Implementing a Relevance Tracker Module

Joachim Jansen[1], Bart Bogaerts[2,1], Jo Devriendt[1]
Gerda Janssens[1], Marc Denecker[1]

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

October 16th, 2016

KU LEUVEN

Aalto University
School of Science

Overview

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

PC(ID), SAT(ID)

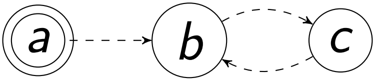
- SAT(ID) = satisfiability check of PC(ID)
- **P**ropositional **C**alculus + **I**nductive **D**efinitions
- PC(ID) encoding $\mathcal{T} = \{p_{\mathcal{T}}, \Delta\}$ (normal form)
- $p_{\mathcal{T}}$ is defined in Δ; must hold for $\mathcal{T}$ to be satisfied.
- Relation with ASP: $p_{\mathcal{T}}$ is a single constraint, all atoms not defined in Δ are *open* (choice rules), Δ contains no recursion over negation (real definition)

PC(ID), SAT(ID)

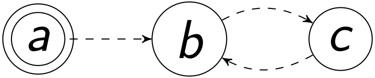
- SAT(ID) = satisfiability check of PC(ID)
- **P**ropositional **C**alculus + **I**nductive **D**efinitions
- PC(ID) encoding $\mathcal{T} = \{p_{\mathcal{T}}, \Delta\}$ (normal form)
- $p_{\mathcal{T}}$ is defined in Δ; must hold for $\mathcal{T}$ to be satisfied.
- Relation with ASP: $p_{\mathcal{T}}$ is a single constraint, all atoms not defined in Δ are *open* (choice rules), Δ 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



Example (continued)



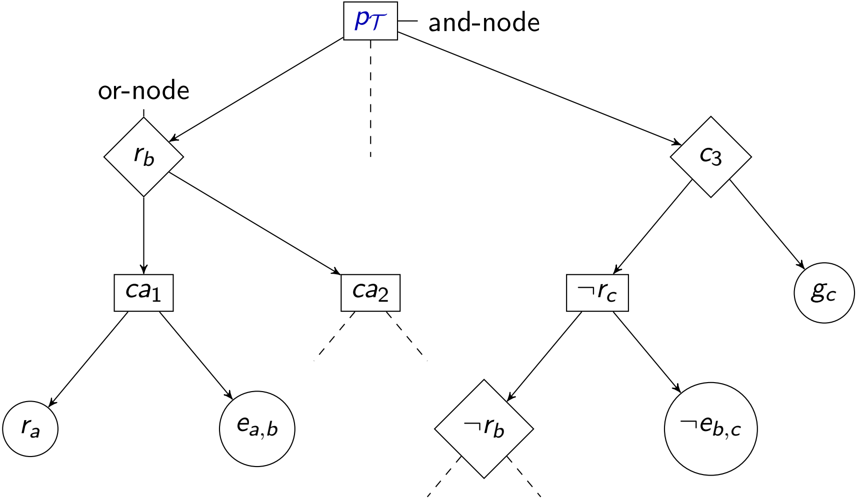
$$\Delta = \left\{ \begin{array}{rcl} {\color{blue} p_{\mathcal{T}}} & \leftarrow & reach_b \ \wedge \ constr_1 \ \wedge \ constr_2 \wedge constr_3. \\ constr_1 & \leftarrow & \neg reach_a \vee green_a. \\ constr_2 & \leftarrow & \neg reach_b \vee green_b. \\ constr_3 & \leftarrow & \neg reach_c \vee green_c. \\ reach_a & . & \\ reach_b & \leftarrow & case_1 \vee case_2. \\ case_1 & \leftarrow & reach_a \wedge edge_{a,b}. \\ case_2 & \leftarrow & reach_c \wedge edge_{c,b}. \\ reach_c & \leftarrow & reach_b \wedge edge_{b,c}. \end{array} \right\}$$

