Temporal Planning with Clock-Based SMT Encodings

Jussi Rintanen

Department of Computer Science
Aalto University
Helsinki, Finland
(Also affiliated with Griffith University, Brisbane, Australia, and the Helsinki Institute of Information Technology, Finland.)

August 2017

Temporal Planning

- classical planning = choose action sequence to reach a goal
- temporal planning = choose actions + schedule (with concurrency)
- pioneering work by Shin & Davis (2005)
  - complex modeling language
  - effective representation in SAT modulo Theories (SMT) framework
  - SMT before used for classical planning with numeric variables (Wolfman & Weld 1999)
- Few follow-ups to Shin&Davis 2005: room for improvement

Temporal Planning

- classical planning = choose action sequence to reach a goal
- temporal planning = choose actions + schedule (with concurrency)
- pioneering work by Shin & Davis (2005)
  - complex modeling language
  - effective representation in SAT modulo Theories (SMT) framework
  - SMT before used for classical planning with numeric variables (Wolfman & Weld 1999)
- Few follow-ups to Shin&Davis 2005: room for improvement

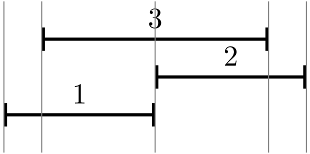
Temporal Planning by Constraints / Logic / SAT / SMT

- Starting point: Shin & Davis 2005 (PDDL 2.1 $\rightarrow$ SMT)
  - Issue 1: ϵ-semantics of PDDL 2.1 $\rightarrow$ far too many steps
  - Issue 2: discretization to integer time not available

  $\Rightarrow$ Poor scalability
- Rintanen (IJCAI 2015): eliminate ϵ semantics; NDL instead of PDDL
  - Advantage 1: Number of steps often halved! Big performance gains.
  - Advantage 2: Reasoning about action dependencies easier
- Rintanen (AAAI 2015): general discretization method
  - Advantage 1: Simpler encodings (often)
  - Advantage 2: Use SAT instead of SMT (many cases)
  - Advantage 3: Optimal makespan practical (when actions "short")
- This work:
  1. Summarized steps: peformance gains through even fewer steps,
  2. Clock-based delays: O(n) size vs. O(n^2) by Shin&Davis 2005

Step-Based Encodings

- Steps part of Kautz&Selman original work (1992)
  - Execution of a plan represented by states/steps $s_0, \ldots, s_n$
  - Encoding expresses transitions $s_i \Rightarrow s_{i+1}$ for all $i \in \{0, \ldots, n-1\}$
- Shin&Davis (2005): steps have a real-valued time



- A step is needed for
  - every action starting point
  - every (discrete) change by action or other event

Variables needed in SMT encodings of Temporal Planning

Let $0, \ldots, N$ be the steps. Following variables needed.

$x@i$	Boolean state variable x is true	$i \in \{0, \ldots, N\}$
$a@i$	action a is taken	$i \in \{0, \ldots, N\}$
$\tau@i$	absolute time at step i	$i \in \{0, \ldots, N\}$
$\Delta@i$	$= \tau@i - \tau@(i-1)$	$i \in \{1, \ldots, N\}$

Constraint: $\Delta@i > 0$

Precondition axioms

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

$$a@i \rightarrow \phi@i \quad (1)$$

where $\phi@i$ is the formula obtained from ϕ by replacing each x by $x@i$.

Effect Axioms and Frame Axioms

causes$(l)@i$ $=$ disjunction of all triggers for l becoming true

Effect axioms

$$\text{causes}(x)@i \rightarrow x@i \tag{2}$$
$$\text{causes}(\neg x)@i \rightarrow \neg x@i \tag{3}$$

Frame axioms

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

Shin&Davis-style encoding of causes$(x)@i$

Effect triggers in the Shin & Davis encodings

Trigger for action a and effect x at $t > 0$ in causes$(x)@i$:

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

This is $\mathcal{O}(n^2)$ size where n is the number of steps.

