015] Jussi Rintanen. Discretization of temporal models with application to planning with SMT. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15)*, pages 3349–3355. AAAI Press, 2015.

[Robinson *et al.*, 2009] Nathan Robinson, Charles Gretton, Duc-Nghia Pham, and Abdul Sattar. SAT-based parallel planning using a split representation of actions. In *ICAPS 2009. Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling*, pages 281–288. AAAI Press, 2009.

[Shin and Davis, 2005] Ji-Ae Shin and Ernest Davis. Processes and continuous change in a SAT-based planner. *Artificial Intelligence*, 166(1):194–253, 2005.

[Wehrle and Rintanen, 2007] Martin Wehrle and Jussi Rintanen. Planning as satisfiability with relaxed $\exists$-step plans. In *AI 2007 : Advances in Artificial Intelligence: 20th Australian Joint Conference on Artificial Intelligence*, number 4830 in Lecture Notes in Computer Science, pages 244–253. Springer-Verlag, 2007.

[Wolfman and Weld, 1999] Steven A. Wolfman and Daniel S. Weld. The LPSAT engine & its application to resource planning. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 310–315. Morgan Kaufmann Publishers, 1999.