- $reach_x$ = 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

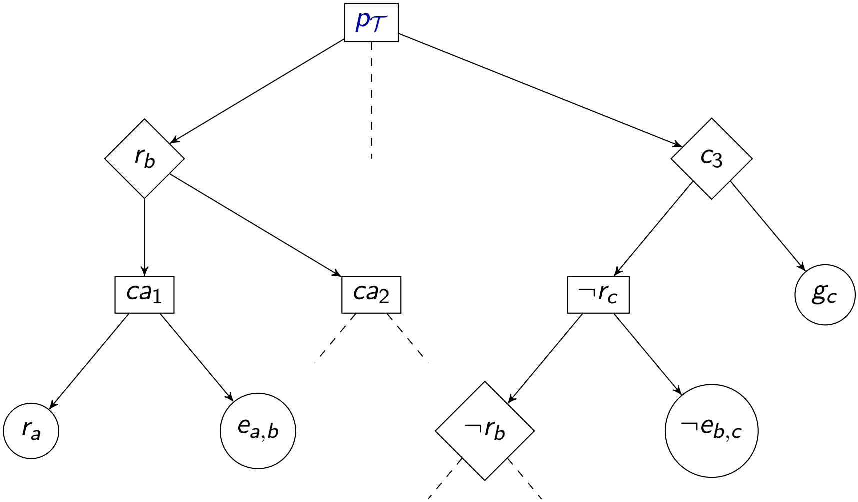
SAT(ID) solver

Typically, a SAT(ID) solver searches for an *assignment (true/false) to all atoms* such that $\mathcal{T}$ is satisfied

