Temporal Planning with Clock-Based SMT Encodings

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

Abstract

We propose more scalable encodings of temporal planning in SMT. The first contribution is practical clock-based encodings of resources and effect delays. Existing encodings of effect delays (Shin and Davis, 2015) have a quadratic size, due to the necessity to determine the time differences between steps for a linear number of steps. Clocks improve this to linear. The second contribution is a new relaxed scheme for steps. Existing schemes require a step for every time point with discontinuous change. This is relaxed, improving scalability.

1 Introduction

After the successes of SAT in solving the classical planning problem [Kautz and Selman, 1996; 1999], Shin and Davis [2005] proposed the encodings of temporal and hybrid systems planning in the SAT modulo Theories (SMT) framework. While the work solves conceptual problems of temporal planning with SMT, it has not proved successful in terms of performance. Shin & Davis adopted a temporal model with ϵ-separation [Fox and Long, 2003], which can double the number of *steps*, with a high performance penalty for constraint-based methods [Rintanen, 2015b]. ITSAT [Rankooh and Ghassem-Sani, 2015] – probably the best scalable temporal planner – avoids the problems of ϵ-separation by reducing the temporal problem to a non-temporal one. ITSAT, however, often generates plans with makespans twice the optimal, which in practice is unacceptable. Rintanen [2015a] adopts a temporal model without ϵ-separation and with opportunities for discretization to integer time, which is not possible with ϵ-separation.

The first application of SMT was classical planning with numeric variables [Wolfman and Weld, 1999]. SMT was little used before the recent works on temporal planning, and very recently on continuous change [Bryce *et al.*, 2015].

*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). We acknowledge the computational resources provided by the Aalto Science-IT project.

In this work, we pursue the fundamentals of temporal (and hybrid) planning further, believing that full temporal model is needed *during search* to achieve good-quality plans. First, we address the number of steps in SMT encodings, analogous to the number of time points in SAT encodings of classical planning [Kautz and Selman, 1996]. Shin and Davis [2005] require a step for every time point in which an action or a discrete change takes place. We propose a relaxed scheme in which one step can summarize discrete changes in multiple preceding time points. Second, we address the *asymptotic size* of encodings, reducing a encoding of *time delays* from quadratic to linear, while avoiding the excessive number of clocks of earlier encodings [Rintanen, 2015a].

The resulting encodings improve earlier state of the art in SMT-based temporal planning, solving dozens more of the standard benchmark problems, and sometimes improving runtimes by two orders of magnitude.

2 Model of Temporal Planning

We adopt the formal framework of Rintanen [2015a]. A *temporal action* consists of a *precondition* ϕ (a propositional formula), and *effects* which are negated or unnegated state variables associated with times ≥ 0, indicating how much after the action the effect will take place. When the action is at absolute time t, then an effect for time t' is at absolute time $t+t'$. We call t' the *delay*. An action can allocate a *resource* for a duration d_0, at some time t_0 relative to the starting point of the action: the resource is allocated for the interval $t+t_0$ to $t+t_0+d_0$, and no other action can take place if it allocates the same resource for an intersecting interval.

Definition 1 (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 tuples*
  - *$(t^s,t^d,r)\subseteq\mathbb{Q}\times\mathbb{Q}\times R$ such that $t^d\geq 0$, and*
  - *$(t^s,t^d,r,n)\subseteq\mathbb{Q}\times\mathbb{Q}\times R\times\mathbb{N}$ such that $t^d\geq 0$, 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 refer to the precondition p of an action a by *prec*(a), the resource requirement q by *rreq*(a), and the effects e by

eff(a). The set R of resources is divided in two types. *Unary resources*, expressed by triples (t^s,t^d,r), model absolute exclusion inside a group of actions: only one action can allocate r at a time. *State resources*, expressed by tuples (t^s,t^d,r,n), where n is the state, can be used by multiple actions as long as the state n is the same.

