# Declarative Extension of SAT Solvers with New Propagators

Shahab Tasharrofi

Joint work with Tomi Janhunen and Eugenia Ternovska

Department of Information and Computer Science, Aalto University

Computational Logic Day

December 8, 2015

# Introduction

Several lines of work involve:

High-level declarative languages

for specifying search problems

(e.g. find a Hamiltonian Path in a graph)

plus

Solvers for use in practice

E.g.: Constraint Modelling Languages (Essence, Zinc, . . . )
ASP, IDP system, Enfragmo, NP-Spec, etc.

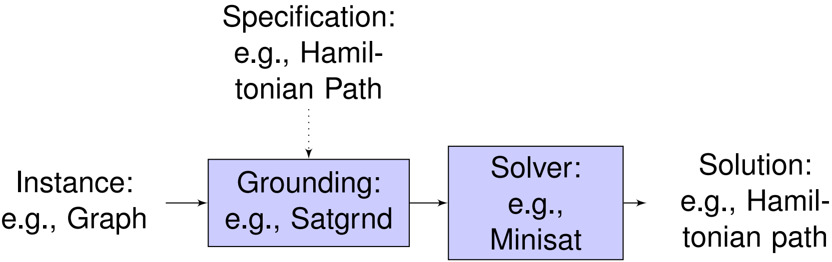
# Example: Hamiltonian Path Specification

The languages possess nice features: arithmetic, aggregates, induction(sometimes)
that allow one to write natural specifications of many problems

**Example:** FO+Arithmetic specification of Hamiltonian Path:

$$\forall u \forall v\ (next(u,v) \Rightarrow arc(u,v)).$$
$$\forall u \forall v \forall v'\ (next(u,v) \wedge next(u,v') \Rightarrow v = v').$$
$$\forall u \forall u' \forall v\ (next(u,v) \wedge next(u',v) \Rightarrow u = u').$$
$$\forall u\ (reach(0,u) \Leftrightarrow u = s).$$
$$\forall n\, \forall u\ (reach(n+1,u) \Leftrightarrow reach(n,u) \vee \exists v\ (reach(n,v) \wedge next(v,u))$$
$$\forall u\ (reach(|nodes|,u)).$$

# Existing Practice




So, a grounder takes the specification and an instance of a problem to generate a ground instance that is solved by a black-box solver

BUT, what if:

- the black-box solver gets stuck?
- or, solver's inference methods are too slow for our problem?

# Our Goals

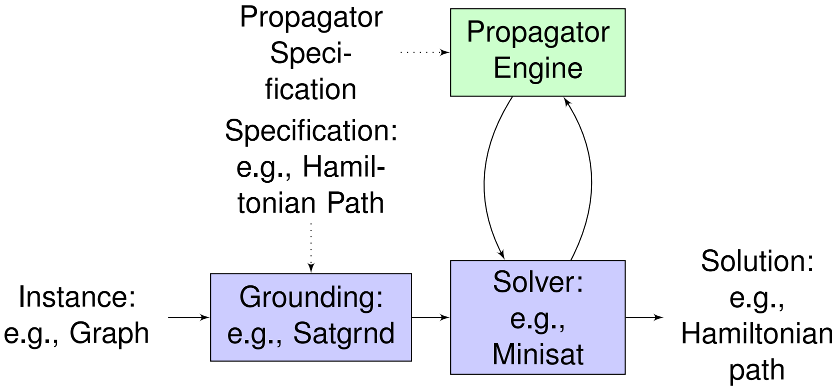
Our final long-term goal is to:

> Develop methods and foundations for declarative specification and automatic construction of optimized problem-specific solvers

In this paper, our goal is to:

> Develop methods to specify problem-specific propagators and to automatically modify a SAT solver to use such mechanisms

# Our Approach




So, we can:

- specify problem-specific propagators, and,
- incorporate those propagators into a solver

# Language $P[R]$ to Specify Problems Modulo Reasonings

Syntactically: $P[R]$ programs take the form of $(\phi, \{\psi_1, \ldots, \psi_n\})$ with $\phi$, $\psi_1$, …, and $\psi_n$ being first order programs
Here, $\phi$ is the main program and $\psi_1, \ldots, \psi_n$ are propagators

Propagator vocabulary of $\psi_i$ is $\tau_i = \mathit{vocab}(\psi_i) \setminus \mathit{vocab}(\phi)$

Semantically: $P[R]$ program $(\phi, \{\psi_1, \ldots, \psi_n\})$ is equivalent to following second-order formula:

$$\phi \wedge \bigwedge_{1 \leq i \leq n} \neg \exists \tau_i \, (\psi_i)$$

The role of propagators is to specify undesirable models.