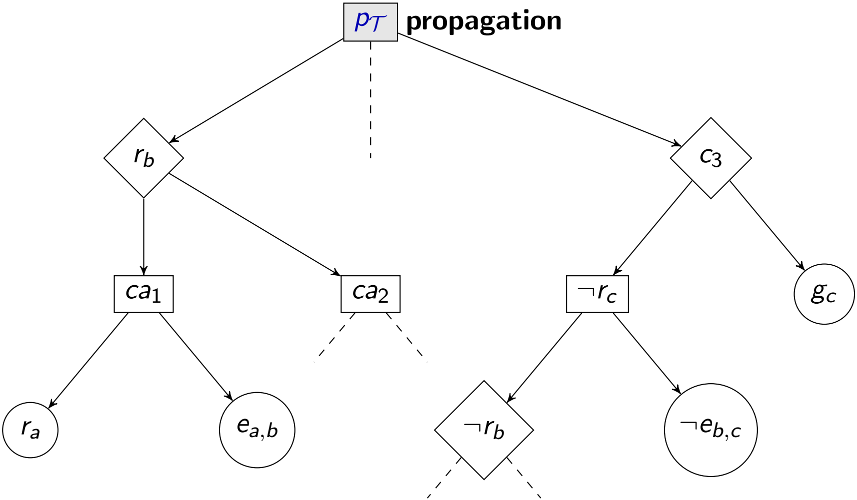
Visualising the hierarchy



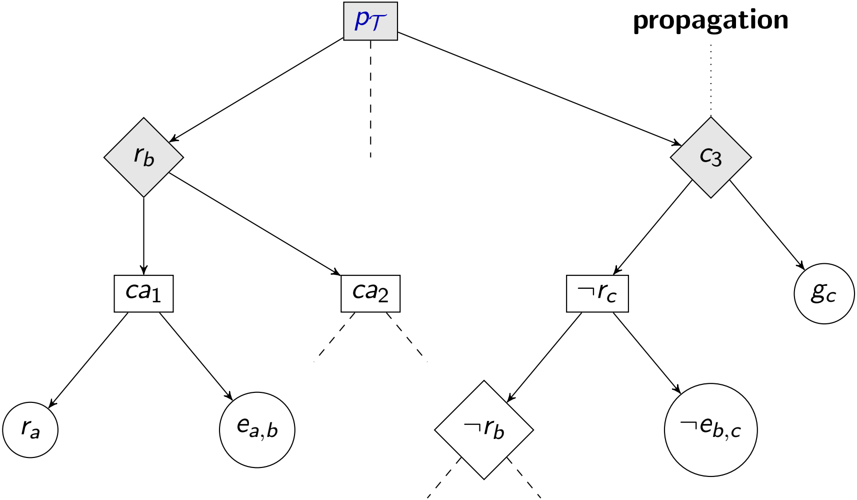
Visualising the Search process



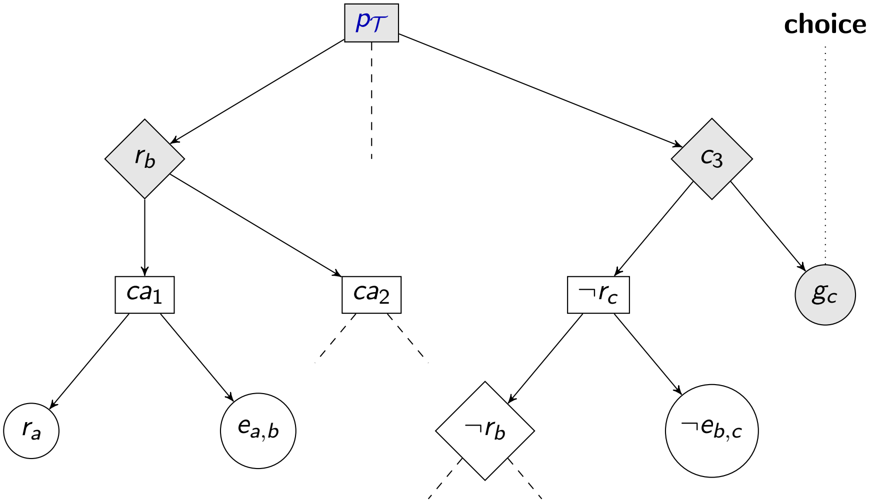
Visualising the Search process



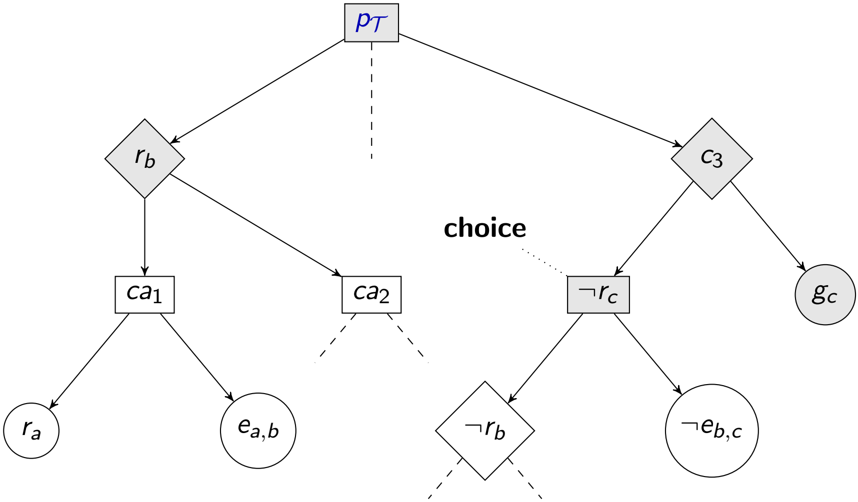
Visualising the Search process



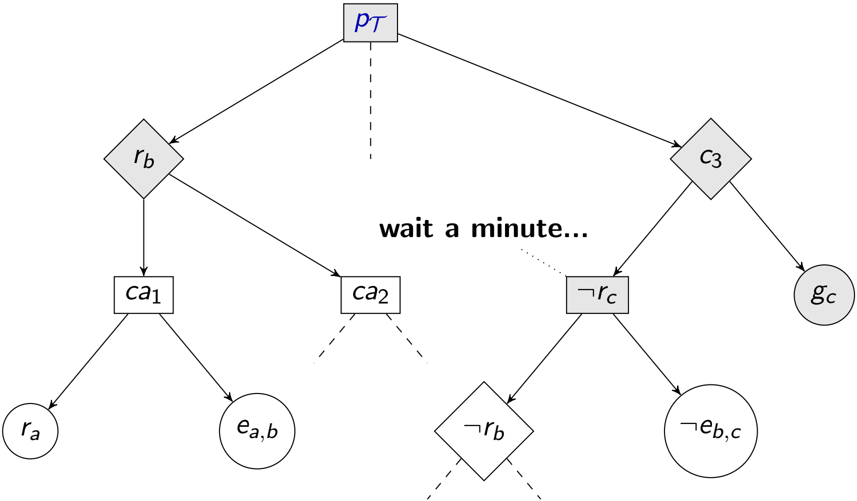
Visualising the Search process