Definition 2 (Plans and Executions with Resources) *For a problem instance $\langle X,R,I,A,G\rangle$, with state variables X, resources R, initial state I, actions A, and goal G, a* plan π *is a finite set of pairs (a,t) such that the following holds.*

1. *$t\geq 0$ is a rational number and $a\in A$ is an action.*
2. *For all $\{(t_1,a_1),(t_2,a_2)\}\subseteq\pi$ such that for some $r\in R$*
   - *$(t^s_1,t^d_1,r)\in$ rreq(a_1) and $(t^s_2,t^d_2,r)\in$ rreq(a_2), or*
   - *$(t^s_1,t^d_1,r,n_1)\in$ rreq(a_1), $(t^s_2,t^d_2,r,n_2)\in$ rreq(a_2), and $n_1\neq n_2$,*

   either
   (a) $t_1+t^s_1+t^d_1\leq t_2+t^s_2$, or
   (b) $t_2+t^s_2+t^d_2\leq t_1+t^s_1$.
3. *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) $v(t,$prec$(a))=1$ for all $(a,t)\in\pi$
   (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 $t_l<t'+t''\leq t_u$*

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

Effects $(0,l)$ that make preconditions of simultaneous actions true (including the action itself) are not allowed.

3 SMT Encodings of Temporal Planning

Timelines in temporal planning are continuous, but it is sufficient to represent explicitly only a finite sequence of time points in which an action starts or an effect takes place, called *steps*. For Boolean state variables $X=\{x_1,\ldots,x_n\}$, actions $A=\{a_1,\ldots,a_m\}$, and $N+1$ steps, we need SMT variables $x@i$ and $a@i$ for all $x\in X$, $a\in A$, and $i\in\{0,\ldots,N\}$. For each step, these SMT variables indicate the values of state variables and whether an action is taken. Variables $\tau@i$ denote the absolute time at step i, and $\Delta@i=\tau@i-\tau@(i-1)$. We constrain these values by $\Delta@i>0$.

If ϕ is the precondition of action a, we have the formula

$$a@i\to\phi@i \tag{1}$$

where $\phi@i$ is the formula obtained from ϕ by replacing each x by $x@i$. By *causes*$(x)@i$ we denote the disjunction of all the conditions under which x becomes true at step i. These formulas typically refer to atomic propositions for earlier steps. Similarly *causes*$(\neg x)@i$ for x becoming false. Hence when *causes*$(l)@i$ is true, the effect l takes place.

$$\textit{causes}(x)@i\to x@i \tag{2}$$
$$\textit{causes}(\neg x)@i\to\neg x@i \tag{3}$$

Frame axioms allow inferring that the value of a state variable remains unchanged.

$$(x@i\wedge\neg x@(i-1))\to\textit{causes}(x)@i \tag{4}$$
$$(\neg x@i\wedge x@(i-1))\to\textit{causes}(\neg x)@i \tag{5}$$

How the disjuncts of *causes*$(x)@i$ are defined depends on when the action causing the change takes place, and how the time difference between the action and the effect is expressed. Here we first present the Shin&Davis style encoding of delay, and will consider other options later.

A disjunct of *causes*$(x)@i$ for effects x at a relative time $t>0$ of an action a is

$$\bigvee_{j=0}^{i-1}(a@j\wedge((\tau@i-\tau@j)=t)) \tag{6}$$

if encoded in the spirit of Shin and Davis' [2005] formula (4.3). These constraints have a quadratic size. When $t>0$, to guarantee that there is a step for every effect, we need

$$a@i\to\bigvee_{j=i+1}^{N}(\tau@j-\tau@i=t). \tag{7}$$

3.1 Resource Constraints

Consider actions a_1 and a_2 such that $(t_1,d_1,r)\in$ rreq(a_1) and $(t_2,d_2,r)\in$ rreq(a_2). If the intervals $]t_1,t_1+d_1[$ and $]t_2,t_2+d_2[$ overlap, then the actions cannot be at the same step. The following constraint prevents these two actions from starting at the same step.

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

Now consider the case in which a_1 may have been taken earlier at time $0-t$, and this may prevent a_2 from taken at the current step at time 0. The requirement that the allocations of the resource by the two actions do not overlap means that the relative time $-t$ where the action a_1 may have been taken satisfies one of the following.

$$0-t+t_1+d_1\leq t_2$$
$$0-t+t_1\geq t_2+d_2$$

This means that either the first action frees the resource before the second allocates it, or vice versa. Simplified these constraints are as follows.

$$t_1+d_1-t_2\leq t$$
$$t_1-t_2-d_2\geq t$$

Hence a_2 is not allowed if a_1 is at $]-t_1+t_2+d_2,-t_1-d_1+t_2[$ relative to the current time point.

$$a_2@i\to\neg\bigvee_{j=0}^{i-1}\left(a_1@j\wedge(t_1-t_2-d_2>\Delta^i_j>t_1+d_1-t_2)\right) \tag{9}$$

Here $\Delta^i_j = \sum_{k=j+1}^{i} \Delta@k$, or, equivalently, $\Delta^i_j = \tau@i - \tau@j$. If $t_1 < t_2 + d_2$, then testing the bound $t_1 - t_2 - d_2$ is unnecessary, because no matter how recently the first action was taken, that action can never allocate the resource after a_2 freed it. Similarly, if $t_2 \geq t_1 + d_1$, the earlier action must already have freed the resource before the second action could allocate, requiring no constraints.

