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
Starting point: Shin & Davis 2005 (PDDL 2.1 → SMT)
  - Issue 1: $\epsilon$-semantics of PDDL 2.1 → far too many steps
  - Issue 2: discretization to integer time not available
⇒ Poor scalability
Rintanen (IJCAI 2015): eliminate $\epsilon$ 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: performance 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: $\Delta@i > 0$
If $\phi$ is the precondition of action $a$, we have the formula

$$a@i \rightarrow \phi@i$$

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

(1)
Effect Axioms and Frame Axioms

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

Effect axioms

causes(x)@i → x@i \hspace{1cm} (2)
causes(\neg x)@i → \neg x@i \hspace{1cm} (3)

Frame axioms

(x@i \land \neg x@(i - 1)) → causes(x)@i \hspace{1cm} (4)
(\neg x@i \land x@(i - 1)) → causes(\neg x)@i \hspace{1cm} (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 \((x)@i\):

\[
\bigvee_{j=0}^{i-1} \left( a@j \land \left( (\tau@i - \tau@j) = t \right) \right) \tag{6}
\]

This is \(O(n^2)\) 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

<table>
<thead>
<tr>
<th>Condition</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock is reset</td>
<td>$a_{@i}</td>
</tr>
</tbody>
</table><p>ightarrow c_a{@i} = 0$                |
| Clock progresses           | $\neg a_{@i}ightarrow c_a{@i} = c_a{@}(i - 1) + \Delta@i$ |
| Must stop at $t$           | $(c_a{@}(i - 1) &lt; t) \rightarrow (c_a{@i} \leq t)$ |
| Effect trigger             | $c_a{@i} = t$                                    |</p>

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 ⇒ 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:

$$ (c_a @ (i - 1) < t) \land (c_a @ i \geq t) $$ (10)
Trade-offs: makespan vs. number of steps

4 steps, makespan 3

3 steps, makespan 4

Sometimes: shorter makespan ⇒ 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
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
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

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

**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

![Runtime in seconds](image)

- ITSAT planner:
  - Runtime in seconds

Jussi Rintanen (Aalto U, Dept of CS)
IJAI 2017 25 / 26
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)