Temporal Planning through Reduction to Satisfiability Modulo Theories

Jussi Rintanen

Department of Computer Science
Aalto University, Finland

December 8, 2016

Outline of the Talk

Temporal Planning = planning for concurrent actions with durations

This work summarizes progress in the last couple of years.

Fundamental improvements to solving temporal planning by SMT

1. improved problem modeling (Rintanen IJCAI-2015)
2. discretization (Rintanen AAAI-2015)
3. relaxed (summarized) steps (unpublished work)

Basic SMT Representation of Temporal Planning

- Starting point: Shin & Davis, AI Journal 2005.
- Working encodings, but not very scalable.
- Issues:
  - encodings have a large size
  - too many steps (unnecessarily high horizon length)
- AI Planning community has instead focused on:
  - reductions to untimed planning
  - explicit state-space search
- state-of-the-art: Rankooh & Ghassem-Sani (AI Journal 2015):
  - reduction to untimed planning and further to SAT, with methods from Rintanen et al. (AIJ 2006)

Basic SMT Representation of Temporal Planning

- Starting point: Shin & Davis, AI Journal 2005.
- Working encodings, but not very scalable.
- Issues:
  - encodings have a large size
  - too many steps (unnecessarily high horizon length)
- AI Planning community has instead focused on:
  - reductions to untimed planning
  - explicit state-space search
- state-of-the-art: Rankooh & Ghassem-Sani (AI Journal 2015):
  - reduction to untimed planning and further to SAT, with methods from Rintanen et al. (AIJ 2006)

problem instance:
$X = \{x_1, \ldots, x_n\}$ (state variables)
$A = \{a_1, \ldots, a_m\}$ (actions)
$0, \ldots, N+1$ (steps)

SMT variables:
$x@i$ for $x \in X$, $i \in \{0, \ldots, N+1\}$
$a@i$ for $a \in A$, $i \in \{0, \ldots, N\}$
$\tau@i$ for absolute time at step i
$\Delta@i = \tau@i - \tau@(i-1)$

Basic SMT Representation of Temporal Planning

SMT Formulas

Preconditions:

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

Effects:

$$\mathit{causes}(x)@i \rightarrow x@i \tag{2}$$

$$\mathit{causes}(\neg x)@i \rightarrow \neg x@i \tag{3}$$

where *causes*$(l)@i =$ all conditions under which literal l becomes true at i.

Frame Axioms:

$$(x@i \wedge \neg x@(i-1)) \rightarrow \mathit{causes}(x)@i \tag{4}$$

$$(\neg x@i \wedge x@(i-1)) \rightarrow \mathit{causes}(\neg x)@i \tag{5}$$

causes$(x)@i$ = disjunction of all

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

for actions a with effect x at t.

There must be a step at time t relative to the action a:

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

Basic SMT Representation of Temporal Planning

causes$(x)@i$

causes$(x)@i$ = disjunction of all

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

for actions a with effect x at t.

There must be a step at time t relative to the action a:

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

Action non-overlap in PDDL 2.1

In PDDL 2.1 (implicit) resources are allocated by a two-step process:

1. Confirm that given resource is available (precondition $x = 0$)
2. Allocate the resource (assign $x := 1$ *at start*)

This takes place inside a 0-duration critical section.

Advantage

Easy to encode as $\neg a_1@i \vee \neg a_2@i$ whenever precondition of a_1 conflicts with time 0 effect of a_2.

Disadvantage

Deallocation and reallocation of a resource cannot be at the same time, leading to ϵ gaps in plans

PDDL 2.1 schedule: $move_{a,b}$ $move_{b,c}$ $move_{c,d}$

Desired schedule: $move_{a,b}$ $move_{b,c}$ $move_{c,d}$

Action non-overlap in PDDL 2.1

In PDDL 2.1 (implicit) resources are allocated by a two-step process:

1. Confirm that given resource is available (precondition $x = 0$)
2. Allocate the resource (assign $x := 1$ *at start*)

This takes place inside a 0-duration critical section.

Advantage

Easy to encode as $\neg a_1@i \vee \neg a_2@i$ whenever precondition of a_1 conflicts with time 0 effect of a_2.

Disadvantage

Deallocation and reallocation of a resource cannot be at the same time, leading to ϵ gaps in plans

Action non-overlap in PDDL 2.1

In PDDL 2.1 (implicit) resources are allocated by a two-step process:

1. Confirm that given resource is available (precondition $x = 0$)
2. Allocate the resource (assign $x := 1$ *at start*)

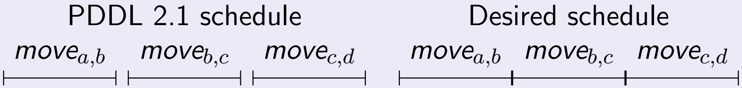
This takes place inside a 0-duration critical section.

Advantage

Easy to encode as $\neg a_1@i \vee \neg a_2@i$ whenever precondition of a_1 conflicts with time 0 effect of a_2.

Disadvantage

Deallocation and reallocation of a resource cannot be at the same time, leading to ϵ gaps in plans



Alternative mechanisms of action non-overlap

Rintanen IJCAI-2015

Make resources explicit in the modeling language!

Advantage

Trivial to have a_1 at 0 and a_2 at 1 when

1. a_1 allocates resource at $]0,1[$, and
2. a_2 allocates resource at $]0,1[$

Disadvantage (...but not really!)