Constraints on state resources are similar. A constraint for a pair of actions is required only if the actions allocate the same state resource with different states.

In all of the above constraints, a reference to a single action (like $a_1@i$) can be replaced by a disjunction of actions (like $a_1@i \vee a_2@i \vee \cdots \vee a_n@i$), when all these actions allocate the same resource for the same interval (and state.)

3.2 Effect Delays with Clocks

Rintanen [2015a] devised clock-based encodings of delays, and then pointed out that there will be impractically many real-valued variables, hampering efficient SMT solving. Here we briefly explain the use of clocks, and in Section 6 we propose a practical scheme for sharing clocks between actions.

The values of a clock c at different steps i are represented by SMT variables $c@i$. A clock c associated with action a is initialized to zero when the action is taken

$$a@i \rightarrow (c@i = 0) \tag{10}$$

and the value of the clock is increased at all other steps. Here $i \in \{1,\ldots,N\}$.

$$\neg a@i \rightarrow (c@i = c@(i-1) + \Delta@i) \tag{11}$$

Trigger for effect x with delay t in $\mathit{causes}(x)@i$ is now[1]

$$c@i = t. \tag{12}$$

To guarantee that there is a step where (12) is true, we need

$$(c@(i-1) < t) \rightarrow (c@i \leq t). \tag{13}$$

One clock per action can be insufficient if an action can overlap itself. Self-overlap is atypical, but when it is required, there are three options. In specific situations one clock can still suffice (see Section 5). If an action can self-overlap only a bounded number of times, a bounded number of clocks is sufficient. In case of unbounded self-overlap [Rintanen, 2007], a fall-back position would be to use a clock-free scheme like that of Shin and Davis.

4 Summarized Effects

A main problem with the encodings is the high number of steps, which increases solver runtimes. In the above encodings, there must be a step for actions' starting points, and also for any time point where a state variable changes.

Example 1 *Consider simultaneously taking three actions respectively with durations 1, 2, and 3, and respectively with effects x, y and z, to reach the goal $x \wedge y \wedge z$. With all encodings considered before, four steps are needed (see Figure 1, left): the starting step of all actions, and steps for the relative time points 1, 2, and 3, in which respectively x, y and z become true, and with the goal $x \wedge y \wedge z$ true at the last step.*

[1] If an action can immediately follow itself, here $c@(i-1) + \Delta@i$ must replace $c@i$. Same fix is later needed in (14), (15) and (16).

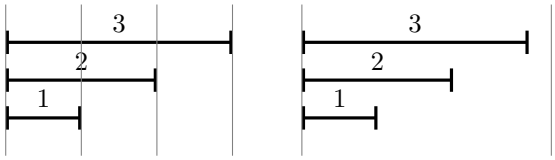

Figure 1: Steps for effects are not needed

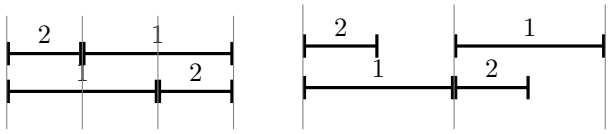

Figure 2: Plan with fewer steps may have a longer makespan

We propose a new scheme, in which a step is not necessary for time points with change, and effect axioms are relaxed to force an effect at a given step if its time is greater than or equal to the change time. Essentially, the infinitely dense line of real or rational time points has to be made explicit as steps only at those time points in which an action is started, or before such effect takes place that contradicts an earlier effect which has not been recorded at a step yet.

In the above example, this scheme allows a 2-step plan in which all three effects, respectively for time points 1, 2 and 3, are recorded at the second step, associated with any time point ≥ 3, as illustrated in Figure 1, right.

This scheme is easy to implement as a modification of the clock-based encoding of effect delays presented earlier. We only need in $\mathit{causes}(x)@i$ for effects (l,t) the formula

$$(c@i \geq t) \wedge (c@(i-1) < t), \tag{14}$$

relaxing and replacing (12), and we don't need formula (13).

