Automatic Reductions from PH into STRIPS
or
How to Generate Short Problems with Very Long Solutions

Aldo Porco    Alejandro Machado    Blai Bonet

ICAPS. Rome, Italy. June 2013.
Recap from ICAPS-2011

Introduced a software tool that maps instances of an NP decision problem expressed in SO∃ into a STRIPS problem such that

1) instance is a positive instance iff the STRIPS problem has a plan
2) translation runs in polynomial time
3) STRIPS problem is decidable in non-deterministic polytime (NP)
4) plan, when exists, encodes the solution to input instance
**Software Tool**

\[
\text{DCT} \quad \sigma, \Phi, \mathcal{A} \quad \rightarrow \quad \text{tool} \quad \rightarrow \quad \text{PDDL} \quad \text{dom & ins}
\]

**Input:**
- signature \(\sigma\) that contains relational symbols
- \(SO\exists\) formula \(\Phi\) that encodes \textbf{NP} problem
- finite structure \(\mathcal{A}\) that encodes \textbf{instance}

**Output:**
- PDDLs for a \textbf{fragment} of STRIPS that is \textbf{decidable in NP}

**Guarantees:**
- runs in polytime for \textbf{fixed} \(\sigma\) and \(\Phi\)
- output is no \textbf{harder} that input (complexity wise)
Contributions

- Extend tool to target Polynomial Time Hierarchy (PH) instead of only NP
- Generated problems are general STRIPS problems
- Translator runs in polynomial time
- Experimental evaluation over (somewhat) difficult instances

Use:
- Leverage current (planning) technology to NP problems
- Design new benchmarks for planning and test planners and heuristics
Descriptive Complexity Theory (DCT)

Studies complexity theory from a logical perspective without commitments to any model of computation

Major complexity classes had been characterized using different fragments of logic:

- NL is captured by $\text{SO}^\exists$-Krom (CNF with $\leq 2$ literals per clause)
- P is captured by $\text{SO}^\exists$-Horn (CNF with $\leq 1$ positive literal)
- NP is captured by $\text{SO}^\exists$
- PH is captured by $\text{SO}$
- PSPACE is captured by $\text{SO} + \text{TC}$ (SO + transitive-closure syntactic construct)
Polynomial Time Hierarchy (PH)

Infinite hierarchy of classes that contains P, NP, NP^{NP}, NP^{NP^{NP}}, etc.

Defined as \( \text{PH} = \bigcup_{k \geq 0} \Sigma_k^P \) where (using oracles):

- \( \Sigma_0^P = P \)
- \( \Sigma_1^P = \text{NP}^{\Sigma_0^P} = \text{NP}^P = \text{NP} \)
- \( \ldots \)
- \( \Sigma_{k+1}^P = \text{NP}^{\Sigma_k^P} \)

\( \text{PH} = \text{Co-PH} \) and hence \( \text{Co-NP} \) and \( \Pi_k^P \in \text{PH} \) for every \( k \geq 0 \)

It is believed that \( \text{PH} \neq \text{PSPACE} \); otherwise \( \text{PSPACE} = \Sigma_k^P \) from some \( k \)

**Cannonical problem** in \( \Sigma_k^P \) is to decide validity of \( \exists x_1 \forall x_2 \exists x_3 \cdots Qx_k \cdot \phi \)
Results 1/3

Random formulas of type $\exists\bar{x}\forall\bar{y}\exists\bar{z}.\varphi(\bar{x}, \bar{y}, \bar{z})$ in $\Sigma^P_3$

<table>
<thead>
<tr>
<th>$\exists\forall\exists$</th>
<th>$#\forall$</th>
<th>$#\exists$</th>
<th>$#\forall$</th>
<th>$n$</th>
<th>+</th>
<th>-</th>
<th>time</th>
<th>len</th>
<th>PDDL in KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>30</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td>4,199.2</td>
<td>—</td>
<td>17.5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>50</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td>2,313.9</td>
<td>—</td>
<td>18.4</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>30</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td>3,210.7</td>
<td>—</td>
<td>18.5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>50</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td>3,166.3</td>
<td>—</td>
<td>19.4</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>30</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
<td>3,313.4</td>
<td>—</td>
<td>19.4</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>50</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td></td>
<td>3,450.9</td>
<td>640</td>
<td>20.4</td>
</tr>
</tbody>
</table>