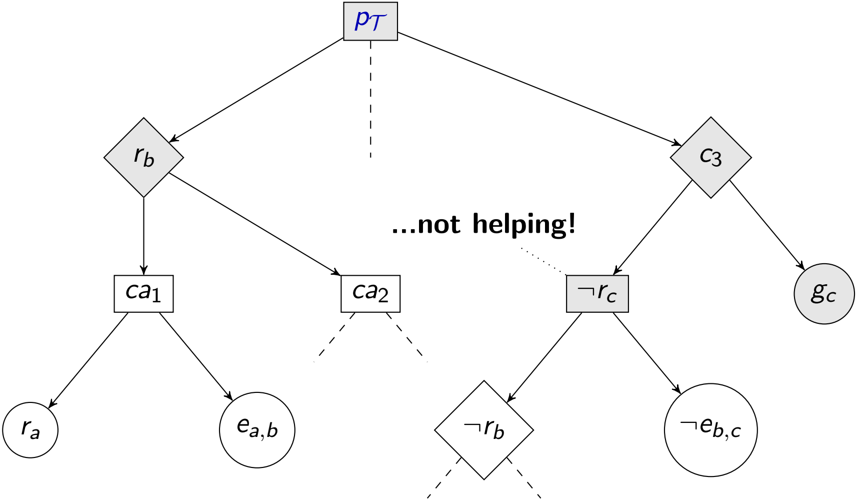
Visualising the Search process



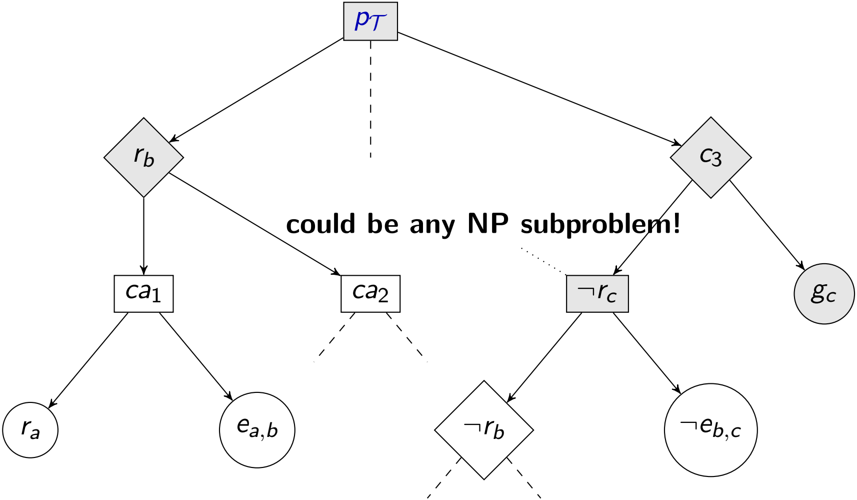
Visualising the Search process



Visualising the Search process



Visualising the Search process



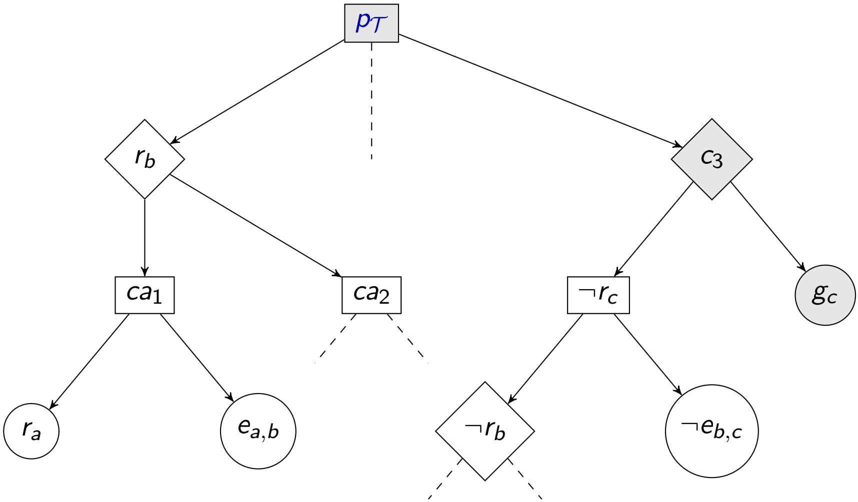
Justifications

- 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.

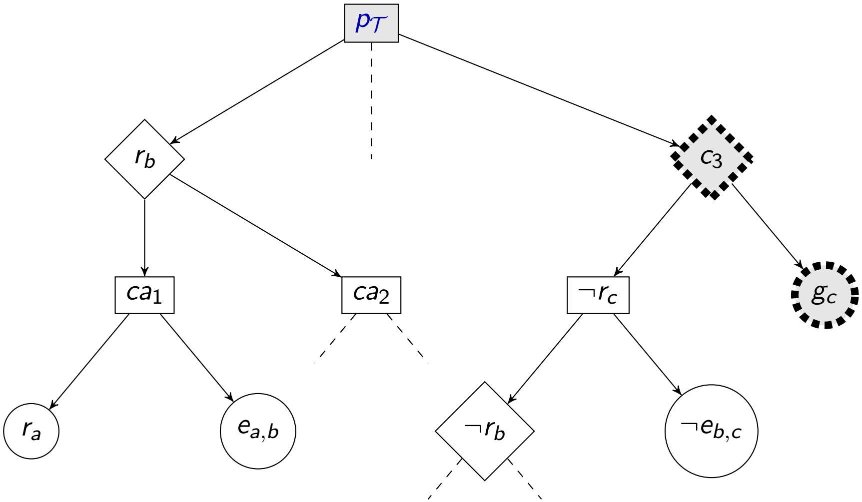