This scheme creates new trade-offs between the makespans and the number of steps, as illustrated in Figure 2.

The relaxed scheme may resemble the idea of *parallel plans* in classical planning [Kautz and Selman, 1996; Rintanen *et al.*, 2006; Wehrle and Rintanen, 2007], where multiple actions are allowed at the same step whenever they are independent and can be ordered to a total order. Our scheme allows merging multiple steps, but still assumes a (non-strict) total ordering on the starting points of actions.

5 Clocks for Resource Constraints

We next present encodings of resource constraints expressed in terms of clocks. Encodings of resource constraints presented earlier are based on pairwise exclusions of pairs of actions or pairs of action sets. This types of encodings can be quadratic in the worst case, although with many temporal planning problems they can be improved to close to linear by lumping together all actions that allocate a given resource for the same (relative) time interval.

Our first observation is that clocks help achieve $\mathcal{O}(n)$ size encodings, as opposed to quadratic size $\mathcal{O}(n^2)$ as in formula (9). As the number of resources is low (some dozens for standard benchmarks), associating the clocks with them is much more practical than associating them with actions.

The core idea is that clocks can express resource allocations through their value in two different ways. First, a clock value in the interval $]t,t+d[$ denotes the allocation of the corresponding resource for a fixed duration d starting from a relative time point t counted from the resetting of the clock to 0 at the step where the action is taken. Second, a clock value <0 can denote an allocation of any duration extending from the current time point (and earlier) until the clock reaches value 0. In this case, an action allocating the resource for duration d from its beginning resets the clock to value $-d$. The second option covers almost all resources in standard temporal planning benchmark problems, which may allocate resources for different durations but in almost all cases do it from the beginning of an action.

Shared clocks cannot always accurately represent resource allocations. Allocations of the same state resource by different actions for different durations cannot be handled by one clock. Similarly for other resources, in specific cases. When clocks cannot be used, the fall-back strategy is to use the formulas (9).

Example 2 *Consider an action that allocates a unary resource for interval* $]9,10[$. *Assume this action does not use any other resource. Assume this action is taken at time points 0 and 1. The resource clock is reset to 0 at both time points. Now the value of the clock from time point 1 on only represents the second instance of this action. Taking an action that requires the same resource for the relative interval* $]0,1[$ *would be possible at time 9, which is not correct.*

Example 3 *If all actions that allocate a given resource allocate it for interval* $]9,10[$*, then all resource conflicts can be detected by using one clock. Assume an action is taken at 0. By checking the clock, it can be determined that a second instance of the same action is not possible before time 1, at which point the clock is reset.*

An important special case, which covers almost all resource conflicts in almost all of the standard benchmark problems for temporal planning, is the following. Let all actions allocate a resource at the action's starting point (relative time 0), for durations d_a, dependent on the action a. The resource clock c is reset to $-d_a$ when action a is taken. An action causes a resource conflict at a given time point iff $c\leq 0$.

Next we set out to devise a general encoding scheme which uses clocks whenever possible and handles all cases correctly. We consider two cases.

1. If at least one action allocates a resource r at relative time 0 (the starting point of the action), use clock c_r^0.

   If action a allocates r with $(0,d,r)$, then at the starting point of the action the clock c_r^0 is reset to $-d$.
2. For any other allocation (t,d,r) (with $t>0$) occurring at least once, use clock $c_r^{t,d}$.

   Any action allocating r with (t,d,r) resets $c_r^{t,d}$ to 0.

Next we discuss the conditions under which resource conflicts between actions can be detected by using the clocks associated with the resource allocations. Note that constraints on actions taken at the same step are always handled by static constraints as described earlier, and here we only address constraints on actions taken at different steps.

Consider actions a_1 and a_2 (this includes the case $a_1=a_2$) which respectively allocate the same unary resource r with $(0,d_1,r)$ and (t_2,d_2,r). The constraint for handling conflicts of this form is the following.

$$a_2@i\rightarrow\big(-c_r^0@i\leq t_2\big) \tag{15}$$

Consider actions a_1 and a_2 which respectively allocate the same unary resource r with (t_1,d_1,r) and (t_2,d_2,r). This is the general case, which cannot always be handled with clocks. The question is whether the clock $c_r^{t_1,d_1}$ represent a sufficient amount of information of earlier allocations (t_1,d_1,r) of r so that the conflict with allocations (t_2,d_2,r) can be detected with it, by using the following constraint.

$$\begin{aligned} a_2@i\rightarrow&\big(t_1+d_1-c_r^{t_1,d_1}@i\leq t_2\big)\\ &\vee\big(t_1-c_r^{t_1,d_1}@i\geq t_2+d_2\big)\end{aligned} \tag{16}$$

It turns out that the clock is insufficient only if the immediately preceding action allocates the resource *relatively later than* the current action does, and some still earlier action allocates the resource in a conflicting way.