# Example: Reachability Propagator for Hamiltonian Path

Main problem $\phi$ asserts that *next* is a subset of arcs *without any two arcs with the same source or the same destination*:

$$\phi := \left\{ \begin{array}{c} \forall x, y\ (next(x, y) \to arc(x, y)). \\ \forall x, y, y'\ (next(x, y) \wedge next(x, y') \to y = y'). \\ \forall x, x', y\ (next(x, y) \wedge next(x', y) \to x = x'). \\ \forall x\ (\neg next(x, s)). \end{array} \right\}$$

Propagator $\psi$ checks if a cut separates *s* from some other node

$$\psi := \left\{ \begin{array}{c} cut(s). \qquad \exists x\ \neg cut(x). \\ \forall x, y\ (next(x, y) \wedge cut(x) \to cut(y)). \end{array} \right\}$$

*P*[*R*] specification $(\phi, \{\psi\})$ finds Hamiltonian paths because propagator $\psi$ guarantees reachability of all nodes from *s*

# Grounding $P[R]$ Specifications

Underapproximate translation of propagator $\psi$, denoted as $[\![\psi]\!]^\varepsilon$, is formula $\psi'$ so that

- Positive occurrences of $R \in \varepsilon$ is replaced by $R_l$
- Negative occurrences of $R \in \varepsilon$ is replaced by $R_u$

Grounding of $P[R]$ specification $(\phi, \{\psi_1, \ldots, \psi_n\})$ w.r.t. $\mathcal{A}$, denoted by $Gnd((\phi, \{\psi_1, \ldots, \psi_n\}); \mathcal{A})$, is

$$(Gnd(\phi; \mathcal{A}), \{Gnd([\![\psi_1]\!]^\varepsilon; \mathcal{A}), \ldots, Gnd([\![\psi_n]\!]^\varepsilon; \mathcal{A})\})$$

where $\varepsilon = vocab(\phi) \setminus vocab(\mathcal{A})$

# Theoretical Foundations

If $\mathcal{B}$ is a $(\sigma \cup \varepsilon)$-structure that partially interprets $\varepsilon$, its 2-valued representation, denoted as $\mathcal{B}_{2v}$, is a $(\sigma \cup \varepsilon_l \cup \varepsilon_u)$-structure so that

- For $R \in \sigma$, $R^{\mathcal{B}_{2v}} = R^{\mathcal{B}}$,
- For $R \in \varepsilon$, $R_l^{\mathcal{B}_{2v}}$ is the lowerbound of $R^{\mathcal{B}}$, and $R_u^{\mathcal{B}_{2v}}$ is the upperbound of $R^{\mathcal{B}}$, i.e.,

$$\begin{aligned} R(\bar{x}) = \mathit{true} &\Rightarrow R_l(\bar{x}) = R_u(\bar{x}) = \mathit{true} \\ R(\bar{x}) = \mathit{unknown} &\Rightarrow R_l(\bar{x}) = \mathit{false}, R_u(\bar{x}) = \mathit{true} \\ R(\bar{x}) = \mathit{false} &\Rightarrow R_l(\bar{x}) = R_u(\bar{x}) = \mathit{false} \end{aligned}$$

Theorem: For partial str. $\mathcal{B}$:
$\mathcal{B}_{2v} \models \exists \tau \, [\![\psi_i]\!]^{\varepsilon} \Rightarrow$ all extensions of $\mathcal{B}$ falsify $(\phi, \{\psi_1, \ldots, \psi_n\})$

# Example: Hamiltonian Path

Underapproximate translation of reachability propagator $\psi$ is

$$[\![\psi]\!]^{\{next\}} := \left\{ \begin{array}{c} cut(s). \qquad \exists x \neg cut(x). \\ \forall x, y\ (next_u(x, y) \wedge cut(x) \rightarrow cut(y)). \end{array} \right\}$$

Here, if $\mathcal{B}_{2v} \models \exists cut\ ([\![\psi]\!]^{\{next\}}) \Rightarrow$ (no matter how many unknown arcs in $next^{\mathcal{B}}$ are made true) $s$ cannot reach all nodes using $next^{\mathcal{B}}$

# Solving $P[R]$ Programs: SAT-to-SAT

For a given propagator $\psi$, we want to incorporate $\psi$ into a SAT solver for the main problem $\phi$

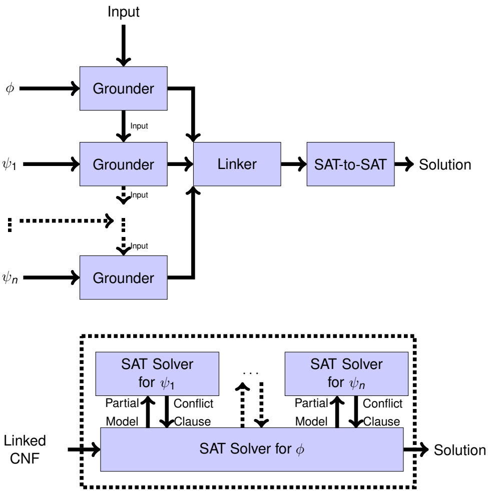
Note that a SAT solver state can be viewed as a partial structure