Justifications

- 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 Δ in which that literal holds.
- Thus... it suffices to prove that $p_{\mathcal{T}}$ is *justified* in some partial interpretation to conclude that $\mathcal{T}$ is satisfiable.

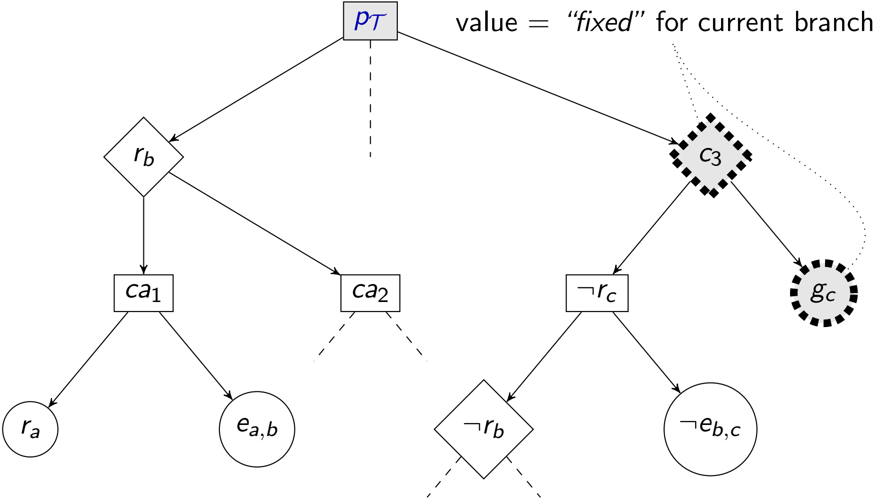
Searching assignment $\rightarrow$ searching $\boxed{\textit{justification}}$