Proposition 1 *The clock* $c_r^{t_1,d_1}$ *is sufficient if* $t_2+d_2\geq t_1$.

Proof: Consider action a_2, taken at time t and allocating r with (t_2,d_2,r), as well as actions a_0 and a_1, both of which allocate the unary resource r with (t_1,d_1,r), with a_1 taken at some time $t'<t$ and a_0 at some time t'' such that $t''+d_1\leq t'$ (because both use the resource for duration d_1).

We claim that if the allocation of r by a_2 at time t_0'' does not conflict with the allocation by a_1, then it will not conflict with the allocation by a_0 either. Hence the conflict can be detected from the clock which only represents the allocation by the later action a_1 but not by a_0. Since a_1 and a_2 do not conflict, their intervals of allocation do not overlap, that is,

$$t'+t_1+d_1\leq t+t2$$

or

$$t'+t_1\geq t+t_2+d_2.$$

By assumption $t_2+d_2\geq t_1$ and $t>t'$, and hence the second condition cannot be true, that is, a_1 could not possibly allocate the resource after a_2 has freed it. Therefore the only possibility of not having a conflict is that

$$t'+t_1+d_1\leq t+t_2.$$

Now, because $t''<t'$ we have

$$t''+t_1+d_1\leq t+t_2.$$

Hence a_2 does not conflict with a_0 either. □

Same considerations apply to state resources.

In summary, the general scheme for deriving resource constraints is as follows. For a given resource r, we consider each possible pair of allocations (t_1,d_1,r) and (t_2,d_2,r) at a time (with all actions making the same allocation considered together), and create a constraint to prevent the conflict between them. If $t_1=0$, then we can always use formula (15). If $t_1>0$, then we use Proposition 1 to check if the clock c^{t_1,d_1} with formula (16) is sufficient, and if not, we use formula (9) instead. For standard benchmarks this scheme generally leads to the very compact constraints (15).

6 Shared Clocks for Effect Delays

Section 3.2 described how can could be used for representing effects' delay (for relative time points >0). The high number of real-valued variables required makes that representation impractical. However, a different kind of scheme for using clocks typically avoids the high number of clocks. The basic observation is that two (or more) actions can share a clock whenever the actions cannot temporally overlap. Temporal overlap is not possible when the actions use the same exclusive resource for their whole duration.

Example 4 *Consider actions for moving an object from location to location. To prevent taking two actions moving the object from one location to two different locations, all these action allocate the same resource specific to the object for their whole duration.*

Now, in cases like above we can use one clock for a large number of actions. The clock is reset when an action is taken, and the actions' effects take place when the clock reaches a value corresponding to the delay of the effect.

The basic idea is to use the clock for those resource allocations that start at relative time point 0 of the action, and that last until the last effect of the action. The actions cannot overlap because they use the same unary resource for their whole duration. Hence this one clock can be shared by multiple actions. This way, practically all standard benchmark problems require only some dozens of clocks.

6.1 Qualitative Clocks

Since a shared clock does not tell which action is active, we must use *qualitative* (Boolean) clocks to do that. These indicate a *qualitative* value range for the clock (a single value, or a range of values).

Note that while the clock is initialized to $-d$ when the action allocates the resource with (t,d,r), the time values of the qualitative clocks run from 0 until d, for clarity. The constraints for connecting the real-valued clock to the qualitative clocks are given as (17)-(19), (27), (28), (36). The connection between the real-valued clock c and the qualitative clock c_q representing the time since the start of the action, to the reset value of the clock $-d_a$ for this action, is

$$c_q = c + d_a.$$

Let $T=\{t_1,\ldots,t_n\}$ be time points, for example those (relative) time points in which the effects of a given action take place.We use the propositional variables, $q^a_{<t_i}@j$ and $q^a_{t_i}@j$, for $i\in\{1,\ldots,n\}$ to abstractly represent the clock value c satisfying respectively $t_{i-1}<c<t_i$ and $c=t_i$. The variables $q^a_{<t_i}$ and $q^a_{t_i}$ are related to the real-valued clock in the obvious way (where $c+d_a=c_q$ as discussed above.)

