The OpenSMT Solver

Conference Paper · March 2010

DOI: 10.1007/978-3-642-12002-2_12 · Source: DBLP

4 authors:

Roberto Bruttomesso
Nozomi Networks

Edgar Pek
University of Illinois, Urbana-Champaign

Natasha Sharygina
Universita della Svizzera Italiana, Lugano, Switzerland

Aliaksei Tsitovich
Sonova AG

The OpenSMT Solver

Roberto Bruttomesso

Edgar Pek, Natasha Sharygina, Aliaksei Tsitovich

University of Lugano, Switzerland
(Università della Svizzera Italiana)

September 18, 2010

opensmt

Outline

opensmt

Introduction

- Satisfiability Modulo Theory (SMT) Solvers are key engines of several verification approaches

Introduction

- Satisfiability Modulo Theory (SMT) Solvers are key engines of several verification approaches

- Efficient solvers however are proprietary (Z3, YICES, BARCELOGIC, MATHSAT, . . .)

opensmt

Introduction

- Satisfiability Modulo Theory (SMT) Solvers are key engines of several verification approaches
- Efficient solvers however are proprietary (Z3, YICES, BARCELOGIC, MATHSAT, ...)
- OPENSMT is an effort of providing a simple, extensible, and efficient infrastructure for the development of customized decision procedures

opensmt

Introduction

- Satisfiability Modulo Theories combines the efficiency of SAT and theory-specific decision procedures

$$a \wedge ((x + y \leq 0) \vee \neg a) \wedge ((x = 1) \vee b)$$

opensmt

Introduction

- Satisfiability Modulo Theories combines the efficiency of SAT and theory-specific decision procedures

$$a \wedge ((x + y \leq 0) \vee \neg a) \wedge ((x = 1) \vee b)$$

opensmt

- Satisfiability Modulo Theories combines the efficiency of SAT and theory-specific decision procedures

$$a \wedge ((x + y \leq 0) \vee \neg a) \wedge ((x = 1) \vee b)$$

Introduction

- Satisfiability Modulo Theories combines the efficiency of SAT and theory-specific decision procedures

$$a \wedge (\underbrace{(x + y \leq 0)}_{c} \vee \neg a) \wedge (\underbrace{(x = 1)}_{d} \vee b)$$

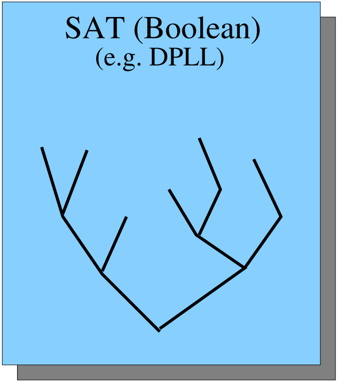
opensmt

Introduction

- We need to reason about Boolean combinations of atoms in a theory T (LRA for instance)

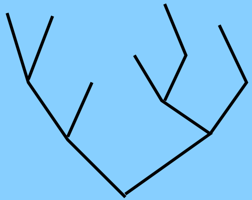
Introduction

- We need to reason about Boolean combinations of atoms in a theory T (LRA for instance)



- We need to reason about Boolean combinations of atoms in a theory T (LRA for instance)

SAT (Boolean)
(e.g. DPLL)



Linear Arithmetic
(e.g. Simplex)