- Random $\exists\forall\exists$ problems with 150 clauses each
- Solved with Rintanen’s SAT-based planner M
- LAMA’11 does not perform well on this type of problems
Results 2/3
Random instances of $3\text{Col}$ in $\Pi^P_1 = \text{Co-NP}$

<table>
<thead>
<tr>
<th>$V$</th>
<th>$n$</th>
<th>$+$</th>
<th>$-$</th>
<th>time / $+$</th>
<th>time / $-$</th>
<th>plan len</th>
<th>PDDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0.1</td>
<td>0.8</td>
<td>1,731</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>0.6</td>
<td>67.9</td>
<td>6,695</td>
<td>0.6</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3.4</td>
<td>464.9</td>
<td>26,163</td>
<td>0.7</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>74.8</td>
<td>1.6</td>
<td>102,935</td>
<td>0.8</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>624.0</td>
<td>5.9</td>
<td>406,851</td>
<td>1.0</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>—</td>
<td>1</td>
<td>—</td>
<td>0.3</td>
<td>—</td>
<td>1.1</td>
</tr>
</tbody>
</table>

- 5 random graphs for each number of vertices ($V$)
- Solved with LAMA’11 and obtained very long plans!
- $M$ does not perform well on this type of problems
- **How come does LAMA’11 find a plan with > 400k actions?**
Results 3/3

Random instances of $3\text{Col}$ in $\Pi^P_1 = \text{Co-NP}$

<table>
<thead>
<tr>
<th>$V$</th>
<th>$n$</th>
<th>$+$</th>
<th>$-$</th>
<th>time $/+$</th>
<th>time $/-$</th>
<th>plan len</th>
<th>PDDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>1,850.1</td>
<td>0.1</td>
<td>1,731</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
<td>3</td>
<td>—</td>
<td>11.7</td>
<td>—</td>
<td>0.6</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td></td>
<td>3</td>
<td>—</td>
<td>81.9</td>
<td>—</td>
<td>0.7</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td></td>
<td>2</td>
<td>—</td>
<td>0.2</td>
<td>—</td>
<td>0.8</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td></td>
<td>2</td>
<td>—</td>
<td>1.0</td>
<td>—</td>
<td>1.0</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td></td>
<td>1</td>
<td>—</td>
<td>0.0</td>
<td>—</td>
<td>1.1</td>
</tr>
</tbody>
</table>

- The same random instances for $3\text{Col}$
- Solved with blind search
- Significantly worse than LAMA’11. Thus, these problems are non-trivial
- **Conjecture:** LAMA’11 solves these instances because of implicit serialization of subgoals enforced by the multiple queues
Example: SAT and UNSAT

Defined over vocabulary $\sigma = \langle P^2, N^2 \rangle$ where:

- $P(x, y)$ tells that variable $x$ appears positive in clause $y$
- $N(x, y)$ tells that variable $x$ appears negative in clause $y$

\[
\Psi_{\text{SAT}} = (\exists T)(\forall y)(\exists x)[(P(x, y) \land T(x)) \lor (N(x, y) \land \neg T(x))]
\]

s.o. unary relation used to encode guessed assignment

\[
\Psi_{\text{UNSAT}} = (\forall T)(\exists y)(\forall x)[(P(x, y) \Rightarrow \neg T(x)) \land (N(x, y) \Rightarrow T(x))]
\]

s.o. unary relation used to encode all assignments
Example: SAT

\[ \Psi_{\text{SAT}} = (\exists T)(\forall y)(\exists x)[(P(x, y) \land T(x)) \lor (N(x, y) \land \neg T(x))] \]

Instance: \((x_0 \lor \neg x_1 \lor x_2) \land (\neg x_0 \lor \neg x_2) \land (\neg x_0 \lor x_1)\)

Instance is satisfiable with model \(\{\neg x_0, \neg x_1, \neg x_2\}\)

STRIPS plan:

```
(begin-proof)
(end-guess-T)
(prove-and-2 var0 var1)
(prove-or-2 var0 var1)
(prove-exists var0)
(prove-forall-base var0)
(prove-and-2 var1 var0)
(prove-or-2 var1 var0)
(prove-exists var1)
(prove-forall-induc var0 var1)
(prove-and-2 var2 var0)
(prove-or-2 var2 var0)
(prove-exists var2)
(prove-forall-induc var1 var2)
(prove-so-exist-T var2)
(prove-goal)
```
Example: UNSAT

\[ \Psi_{\text{UNSAT}} = (\forall T)(\exists y)(\forall x)[(N(x, y) \Rightarrow T(x)) \land (P(x, y) \Rightarrow \neg T(x))] \]