$$q^a_{<t_i}@j \;\rightarrow\; (t_{i-1}<c@j+d_a) \tag{17}$$
$$q^a_{<t_i}@j \;\rightarrow\; (c@j+d_a<t_i) \tag{18}$$
$$q^a_{t_i}@j \;\rightarrow\; (c@j+d_a=t_i) \tag{19}$$

Qualitative clocks are also useful because reasoning about the clock values is possible before any of the relevant real-valued variables have been set a value by the SMT solver. To support these inferences we use the following formulas for all $i\in\{1,\ldots,n-1\}$ and $k\in\{2,\ldots,n\}$.

$$a@j \;\rightarrow\; q^a_{<t_1}@(j+1)\vee q^a_{t_1}@(j+1) \tag{20}$$
$$q^a_{<t_i}@j \;\rightarrow\; q^a_{<t_i}@(j+1)\vee q^a_{t_i}@(j+1) \tag{21}$$
$$q^a_{t_i}@j \;\rightarrow\; q^a_{<t_{i+1}}@(j+1)\vee q^a_{t_{i+1}}@(j+1) \tag{22}$$
$$q^a_{<t_k}@j \;\rightarrow\; q^a_{<t_k}@(j-1)\vee q^a_{t_{k-1}}@(j-1) \tag{23}$$
$$q^a_{t_k}@j \;\rightarrow\; q^a_{<t_k}@(j-1)\vee q^a_{t_{k-1}}@(j-1) \tag{24}$$
$$q^a_{<t_1}@j \;\rightarrow\; q^a_{<t_1}@(j-1)\vee a@(j-1) \tag{25}$$
$$q^a_{t_1}@j \;\rightarrow\; q^a_{<t_1}@(j-1)\vee a@(j-1) \tag{26}$$

All the qualitative clock variables for a given step are mutually exclusive. Further, when the qualitative clocks of different actions represent the same real-valued clock, all the qualitative clock variables also for all these different actions are mutually exclusive.

6.2 Summarized Effects

For the encodings that allow changes at time points not made explicit as a step, as proposed earlier, we introduce variables

$$q^a_{\geq t_i}@j$$

for indicating that the clock reaches value t_i at step j or between steps $j-1$ and j. These variables are related to the clock values as follows.

$$q^a_{\geq t_i}@j \rightarrow (c@(j-1)+d_a+\Delta_i\geq t_i) \tag{27}$$
$$q^a_{\geq t_i}@j \rightarrow (c@(j-1)+d_a<t_i) \tag{28}$$

The $q^a_{\geq t_i}$ variables are used with $q^a_{<t_i}$ (and without $q^a_{t_i}$, replacing them), and they connect together more loosely. We have for all $i\in\{1,\ldots,n-1\}$ and $k\in\{2,\ldots,n\}$:

$$a@j\rightarrow q^a_{<t_1}@(j+1)\vee q^a_{\geq t_1}@(j+1) \tag{29}$$
$$q^a_{<t_i}@j\rightarrow q^a_{<t_i}@(j+1)\vee q^a_{\geq t_i}@(j+1) \tag{30}$$
$$q^a_{<t_k}@j\rightarrow q^a_{<t_k}@(j-1)\vee q^a_{\geq t_{k-1}}@(j-1) \tag{31}$$
$$q^a_{<t_1}@j\rightarrow q^a_{<t_1}@(j-1)\vee a@(j-1) \tag{32}$$
$$q^a_{\geq t_i}@j\rightarrow q^a_{\geq t_{i+1}}@j\vee q^a_{<t_{i+1}}@(j{+}1)\vee q^a_{\geq t_{i+1}}@(j{+}1) \tag{33}$$
$$q^a_{\geq t_k}@j\rightarrow q^a_{\geq t_{k-1}}@j\vee q^a_{<t_k}@(j{-}1)\vee q^a_{\geq t_{k-1}}@(j{-}1) \tag{34}$$
$$q^a_{\geq t_1}@j\rightarrow q^a_{<t_1}@(j-1)\vee a@(j-1) \tag{35}$$

Note that we can have $q^a_{\geq t_i}@j$ and $q^a_{\geq t_j}@j$ true at the same step j, even for multiple t_j, $i\neq j$. Additionally, variables $q^a_{<t_i}$ satisfy the following.