$a_1x_1 + \ldots + a_nx_n + b \leq 0$

$\underline{\hspace{6cm}} \leq 0$

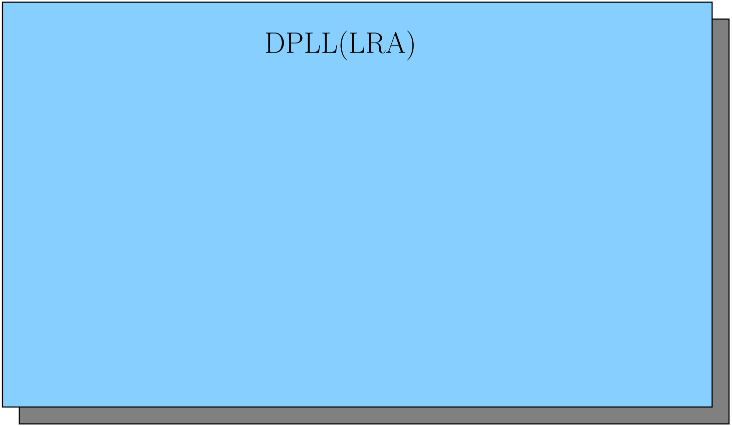
$\underline{\hspace{6cm}} \leq 0$

$\underline{\hspace{6cm}} \leq 0$

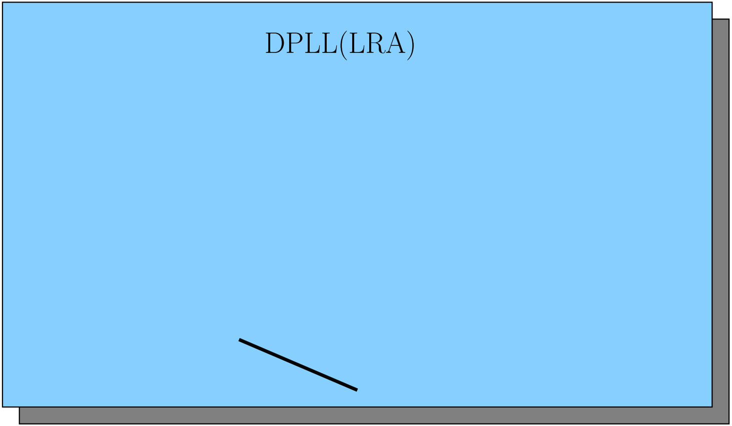
$\underline{\hspace{6cm}} \leq 0$

opensmt

- DPLL + LRA $\Rightarrow$ DPLL(LRA)

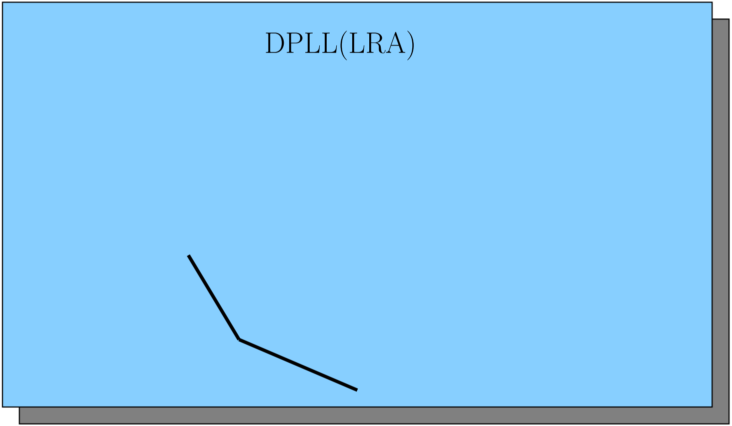


- DPLL + LRA ⇒ DPLL(LRA)

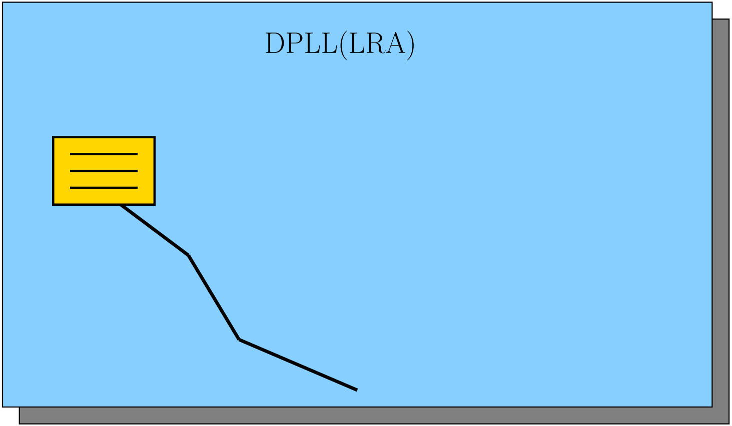


Introduction

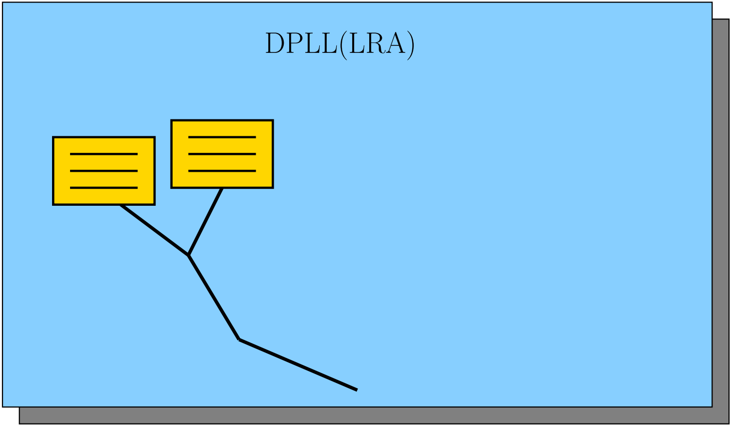
- DPLL $+$ LRA $\Rightarrow$ DPLL(LRA)