Step must exist

If action a has an effect at t, a step at relative time t must exist:

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

Clock-based encodings (Rintanen 2015)

Idea: every action has its own clock

Encoding

Let action a have effect x at t.

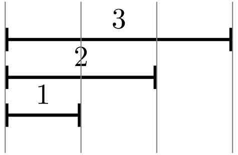
Clock is reset	$a@i \rightarrow c_a@i = 0$
Clock progresses	$\neg a@i \rightarrow c_a@i = c_a@(i-1) + \Delta@i$
Must stop at t	$(c_a@(i-1) < t) \rightarrow (c_a@i \leq t)$
Effect trigger	$c_a@i = t$

Encoding has a linear size. But, far too many real-valued variables. Slow!

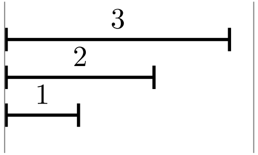
Contribution 1: Summarized Effects

Summarize discrete changes from multiple time points in a single step

Shin & Davis AIJ'05



Rintanen IJCAI'17



Fewer steps $\Rightarrow$ significant performance gain

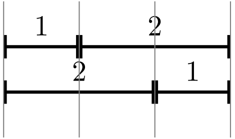
Contribution 1: Implementation

1. No need for axioms requesting a step a specific time!!!
2. Trigger $c_a@i = t$ for effects at relative time t becomes:

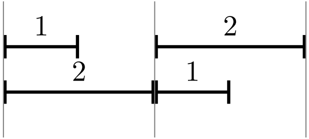
$$(c_a@(i-1) < t) \wedge (c_a@i \geq t) \tag{10}$$

Trade-offs: makespan vs. number of steps

4 steps, makespan 3



3 steps, makespan 4



Sometimes: shorter makespan $\Rightarrow$ more steps

(*Makespan – number of steps* trade-off in all encodings when one long action interchangeable with two short ones)

Contribution 2: Practical Encodings with Clocks

- Rintanen AAAI'15: clock for every action $\Rightarrow$ too many clocks $\Rightarrow$ slow
- New encodings: clocks shared by multiple actions

Idea:

- associate clocks with a resources
- resources represent exclusions of actions $\Rightarrow$ same clock can represent delays of exclusive actions

Why is this good?

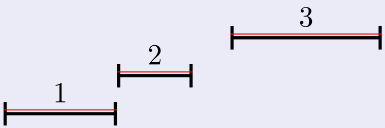
- Number of resources typically low (e.g. 30 resources vs. 1000 actions)
- Number of clocks low $\Rightarrow$ number of real variables low $\Rightarrow$ fast

Contribution 2: Practical Encodings with Clocks

Common easy case

All actions a with resource R:

1. allocation from 0 to d_a
2. last effects at d_a



Clock c_R for resource R can be used for all delays

1. action a can be taken if $c_R \geq 0$
2. action a sets $c_R := -d_a$
3. effect at t triggered when $d_a - c_R = t$

See paper: very general condition for the sufficiency of one clock
See paper: examples where one clock not enough

Contribution 2: Qualitative Clocks for Shared Clocks

A shared clock does not indicate which action is currently active.
Need qualitative (Boolean) clocks for every action.

- The "clock" distinguishes between "qualitative" values:
  1. start of action (action variable is true)
  2. time points where action *active* but no effects
  3. time points where action's effects take place
- Encode rules for transitions from one value to the next
- Connect qualitative clock variables to real-valued clock variables

Experiments

1. SD – encoding of delays/steps as proposed by Shin&Davis
2. C – our new clock encoding
3. R – our new clock encoding + relaxed/summarized steps
4. ITSAT – leading temporal planner (Rankooh & Ghassem-Sani 2013, AIJ'15)
    - Reduction to untimed/classical planning
    - Solved with efficient classical SAT encodings (Rintanen et al. 2006)
    - If solution schedulable to correct plan, done.
    - Otherwise add constraints and try again.