Encodings are more complicated! However, there are encodings that are (Rintanen 2017, unpublished)

- close to linear-size in practice,
- require only a small number of real-valued SMT variables,
- far better scalable than earlier encodings.

Alternative mechanisms of action non-overlap

Rintanen IJCAI-2015

Make resources explicit in the modeling language!

Advantage

Trivial to have a_1 at 0 and a_2 at 1 when

1. a_1 allocates resource at $]0, 1[$, and
2. a_2 allocates resource at $]0, 1[$

Disadvantage (...but not really!)

Encodings are more complicated! However, there are encodings that are (Rintanen 2017, unpublished)

- close to linear-size in practice,
- require only a small number of real-valued SMT variables,
- far better scalable than earlier encodings.

Discretization

Rintanen AAAI-2015

- Temporal planning generally defined with real or rational time
- Not always obvious if integer time can be used instead
- However, automated methods to recognize this exist (Rintanen AAAI-2015), covering most of the practically occurring problems
- SAT fragment of SMT sufficient (and practical) when
  1. problem instance discretizable,
  2. all action durations short, like 1 or 2 or 3, and
  3. there are no real-valued state variables.
- Leads to large performance gains!

From Implicit (PDDL) to Explicit (NDL) Resources

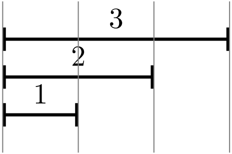
		Z3 SMT solver			
		PDDL	NDL	dNDL	ITSAT
2008-PEGSOL	30	28	30	30	30
2008-SOKOBAN	30	1	5	13	16
2011-FLOORTILE	20	0	5	18	20
2011-MATCHCELLAR	10	3	5	8	10
2011-PARKING	20	3	7	8	10
2011-TURNANDOPEN	20	4	10	16	20
2008-CREWPLANNING	30	4	10	9	30
2008-ELEVATORS	30	0	4	7	15
2008-TRANSPORT	30	0	0	4	error
2011-TMS	20	7	8	8	20
2008-OPENSTACKS	30	0	0	0	24
2008-OPENSTACKS-ADL	31	0	2	3	error
2011-STORAGE	19	0	0	0	error
total	320	50	86	124	195
weighted score	13	2.10	3.70	5.50	8.33

Comment: dNDL = NDL + discretization
Comment: ITSAT's problem representation ignores time & makespan ⇒ cannot be (easily) modified to improve quality of plans

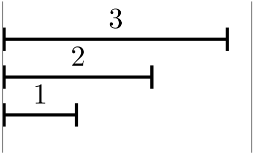
Relaxed (Summarized) Step Scheme

Reduction in the number of steps

Traditional encodings require a step for every effect:



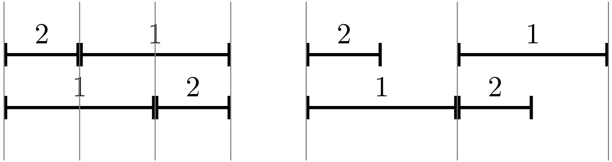
Our relaxed (summarized) encoding needs (far) fewer steps:



Relaxed (Summarized) Step Scheme

Increase in makespan

Shortest makespan may require more steps:



Experiments

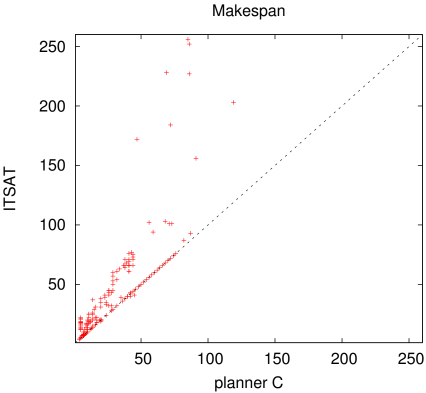
- Demonstration of scalability improvements
  1. better models with explicit resources (Rintanen IJCAI-2015)
  2. discretization (Rintanen AAAI-2015)
  3. encodings with clocks + relaxed (summarized) steps (unpublished)
- Comparison to ITSAT (Rankooh & Ghassem-Sani AI Journal 2015): reduction to untimed planning followed by reduction to SAT with best parallel encodings (Rintanen et al. 2006)

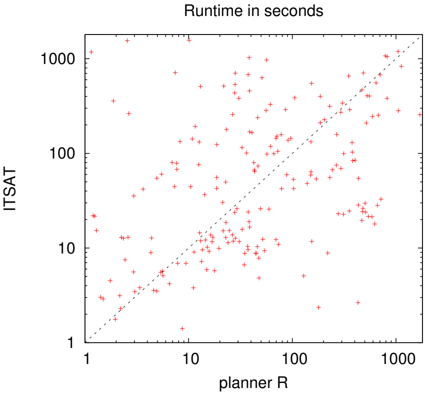
  ITSAT search phase ignores time information $\Rightarrow$ no effective minimization of plan duration (makespan)
- Conclusion: impressive improvements, but runtimes still behind ITSAT

Impact of Clock Encodings and Relaxed Step Scheme

		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	560	303	260	279	310

Impact of Clock Encodings and Relaxed Step Scheme





Conclusion

- Dramatic performance improvements in Planning by SMT:
  1. change in temporal model, explicit resources
  2. discretization
  3. relaxed (summarized) steps
- quality of plans (makespan) far better than in competition
- scalability a bit behind (possibly due to SMT/SAT solver differences)