$\Rightarrow$ (by previous theorem) In state $\mathcal{B}$ of SAT solver for $\phi$, if $\mathcal{B}_{2v}$ satisfies $\exists\tau\ [\![\psi]\!]^{\varepsilon}$ then SAT solver for $\phi$ should backtrack (because there cannot be a model in this branch of search)

Also, note that, using a (new) SAT solver, we can check if a state $\mathcal{B}$ satisfies some propagator specification

$\Rightarrow$ The idea is to have a SAT solver for $\phi$ that communicates with other SAT solvers for $\psi_i$'s (hence the name SAT-to-SAT)

# $P[R]$ + SAT-to-SAT: In a Glance



# SAT-to-SAT: Conflict Clauses and Triggers

Implemented a propagator that, given state $\mathcal{B}$ of external solver:

- Computes 2-valued representation $\mathcal{B}_{2v}$ of $\mathcal{B}$,
- Runs an internal SAT solver to check if $\mathcal{B}_{2v} \models \exists\tau \; [\![\psi_i]\!]^{\varepsilon}$,
- If so, it returns $\langle SAT, J\rangle$ with $J$ a set of literals such that $\mathcal{B}' \models_3 \bigwedge_{x\in J} x \Rightarrow \mathcal{B}'_{2v} \models \exists\tau \; [\![\psi_i]\!]^{\varepsilon}$, and
- If not, it returns $\langle UNSAT, J\rangle$ with $J$ a set of literals such that $\mathcal{B}' \not\models_3 \bigvee_{x\in J} x \Rightarrow \mathcal{B}'_{2v} \not\models \exists\tau \; [\![\psi_i]\!]^{\varepsilon}$

Lazy conflict clause generation: If internal solver returned $\langle SAT, J\rangle$, add clause $\bigvee_{x\in J} \neg x$ to external solver
Triggers: If internal solver returned $\langle UNSAT, J\rangle$, do not run it again until external solver has assigned some $x \in J$ to true

Theorem: SAT-to-SAT algorithm with clause generation and triggers as above is sound and complete for $P[R]$ specifications

# Example: Hamiltonian Path with Reachability

For propagator $\psi$ in Hamiltonian path:

Conflict clauses always take the following form:

$$\bigvee_{u \in cut, v \notin cut, arc(u,v)} next(u, v)$$

where

- *cut* is a proper subset of nodes including starting node *s*, and
- $next(u, v)$ is assigned to false whenever $u \in cut$ and $v \notin cut$

Triggers will always be of the form
$T = \{\neg next(u_1, v_1), \ldots, \neg next(u_k, v_k)\}$
where $T$ contains a rooted spanning tree of $\mathcal{G}$ with root $s$

# Experiments: Hamiltonian Path

| Hamiltonian Path Instances (15m time limit) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Size | Total Inst. | Glucose Direct Enc. | | SAT-to-SAT | | | |
| | | | | Reachability Enc. | | Acyclicity Enc. | |
| | | # | Time | # | Time | # | Time |
| 50 | 20 | 20 | 4.85s | **20** | **0.02s** | **20** | **0.02s** |
| 100 | 20 | 4 | 390s | **20** | **0.13s** | 20 | 0.63s |
| 150 | 20 | 0 | — | **20** | **1.14s** | 20 | 7.52s |
| 200 | 20 | 0 | — | **20** | **9.00s** | 20 | 74.0s |
| 250 | 20 | 0 | — | **20** | **82.3s** | 18 | 283s |
| 300 | 20 | 0 | — | **9** | **288s** | 5 | 639s |

Table: Solving Hamiltonian path using SAT-to-SAT on two different encodings plus using Glucose on a direct encoding.

# Future Directions

Upcoming soon (with Bart and Tomi):

- Solving QBF: i.e., arbitrary levels of nesting in SAT-to-SAT
- Generating KR solvers from their semantics: i.e., a second-order front-end for SAT-to-SAT

And, later:

- Developing methods to reason about a solver's state and history, e.g., decision variables and their levels or propagated variables and their reasons
- Developing foundations of temporal reasoning about solving process

# Thank You

Questions?

# Example: Acyclicity Propagator for Hamiltonian Path

*P*[*R*] Specification $(\phi', \{\psi'\})$ also specifies Hamiltonian paths:

$$\phi' := \left\{ \begin{array}{l} \forall x, y\ (next(x, y) \rightarrow arc(x, y)). \\ \forall x, y, y'\ (next(x, y) \wedge next(x, y') \rightarrow y = y'). \\ \forall y\ (y \neq s \rightarrow \exists x\ (next(x, y))). \end{array} \right\}$$

$$\psi' := \left\{ \begin{array}{l} \forall y\ (cycle(y) \rightarrow \exists x\ (cycle(x) \wedge next(x, y))). \\ \exists x\ cycle(x). \end{array} \right\}$$

Here, propagator $\psi'$ checks if *next* has become cyclic
$(\phi', \{\psi'\})$ specifies Hamiltonian Paths because *next* has to be an acyclic path of size $n - 1$