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 relaxed (summarized) steps (unpublished work)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

- 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

- 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:
\[ X = \{x_1, \ldots, x_n \} \text{ (state variables)} \]
\[ A = \{a_1, \ldots, a_m \} \text{ (actions)} \]
\[ 0, \ldots, N + 1 \text{ (steps)} \]

SMT variables:
\[ x@i \text{ for } x \in X, i \in \{0, \ldots, N + 1 \} \]
\[ a@i \text{ for } a \in A, i \in \{0, \ldots, N \} \]
\[ \tau@i \text{ for absolute time at step } i \]
\[ \Delta@i = \tau@i - \tau@{(i - 1)} \]
Basic SMT Representation of Temporal Planning

SMT Formulas

Preconditions:

\[ a@i \rightarrow \phi@i \]  (1)

Effects:

\[ causes(x)@i \rightarrow x@i \]  (2)
\[ causes(\neg x)@i \rightarrow \neg x@i \]  (3)

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

Frame Axioms:

\[ (x@i \land \neg x@(i - 1)) \rightarrow causes(x)@i \]  (4)
\[ (\neg x@i \land x@(i - 1)) \rightarrow causes(\neg x)@i \]  (5)
Basic SMT Representation of Temporal Planning

$$causes(x)@i$$

$$causes(x)@i = \text{disjunction of all}$$

$$\bigvee_{j=0}^{i-1} (a@j \land ((\tau@i - \tau@j) = t))$$  \hspace{1cm} (6)

for actions $$a$$ with effect $$x$$ at $$t$$.

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

$$a@i \rightarrow \bigvee_{j=i+1}^{N} (\tau@j - \tau@i = t).$$  \hspace{1cm} (7)
Basic SMT Representation of Temporal Planning

\textit{causes}(x)@i

\textit{causes}(x)@i = \text{disjunction of all}

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

for actions \( a \) with effect \( x \) at \( t \).

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

\[ a@i \rightarrow \bigvee_{j=i+1}^{N} (\tau@j - \tau@i = t). \]  \hspace{1cm} (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 $\neg a_1@i \lor \neg a_2@i$ 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 $\epsilon$ gaps in plans

<table>
<thead>
<tr>
<th>PDDL 2.1 schedule</th>
<th>Desired schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$move_{a,b}$</td>
<td>$move_{a,b}$</td>
</tr>
<tr>
<td>$move_{b,c}$</td>
<td>$move_{b,c}$</td>
</tr>
<tr>
<td>$move_{c,d}$</td>
<td>$move_{c,d}$</td>
</tr>
</tbody>
</table>
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 \( x := 1 \) \textit{at start})

This takes place inside a 0-duration critical section.

**Advantage**

Easy to encode as \( \neg a_1 @i \lor \neg a_2 @i \) 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 \( \epsilon \) gaps in plans.

<table>
<thead>
<tr>
<th>PDDL 2.1 schedule</th>
<th>Desired schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( move_{a,b} )</td>
<td>( move_{a,b} )</td>
</tr>
<tr>
<td>( move_{b,c} )</td>
<td>( move_{b,c} )</td>
</tr>
<tr>
<td>( move_{c,d} )</td>
<td>( move_{c,d} )</td>
</tr>
</tbody>
</table>
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 \( \neg a_1 @i \lor \neg a_2 @i \) 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 \( \epsilon \) gaps in plans.

<table>
<thead>
<tr>
<th>PDLL 2.1 schedule</th>
<th>Desired schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( move_{a,b} )</td>
<td>( move_{a,b} )</td>
</tr>
<tr>
<td>( move_{b,c} )</td>
<td>( move_{b,c} )</td>
</tr>
<tr>
<td>( move_{c,d} )</td>
<td>( move_{c,d} )</td>
</tr>
</tbody>
</table>
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.
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.
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

<table>
<thead>
<tr>
<th>Problem Set</th>
<th>Z3 SMT solver</th>
<th></th>
<th></th>
<th></th>
<th>ITSAT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PDDL</td>
<td>NDL</td>
<td>dNDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2008-PEGSOl</td>
<td>30</td>
<td>28</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>2008-SOKOBAN</td>
<td>30</td>
<td>1</td>
<td>5</td>
<td>13</td>
<td>16</td>
</tr>
<tr>
<td>2011-FLOOR TILE</td>
<td>20</td>
<td>0</td>
<td>5</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>2011-MATCHCELLAR</td>
<td>10</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>2011-PARKING</td>
<td>20</td>
<td>3</td>
<td>7</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>2011-TURNANDOPEN</td>
<td>20</td>
<td>4</td>
<td>10</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>2008-CREWPLANNING</td>
<td>30</td>
<td>4</td>
<td>10</td>
<td>9</td>
<td>30</td>
</tr>
<tr>
<td>2008-ELEVATORS</td>
<td>30</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>2008-TRANSPORT</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>error</td>
</tr>
<tr>
<td>2011-TMS</td>
<td>20</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>2008-OPENSTACKS</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>2008-OPENSTACKS-ADL</td>
<td>31</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>error</td>
</tr>
<tr>
<td>2011-STORAGE</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>error</td>
</tr>
<tr>
<td><strong>total</strong></td>
<td>320</td>
<td>50</td>
<td>86</td>
<td>124</td>
<td>195</td>
</tr>
<tr>
<td><strong>weighted score</strong></td>
<td>13</td>
<td>2.10</td>
<td>3.70</td>
<td>5.50</td>
<td>8.33</td>
</tr>
</tbody>
</table>

**Comment:** dNDL = NDL + discretization

**Comment:** ITSAT's problem representation ignores time & makespan ⇒ cannot be (easily) modified to improve quality of plans

---

Jussi Rintanen (Aalto DCS)  
Temporal Planning  
CL Day 10 / 16
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

<table>
<thead>
<tr>
<th>Project</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>-</td>
<td>4</td>
<td>8</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>-</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>08-WOODWORKING</td>
<td>30</td>
<td>-</td>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>08-OPENSTACKS-ADL</td>
<td>30</td>
<td>-</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>08-OPENSTACKS-NUM-ADL</td>
<td>30</td>
<td>-</td>
<td>5</td>
<td>9</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>20</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</strong></td>
<td><strong>560</strong></td>
<td><strong>303</strong></td>
<td><strong>260</strong></td>
<td><strong>279</strong></td>
</tr>
</tbody>
</table>
Impact of Clock Encodings and Relaxed Step Scheme

Makespan

Runtime in seconds

Jussi Rintanen (Aalto DCS)
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)