$$q^a_{<t_i}@j\rightarrow (c@(j+1)+d_a<t_i) \tag{36}$$

In formulas *causes*$(x)@i$ the qualitative clocks are used similarly to real-valued action-specific clocks. Effect triggering in effect and frame axioms (2) and (5) in different encoding styles is summarized below.

disjunct of *causes*$(x)@i$	for clock type
$c@i=t$	own
$q^a_t@i$	shared
$c@i\geq t\wedge c@(i-1)<t$	own, summarization
$q^a_{\geq t}@i$	shared, summarization

		ITSAT	SD	C	R
08-crewplanning	30	30	10	14	15
08-elevators	30	16	4	6	9
08-elevators-num	30	-	4	8	13
08-openstacks	30	30	4	5	7
08-pegsol	30	30	30	30	30
08-sokoban	30	17	17	17	16
08-transport	30	-	4	6	8
08-woodworking	30	-	16	15	23
08-openstacks-adl	30	-	3	5	8
08-openstacks-num-adl	30	-	5	9	18
11-floortile	20	20	20	20	20
11-matchcellar	10	10	10	10	10
11-parking	40	9	12	12	12
11-storage	20	10	0	0	0
11-tms	20	20	20	20	20
11-turnandopen	20	20	18	18	18
14-floortile	20	20	20	20	20
14-matchcellar	20	20	19	20	19
14-parking	20	18	19	19	19
14-tms	20	20	20	20	20
14-turnandopen	20	9	5	5	5
14-driverlog	30	4	0	0	0
total (w/o numeric)	410	303	228	236	240
total	560	303	260	279	310

Table 1: Instances solved in 1800 seconds by domain

Only for the trigger $c@i{=}t$ we need to force a step for time t (formula (13)). For triggers $q^a_t@i$ this follows from the axioms for qualitative clocks, requiring that all qualitative clock-value ranges are visited in turn.

7 Experiments

We compare our result to the Shin-Davis style step scheme and ITSAT which is one of the the strongest temporal planners [Rankooh and Ghassem-Sani, 2015], shown to outperform earlier planners [Gerevini *et al.*, 2006; Coles *et al.*, 2010; Eyerich *et al.*, 2012; Lu *et al.*, 2013].

Experimentation was based on Rintanen's [2015a] code, including the discretization method to eliminate time variables whenever possible. We used MathSAT 5.3.6 [Audemard *et al.*, 2005; Cimatti *et al.*, 2013] for instances with real-valued variables, and PrecoSAT [Biere, 2010] for purely Boolean ones. Experiments were run in Intel Xeon CPUs.

From the SMT approach, SD is the baseline encoding [Rintanen, 2015a]. C is obtained from SD by using the clock-based encodings of resource constraints and delays (Sections 5 and 6). R additionally uses *summarized* steps (Section 4).

Table 1 lists the number of IPC instances solved in 1800 seconds. ITSAT doesn't handle numeric variables and cannot solve the problems indicated with a dash. Differences between SD, C and R are not clearly visible here: many problem series are solved (almost) completely by all planners, some series are too difficult, and e.g. Parking gets fully discretized and all three planners use the same SAT encoding. Figure 3 plots makespans for instances solved by both ITSAT and R, with ITSAT makespans often close to twice those of R. R makespans are higher only with few instances of TMS. On these, C makespans are the same as ITSAT's. Figure 4 shows

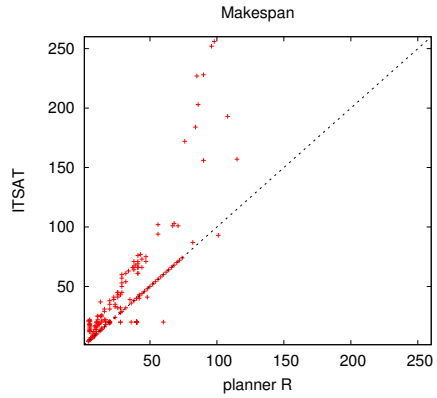

Figure 3: Comparison of makespans ITSAT vs. R

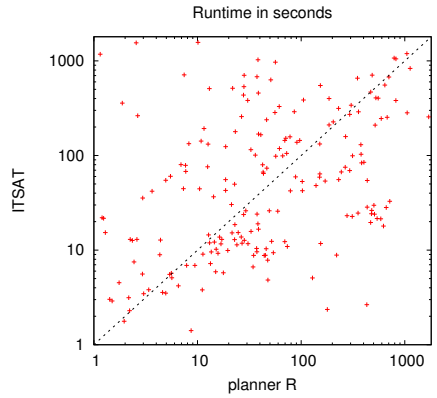

Figure 4: Comparison of runtimes ITSAT vs. R

that ITSAT's overall runtime advantage is not very systematic. Other data shows that R quite systematically improves on C, but not on SD (despite overall improvement).