Searching assignment $\rightarrow$ searching $\boxed{\textit{justification}}$



Searching assignment $\rightarrow$ searching *justification*



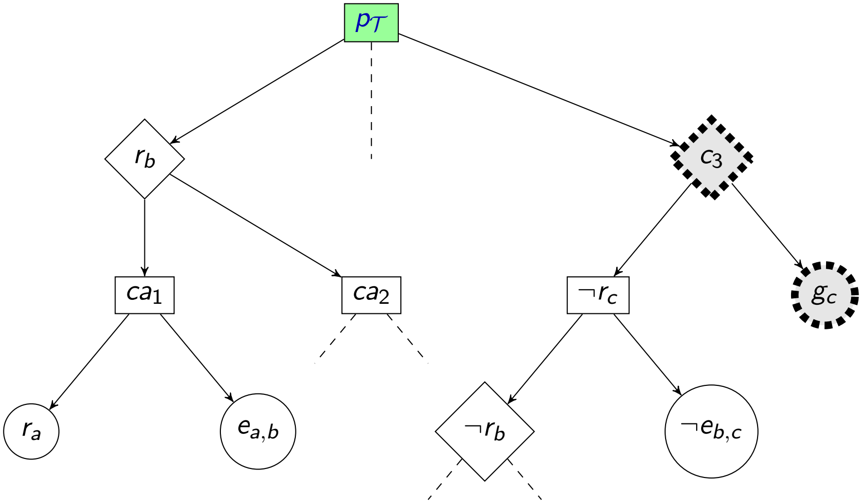
Relevance

Definition

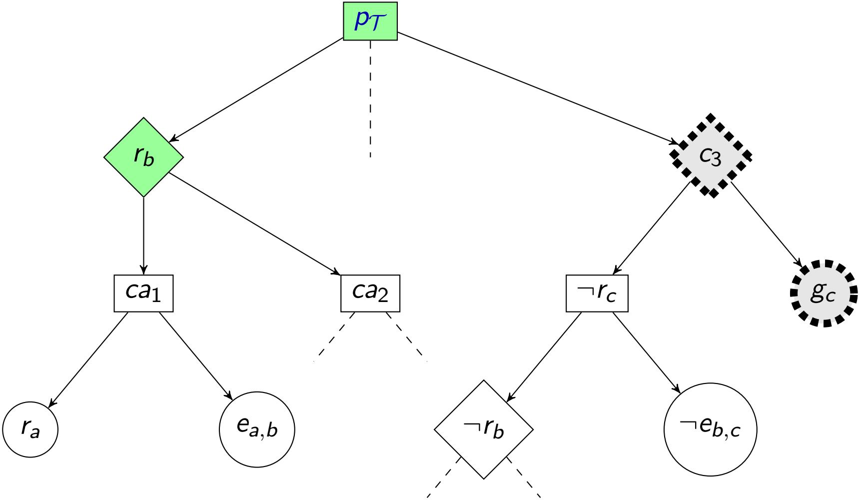
Given a PC(ID) theory $\mathcal{T} = \{p_{\mathcal{T}}, \Delta\}$ and a partial interpretation $\mathcal{I}$, we inductively define the set of relevant literals, denoted $\mathcal{R}_{\mathcal{T},\mathcal{I}}$, as follows

- $p_{\mathcal{T}}$ 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.

