Implementing a Relevance Tracker Module

Joachim Jansen¹, Bart Bogaerts²,¹, Jo Devriendt¹
Gerda Janssens¹, Marc Denecker¹

1 KU Leuven, Belgium, firstname.lastname@kuleuven.be
2 Aalto University, Finland, bart.bogaerts@aalto.fi

October 16th, 2016
Overview

- Background: SAT(ID)
- Background: Relevance for SAT(ID)
- Implementing Relevance
PC(ID), SAT(ID)

- SAT(ID) = satisfiability check of PC(ID)
- Propositional Calculus + Inductive Definitions
- PC(ID) encoding $\mathcal{T} = \{p_\mathcal{T}, \Delta\}$ (normal form)
- $p_\mathcal{T}$ is defined in $\Delta$; must hold for $\mathcal{T}$ to be satisfied.
- Relation with ASP: $p_\mathcal{T}$ is a single constraint, all atoms not defined in $\Delta$ are open (choice rules), $\Delta$ contains no recursion over negation (real definition)
PC(ID), SAT(ID)

- SAT(ID) = satisfiability check of PC(ID)
- Propositional Calculus + Inductive Definitions
- PC(ID) encoding $\mathcal{T} = \{p_{\mathcal{T}}, \Delta\}$ (normal form)
- $p_{\mathcal{T}}$ is defined in $\Delta$; must hold for $\mathcal{T}$ to be satisfied.
- Relation with ASP: $p_{\mathcal{T}}$ is a single constraint, all atoms not defined in $\Delta$ are open (choice rules), $\Delta$ contains no recursion over negation (real definition)

Example

- Choose edges and colors of nodes s.t.
  - node $b$ is reachable from $a$
  - every node reachable from $a$ is colored green

![Diagram of nodes a, b, c with edges and colors]
Example (continued)

\[ \Delta = \begin{cases} 
  p_T & \leftarrow reach_b \land constr_1 \land constr_2 \land constr_3. \\
  constr_1 & \leftarrow \neg reach_a \lor green_a. \\
  constr_2 & \leftarrow \neg reach_b \lor green_b. \\
  constr_3 & \leftarrow \neg reach_c \lor green_c. \\
  reach_a & . \\
  reach_b & \leftarrow case_1 \lor case_2. \\
  case_1 & \leftarrow reach_a \land edge_{a,b}. \\
  case_2 & \leftarrow reach_c \land edge_{c,b}. \\
  reach_c & \leftarrow reach_b \land edge_{b,c}. 
\end{cases} \]

- \( reach_x \) = node \( x \) is reachable from \( a \)
- \( constr_x \) = color constraints on node \( x \)
- \( green_x \) = node \( x \) is green
- \( edge_{x,y} \) = edge from \( x \) to \( y \) selected
Typically, a SAT(ID) solver searches for an assignment (true/false) to all atoms such that $\mathcal{T}$ is satisfied
Visualising the hierarchy

- $p_T$ (and-node)
- $r_b$ (or-node)
- $c_1$
- $c_2$
- $\neg r_c$
- $g_c$
- $r_a$
- $e_{a,b}$
- $\neg r_b$
- $\neg e_{b,c}$
Visualising the Search process
Visualising the Search process

$p_T$ propagation

$rb$

$ca_1$

$ra$

$ca_2$

$ea,b$

$neg rb$

$neg rc$

$bc$

$g_c$

$c_3$
Visualising the Search process

$p_T$

$C_3$

$r_b$

$C_1$

$C_2$

$\neg r_c$

$g_c$

$r_a$

$e_{a,b}$

$\neg r_b$

$\neg e_{b,c}$
Visualising the Search process

Diagram showing a search process with nodes labeled $r_b$, $ca_1$, $ca_2$, $\neg r_c$, $g_c$, and $\neg e_{b,c}$. The diagram includes a choice point labeled $p_T$. The process involves logical conditions such as $r_a$, $e_{a,b}$, and $\neg r_b$.
Visualising the Search process
Visualising the Search process