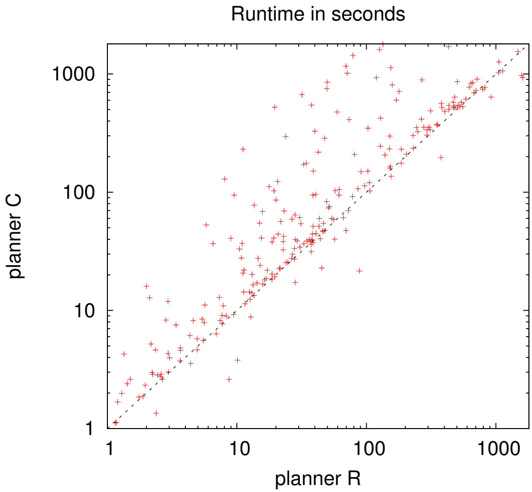
SAT solver ignores metric time: scalability (often) good, plans not!

Experiments: Solved Instances

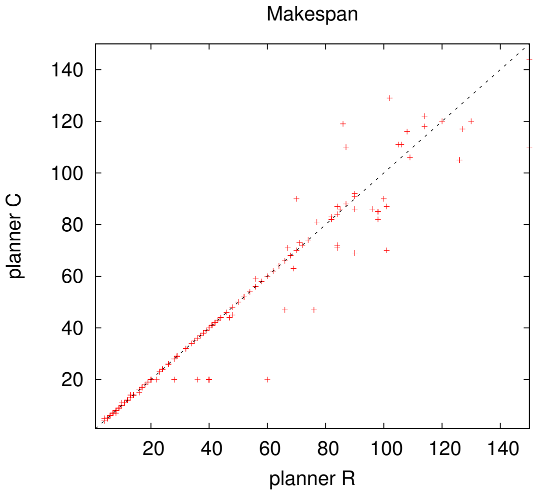
		ITSAT	SD	C	R	
08-crewplanning	30	30	10	14	15	
08-elevators	30	16	4	6	9	
08-elevators-num	30	-	4	8	13	(numeric vars: ITSAT not applicable)
08-openstacks	30	30	4	5	7	
08-pegsol	30	30	30	30	30	
08-sokoban	30	17	17	17	16	
08-transport	30	-	4	6	8	(numeric vars: ITSAT not applicable)
08-woodworking	30	-	16	15	23	(numeric vars: ITSAT not applicable)
08-openstacks-adl	30	-	3	5	8	(numeric vars: ITSAT not applicable)
08-openstacks-num-adl	30	-	5	9	18	(numeric vars: ITSAT not applicable)
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: Instances solved in 1800 seconds by domain

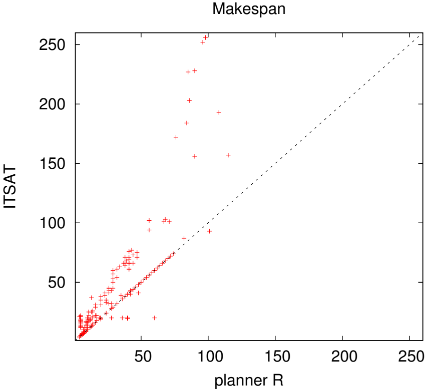
Experiments: Summarization Improves Runtimes



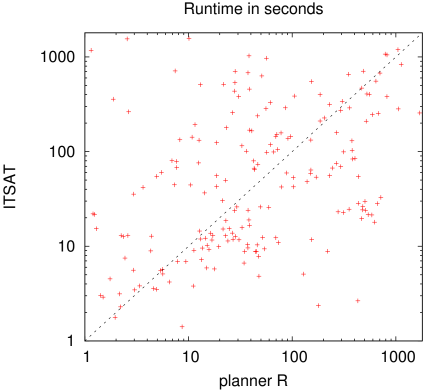
Experiments: Summarization Can Worsen Makespans



Experiments: Far Better Makespans than with ITSAT



Experiments: ITSAT's Runtime Advantage Unsystematic



Conclusion

- First competitive clock-based encodings with clock-sharing and resources
- Summarization reduces number of steps $\Rightarrow$ better scalability
- Comparison to ITSAT:
    - On average, scalability still not as good
    - Often far superior plans (shorter makespan)