- DPLL + LRA $\Rightarrow$ DPLL(LRA)

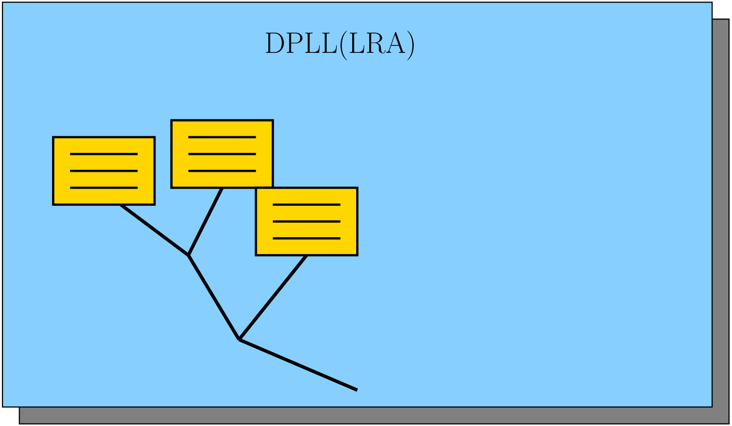


- DPLL + LRA $\Rightarrow$ DPLL(LRA)



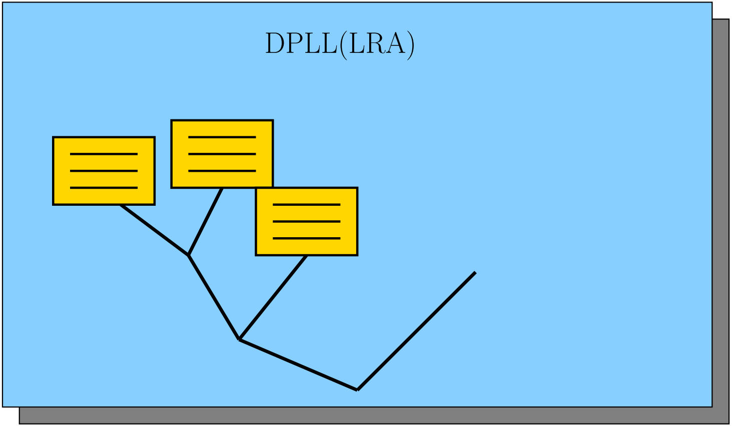
opensmt

- DPLL + LRA $\Rightarrow$ DPLL(LRA)



opensmt

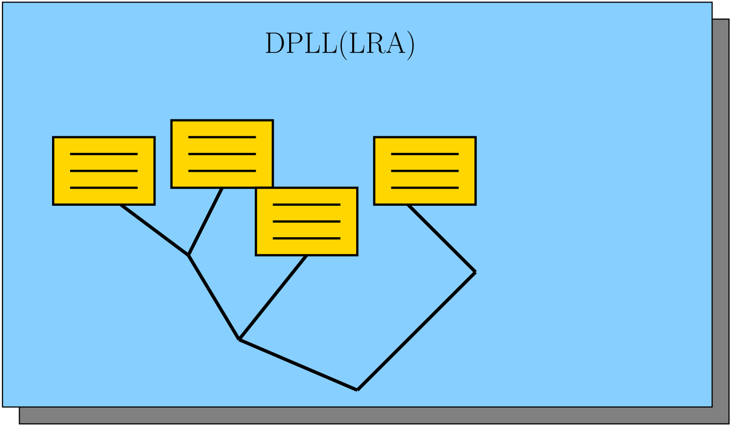
- DPLL + LRA $\Rightarrow$ DPLL(LRA)



opensmt

Introduction

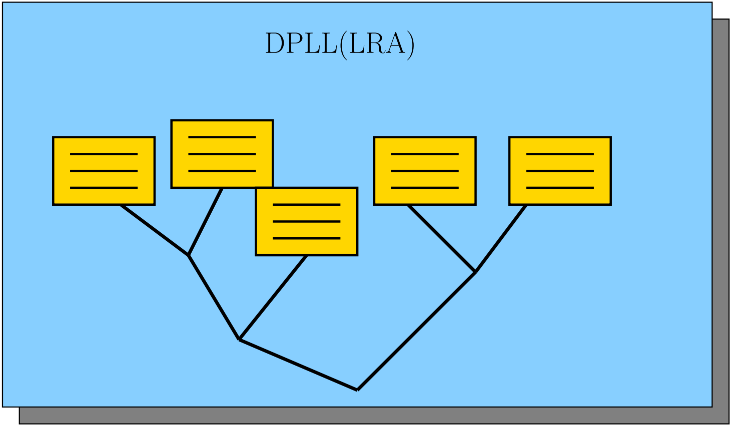
- DPLL $+$ LRA $\Rightarrow$ DPLL(LRA)



opensmt

Introduction

- DPLL + LRA $\Rightarrow$ DPLL(LRA)