wait a minute...
Visualising the Search process

...not helping!
Visualising the Search process

could be any NP subproblem!
Justifications

- Defined by Denecker and De Schreye (1993) and Denecker, Brewka and Strass (2015)
- Intuitively, a literal is \textit{justified} given a partial assignment if there exists a (recursive) explanation why it must hold in terms of true open literals.
- Thus... it suffices to prove that $p$ is justified in some partial interpretation to conclude that $T$ is satisfiable.
Justifications

- Defined by Denecker and De Schreye (1993) and Denecker, Brewka and Strass (2015)

- Intuitively, a literal is *justified* given a partial assignment if there exists a (recursive) explanation why it must hold in terms of true open literals.

- If a literal is justified in a partial assignment, then there exists a model of $\Delta$ in which that literal holds.

- Thus... it suffices to prove that $p_T$ is *justified* in some partial interpretation to conclude that $T$ is satisfiable.
Searching assignment → searching justification
Searching assignment $\rightarrow$ searching justification
Searching assignment $\rightarrow$ searching justification

value = "fixed" for current branch
Relevance

Definition
Given a PC(ID) theory $\mathcal{T} = \{p_\mathcal{T}, \Delta\}$ and a partial interpretation $\mathcal{I}$, we inductively define the set of relevant literals, denoted $\mathcal{R}_{\mathcal{T},\mathcal{I}}$, as follows

- $p_\mathcal{T}$ is relevant if $p_\mathcal{T}$ is not justified,
- $l$ is relevant if $l$ is not justified and there exists some $l'$ such that $(l', l) \in dd_\Delta$ and $l'$ is relevant.
Relevant $\approx$ can help justify $p_T$

Diagram:
- $p_T$
- $r_b$
- $ca_1$
  - $r_a$
  - $e_{a,b}$
- $ca_2$
- $\neg r_c$
  - $\neg e_{b,c}$
- $c_3$
- $g_c$
Relevant $\approx$ can help justify $p_T$
Relevant $\approx$ can help justify $p_\tau$
Relevant ≈ can help justify $p_T$
Adjusting the Solver

- Decide only on Relevant literals.
- Stop search when \( p_T \) is justified
  - Guarantee that a two-valued solution can be generated efficiently
  - More tolerant to faulty choices of the solver
  - Expectation: less choices made by solver
Implementation

- How to keep track of justified literals?
- How to keep track of relevant literals?
Keeping track of justified literals

- For each defined atom \( p \), introduce a new atom \( j_p \).
- Intended interpretation: \( j_p \) is true (in a partial interpretation) iff \( p \) is justified; \( j_p \) is false iff \( \neg p \) is justified; \( j_p \) is unknown otherwise.
- Duplicate definition \( \Delta \) to a new definition \( \Delta' \), obtained by replacing each defined atom \( p \) by \( j_p \) (note: open literals remain).
- Modify solver: forbidden to make choices on \( j_p \).
- **Claim**: after the standard propagation is executed, \( j_p \) satisfies the “intended interpretation” above.
Theorem
Let $\Delta$ be a (total) definition and $\mathcal{I}$ a partial interpretation in which all defined symbols of $\Delta$ are interpreted as $u$. Let $l$ be a defined literal in $\Delta$. In this case $l$ is justified in $\mathcal{I}$ if and only if $l$ is derivable by unit propagation on the completion of $\Delta$ and unfounded set propagations.
Keeping track of justified literals

- Without major modifications to the solver, we obtain a method to keep track of justified literals.
- Only modification: do not make choices on certain atoms.
Recall:

Definition
Given a PC(ID) theory $\mathcal{T} = \{\rho_\mathcal{T}, \Delta\}$ and a partial interpretation $\mathcal{I}$, we inductively define the set of relevant literals as follows