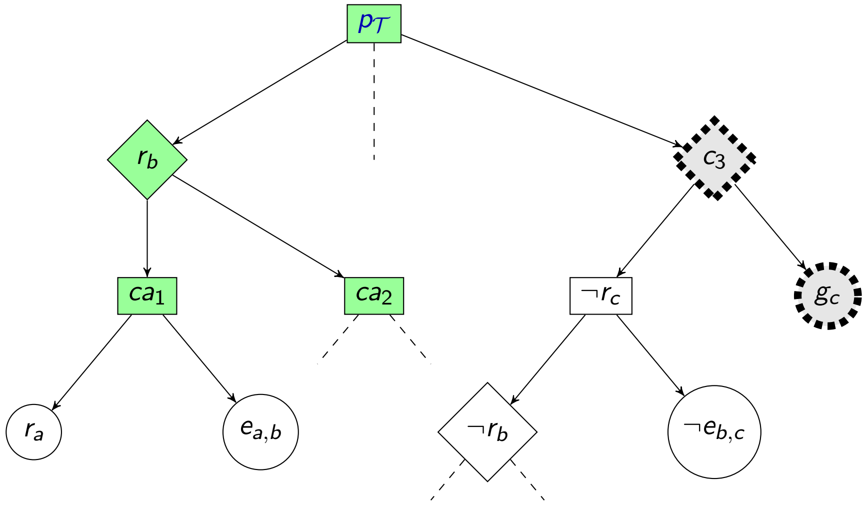
Relevant $\approx$ can help justify $p_{\mathcal{T}}$



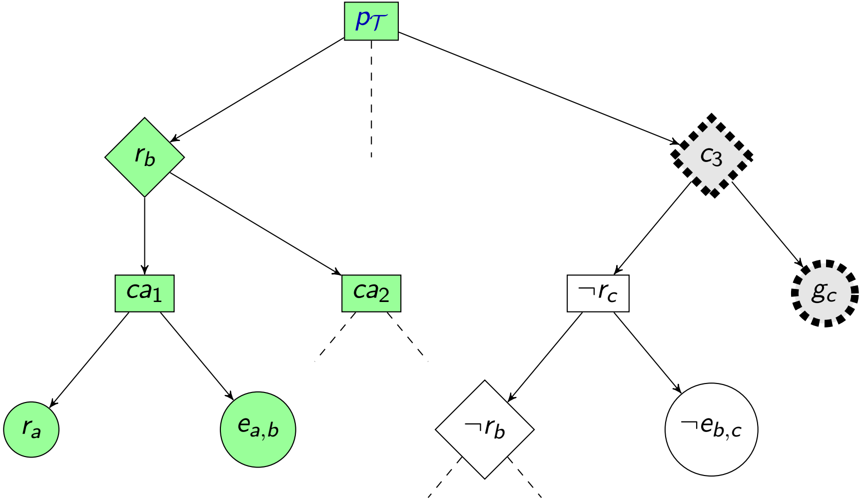
Relevant $\approx$ can help justify $p_{\mathcal{T}}$



$\boxed{\textit{Relevant}}$ $\approx$ can help justify $p_{\mathcal{T}}$



$\boxed{\textit{Relevant}}$ $\approx$ can help justify $p_{\mathcal{T}}$



Adjusting the Solver

- Decide only on $\boxed{\textit{Relevant}}$ literals.
- Stop search when $p_{\mathcal{T}}$ 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 Δ to a new definition Δ', obtained by a 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.

Keeping track of justified literals

Theorem

Let Δ be a (total) definition and $\mathcal{I}$ a partial interpretation in which all defined symbols of Δ are interpreted as $\mathbf{u}$. Let l be a defined literal in Δ. In this case l is justified in $\mathcal{I}$ if and only if l is derivable by unit propagation on the completion of Δ 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.

Keeping track of relevant literals

Recall:

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 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.