8 Conclusion

We proposed new ways of using clocks in SMT encodings of temporal planning, decreasing the asymptotic size of some of the core constraints for temporal planning, and showed that the number of steps – a main factor in SAT/SMT solver performance – can be reduced with a relaxed encoding scheme. Both contributions improve the scalability of SMT in temporal planning. Our experiments showed that although the scalability and runtime improvements are not yet sufficient to match the state of the art, represented by the ITSAT planner, plan quality is much better than with ITSAT, which often generates plans with a makespan twice or more of the optimal.

Future work includes further investigation of schemes for reducing the complexity of handling temporal dependencies between actions, as well as better implementation technologies for temporal planning in general, for example following the lines that have proved successful with classical planning [Rintanen, 2012a; 2012b].

References

[Audemard *et al.*, 2005] Gilles Audemard, Marco Bozzano, Alessandro Cimatti, and Roberto Sebastiani. Verifying industrial hybrid systems with MathSAT. *Electronic Notes in Theoretical Computer Science*, 119(2):17–32, 2005.

[Biere, 2010] Armin Biere. Lingeling, Plingeling, PicoSAT and PrecoSAT at SAT Race 2010. Technical Report 1, Institute for Formal Models and Verification, Johannes Kepler University, 2010.

[Bryce *et al.*, 2015] Daniel Bryce, Sicun Gao, David J Musliner, and Robert P. Goldman. SMT-based nonlinear PDDL+ planning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15)*, pages 3247–3253. AAAI Press, 2015.

[Cimatti *et al.*, 2013] Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. The MathSAT5 SMT solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 93–107. Springer-Verlag, 2013.

[Coles *et al.*, 2010] Amanda Jane Coles, Andrew Coles, Maria Fox, and Derek Long. Forward-chaining partial-order planning. In *ICAPS 2010. Proceedings of the Twentieth International Conference on Automated Planning and Scheduling*, pages 42–49. AAAI Press, 2010.

[Eyerich *et al.*, 2012] Patrick Eyerich, Robert Mattmüller, and Gabriele Röger. Using the context-enhanced additive heuristic for temporal and numeric planning. In *Towards Service Robots for Everyday Environments*, pages 49–64. Springer-Verlag, 2012.

[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.*, 2006] Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. An approach to temporal planning and scheduling in domains with predictable exogenous events. *Journal of Artificial Intelligence Research*, 25:187–231, 2006.

[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.

[Kautz and Selman, 1999] Henry Kautz and Bart Selman. Unifying SAT-based and graph-based planning. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 318–325. Morgan Kaufmann Publishers, 1999.

[Lu *et al.*, 2013] Qiang Lu, Ruoyun Huang, Yixin Chen, You Xu, Weixiong Zhang, and Guoliang Chen. A SAT-based approach to cost-sensitive temporally expressive planning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):18, 2013.

[Rankooh and Ghassem-Sani, 2015] Masood Feyzbakhsh Rankooh and Gholamreza Ghassem-Sani. ITSAT: an efficient SAT-based temporal planner. *Journal of Artificial Intelligence Research*, 53:541–632, 2015.

[Rintanen *et al.*, 2006] Jussi Rintanen, Keijo Heljanko, and Ilkka Niemelä. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence*, 170(12-13):1031–1080, 2006.

[Rintanen, 2007] Jussi Rintanen. Complexity of concurrent temporal planning. In *ICAPS 2007. Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, pages 280–287. AAAI Press, 2007.

[Rintanen, 2012a] Jussi Rintanen. Engineering efficient planners with SAT. In *ECAI 2012. Proceedings of the 20th European Conference on Artificial Intelligence*, pages 684–689. IOS Press, 2012.

[Rintanen, 2012b] Jussi Rintanen. Planning as satisfiability: heuristics. *Artificial Intelligence*, 193:45–86, 2012.

[Rintanen, 2015a] 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.

[Rintanen, 2015b] Jussi Rintanen. Models of action concurrency in temporal planning. In *IJCAI 2015, Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1659–1665. AAAI Press, 2015.

[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, Surfers Paradise, Gold Coast, Australia, December 2-6, 2007, Proceedings*, 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.