- $\rho_\mathcal{T}$ is relevant if $\rho_\mathcal{T}$ is not justified,
- $l$ is relevant if $l$ is not justified and there exists some $l'$ such that $(l', l) \in dd_\Delta$ and $l'$ is relevant.
Keeping track of relevant literals

- For each relevant literal (except $p_T$), we maintain one relevant parent in $dd_{\Delta}$: the reason why this literal is relevant.
- Thus, we maintain a subgraph of $dd_{\Delta}$.
- We incrementally update this subgraph (as the justification status of certain literals changes).
- Biggest challenge: keeping this graph acyclic. (how to choose the “right” parent)
Keeping track of relevant literals

- For each relevant literal (except $p_T$), we maintain one relevant parent in $dd_\Delta$: the reason why this literal is relevant.
- Thus, we maintain a subgraph of $dd_\Delta$.
- We incrementally update this subgraph (as the justification status of certain literals changes)
- Biggest challenge: keeping this graph acyclic. (how to choose the “right” parent)
- Turns out... this cycle detection is the same problem as tackled in unfounded set propagators.
- Only difference: works on a (slightly) different graph.
Keeping track of relevant literals

- In the paper, we also detail the used data structures and an event-driven implementation
Experiment Setup (1)

- Problems from previous ASP competitions
- Solver = Minisatid, Heuristic = VSIDS
Experiment Setup (1)

- Problems from previous ASP competitions
- Solver = Minisatid, Heuristic = VSIDS
- Measuring
  - Ratio of irrelevant decisions (%)
  - Ratio of conflicts originating from irrelevant decisions (%)
Experimental Results (1)

<table>
<thead>
<tr>
<th>Problem</th>
<th>% Irr. Decisions</th>
<th>% Irr. Conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP</td>
<td>27.37%</td>
<td>36.99%</td>
</tr>
<tr>
<td>NQueens</td>
<td>22.55%</td>
<td>0.43%</td>
</tr>
<tr>
<td>PPM</td>
<td>22.93%</td>
<td>4.98%</td>
</tr>
<tr>
<td>Sokoban</td>
<td>48.20%</td>
<td>0.96%</td>
</tr>
<tr>
<td>Solitaire</td>
<td>13.32%</td>
<td>3.95%</td>
</tr>
<tr>
<td>SM</td>
<td>96.40%</td>
<td>0.01%</td>
</tr>
<tr>
<td>Visit All</td>
<td>15.02%</td>
<td>16.45%</td>
</tr>
</tbody>
</table>
Experiment Setup (2)

- Problems from previous ASP competitions
- Solver = Minisatid, Heuristic = VSIDS
Experiment Setup (2)

- Problems from previous ASP competitions
- Solver = Minisatid, Heuristic = VSIDS
- Measuring
  - Number of decisions (\#)
  - Number of conflicts (\#)
Experimental Results (2)

# Decisions Made

- Without Relevance
- With Relevance

Instances vs. # Decisions Made
Experimental Results (2)

The graph illustrates the number of conflicts over instances. The x-axis represents the number of instances, while the y-axis shows the number of conflicts on a logarithmic scale. Two curves are plotted:
- The red curve labeled 'Without Relevance' represents the number of conflicts without relevance consideration.
- The green curve labeled 'With Relevance' indicates the number of conflicts with relevance consideration.

As the number of instances increases, both curves show an increasing trend, with the 'With Relevance' curve consistently lower than the 'Without Relevance' curve, indicating a lower number of conflicts when relevance is taken into account.
Take-away messages

- Exploit problem hierarchy using Relevance
Take-away messages

- Exploit problem hierarchy using \textit{Relevance}
- Preliminary promising results: fewer \textit{decisions}
- A relevance tracker can be \textit{implemented} reusing existing methods:
  - Justification status: unit propagation and unfounded set propagation
  - Relevance status: unfounded set algorithms
Questions?