Keeping track of relevant literals

- For each relevant literal (except $p_{\mathcal{T}}$), we maintain one *relevant parent in* dd_Δ: the *reason* why this literal is relevant.
- Thus, we maintain a subgraph of dd_Δ.
- 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 $p_{\mathcal{T}}$), we maintain one *relevant parent in* dd_{Δ}: the *reason* why this literal is relevant.
- Thus, we maintain a subgraph of dd_{Δ}.
- 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)

Problem	% Irr. **Decisions**	% Irr. **Conflicts**
HP	27.37%	36.99%
NQueens	22.55%	0.43%
PPM	22.93%	4.98%
Sokoban	48.20%	0.96%
Solitaire	13.32%	3.95%
SM	96.40%	0.01%
Visit All	15.02%	16.45%

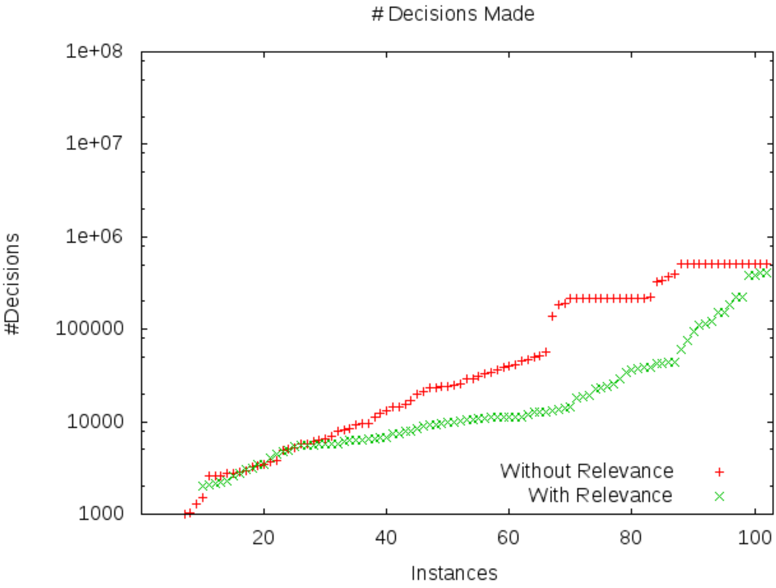
Experiment Setup (2)

- Problems from previous ASP competitions
- Solver = Minisatid, Heuristic = VSIDS

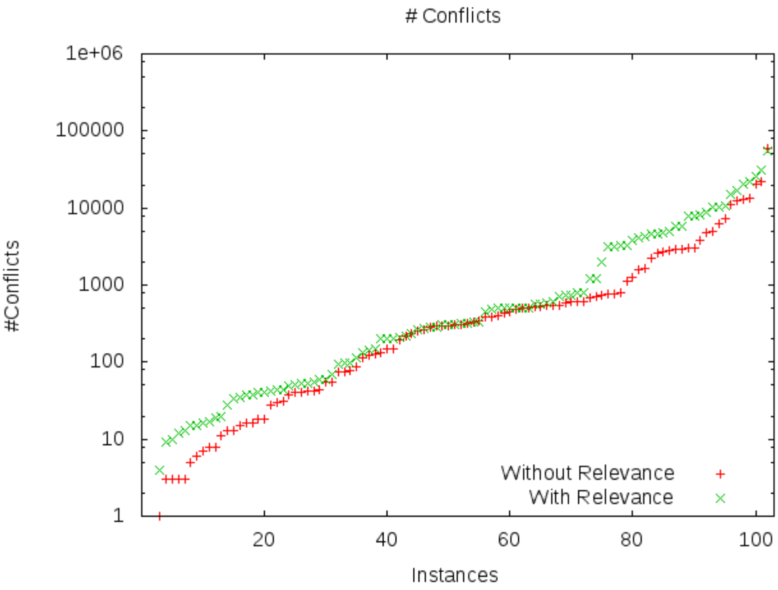
Experiment Setup (2)

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

Experimental Results (2)



Experimental Results (2)



Take-away messages

- Exploit problem hierarchy using $\boxed{\textit{Relevance}}$

Take-away messages

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

Questions?