Introduction

- DPLL(LRA) seems easy to achieve
  - Let DPLL enumerate Boolean models
  - Check LRA constraints with Simplex

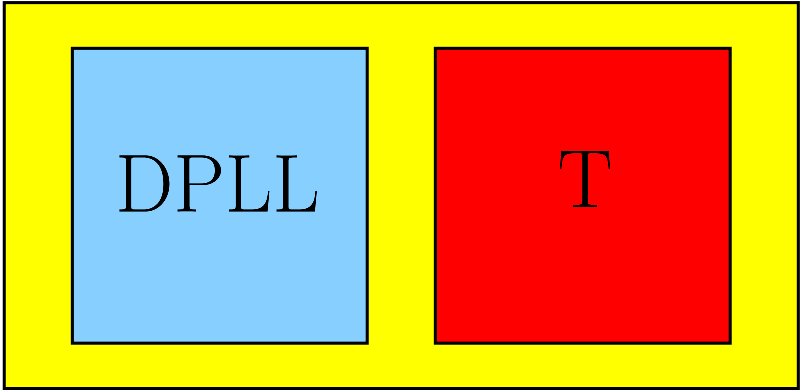
opensmt

Introduction

- DPLL(LRA) seems easy to achieve
  - Let DPLL enumerate Boolean models
  - Check LRA constraints with Simplex
- However a lot more has to be done to make it efficient
  - Don't wait for complete Boolean model
  - Theory Propagation
  - Preprocessing
  - Conversion to CNF
  - Theory Layering
  - ...

opensmt

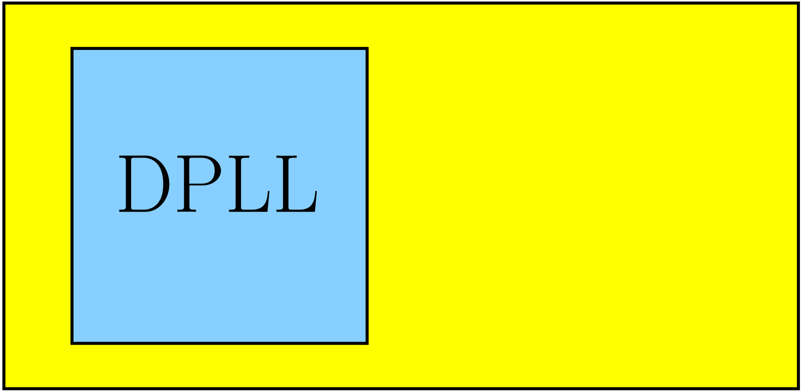
$$e(DPLL(T)) = \boxed{e(DPLL)} + \boxed{e(T)} + \boxed{e(COMM)}$$



opensmt

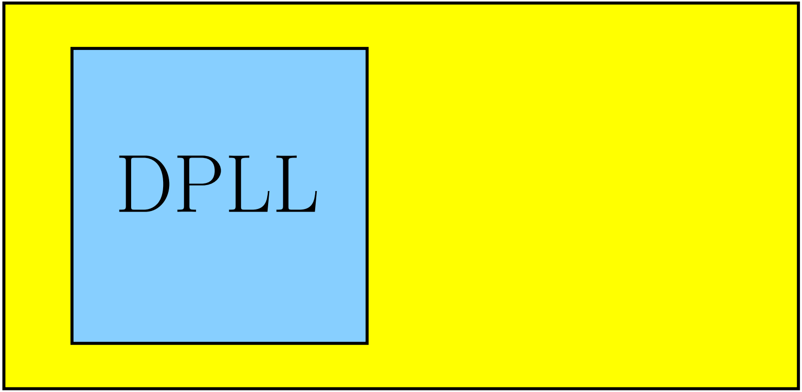
Introduction

$$\text{e(DPLL(T))} = \boxed{\text{e(DPLL)}} + \boxed{\text{e(T)}} + \boxed{\text{e(COMM)}}$$