Instance: 
\[ (x_0) \land (\neg x_0 \lor \neg x_1) \land (\neg x_0 \lor x_1) \]

Instance is unsatisfiable

STRIPS plan:

```
(begin-proof) (prove-or_2_2 var2 var1) (prove-or_1_2 var1 var0)
(init-so-forall-T) (prove-and var2 var1) (prove-or_2_1 var1 var0)
(prove-or_1_1 var0 var0) (prove-forall_induc var2 var0 var1)
(prove-and var0 var0) (prove-exists var2 var1)
(prove-or_2_2 var0 var0) (prove-or_1_2 var1 var1)
(prove-forall_base var0 var0) (prove-and var1 var0)
(prove-or_1_1 var0 var1) (prove-or_2_1 var1 var0)
(prove-exists var1 var1) (prove-and var0 var0)
(change_for_coin_T var0) (prove-or_2_2 var0 var0)
(prove-or_1_2 var0 var1) (prove-forall_induc var1 var0 var1)
(prove-or_2_1 var0 var1) (prove-exists var0 var1)
(prove-and var0 var0) (one_plus_one_0_T var0 var1)
(prove-forall_base var0 var0) (prove-or_1_2 var0 var1)
(prove-or_2_1 var0 var1) (prove-forall_induc var1 var0 var1)
(prove-and var1 var1) (prove-or_2_1 var1 var0)
(prove-or_1_1 var2 var1) (prove-forall_base var2 var0 var0)
(prove-exists var0 var0) (prove-or_2_1 var2 var0)
(prove-forall_induc var0 var0 var1) (prove-or_1_2 var0 var1)
(prove-or_2_1 var0 var1) (prove-forall_base var0 var0)
(prove-and var0 var0) (prove-or_1_1 var2 var1)
(prove-or_2_1 var2 var0) (prove-exists var0 var0)
(prove-or_2_1 var2 var0) (prove-forall_base var2 var0 var0)
(prove-or_1_1 var2 var1) (prove-or_2_2 var0 var1)
(prove-and var0 var0) (change_for_coin_T var0)
(prove-or_2_2 var0 var0) (prove-or_1_2 var2 var0)
(prove-exists var0 var1) (zero_plus_one_T var1)
(prove-forall_base var0 var0) (prove-or_1_2 var0 var1)
(prove-or_2_1 var0 var1) (prove-or_2_2 var0 var0)
(prove-and var0 var0) (zero_plus_one_T var0)
(prove-or_2_1 var1 var1) (change_for_coin_T var0)
(prove-or_2_1 var1 var1) (prove-goal)
```


Idea of Translation

For FO formulas (from ICAPS-11):

- fluents represent validity of subformulas where parameters stand for free variables
- operators establish validity of formulas (fluents) from validity of subformulas (inductively in structure of formulas)

For SO existential quantifiers (similar to ICAPS-11):

- plan chooses one interpretation of quantified symbol
- plan then moves and proves validity with chosen interpretation

For SO universal quantifiers:

- plan iterates over all interpretations of quantified symbol
- for each such interpretation, plan proves validity
Consider a unary relation $T$ and a universe with $n$ objects.

There are $2^n$ different interpretations of $T$ that can be identified with the $2^n$ different binary words of length $n$: 

$i$-th element belongs to $T$'s interpretation iff $i$-th bit in word is 1

Iterating over interpretations is done by iterating over such words.

The word is treated as a counter that starts at ‘00⋯0’ and is incremented until ‘11⋯1’ by adding 1.
How to Capture PSPACE

PSPACE = SO + TC

TC[Ψ](x, y) denotes connectivity on a graph defined by the formula Ψ(¯u, ¯v).

If we use the fluent F[Ψ](u, v) to denote the validity of Ψ(¯u, ¯v), then we can design actions to prove the validity of TC[Ψ](x, y).

Therefore, implementing TC[Ψ] in STRIPS is straightforward:

it is just finding a path on a graph whose edges are given by fluents F[Ψ](u, v).
Summary

- Extended the tool presented in ICAPS-11 so that:
  - targets the much bigger complexity class PH
  - implements a type system that permits more efficient translations
- Performed experiments and got interesting results for LAMA’11
- Tool can be used to design challenging benchmarks for planners
- Tool can be extended to target whole PSPACE without much work
Thanks. Questions?