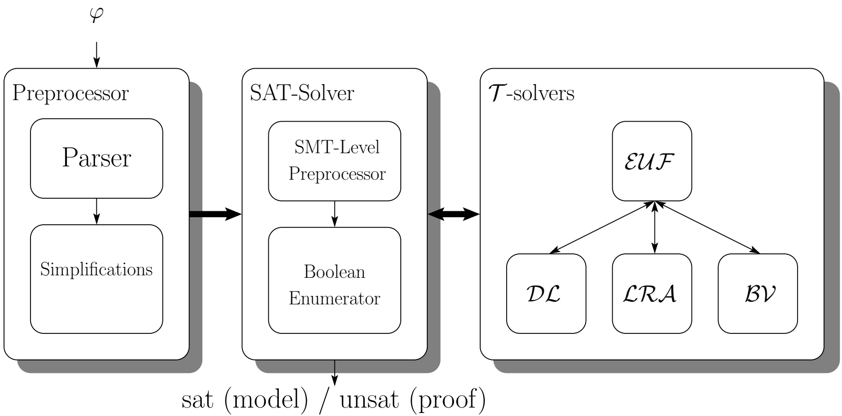
opensmt

$$e(DPLL(T)) \approx \boxed{e(T)}$$



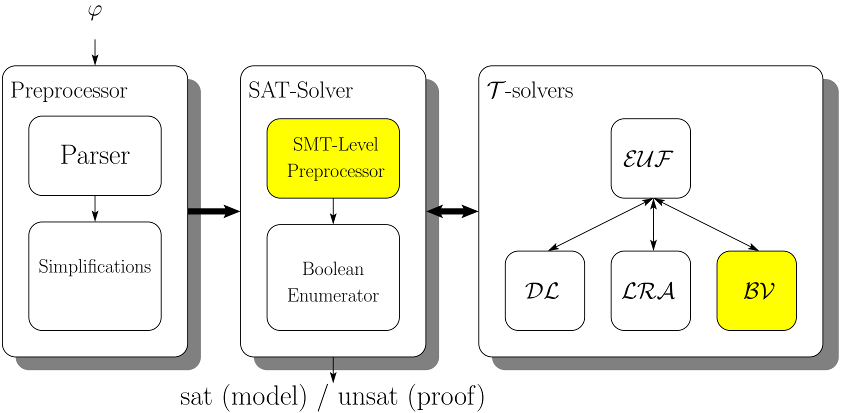
Outline

Architecture

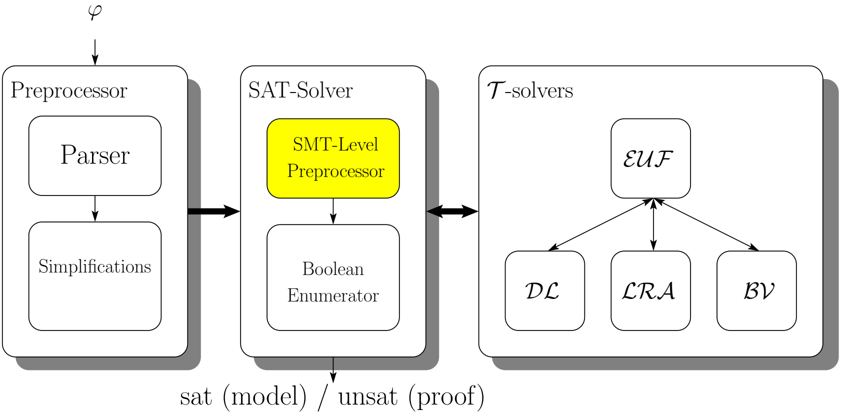


opensmt

Architecture



Architecture



opensmt

Outline

1. Introduction
2. Architecture
3. A Variable Elimination Techique for SMT
   - DP + FM = DPFM
   - A crazy benchmark suite
   - Related Work
4. Extending and Using OpenSMT
   - Extending OpenSMT
5. Conclusion

A Generic Template for Variable Elimination Procedures

Variable Types: T_1, T_2, ...
Resolution Rules: R_1, R_2, ...
Algorithm: Input: a set of constraints Repeat Choose a variable X of type T_i to eliminate Combine positive and negative occurrences of X, using R_i

opensmt

The Davis-Putnam Procedure [DP60]

Variable Types:
Resolution Rules:
Algorithm: Input: Repeat Choose a variable X of type to eliminate Combine positive and negative occurrences of X, using

opensmt

The Davis-Putnam Procedure [DP60]

Variable Types: **Bool**
Resolution Rules:
Algorithm: Input: Repeat Choose a variable X of type to eliminate Combine positive and negative occurrences of X, using

opensmt

The Davis-Putnam Procedure [DP60]

Variable Types: **Bool**
Resolution Rules: **Boolean Resolution (BR)**
Algorithm: Input: Repeat Choose a variable X of type to eliminate Combine positive and negative occurrences of X, using

opensmt

The Davis-Putnam Procedure [DP60]

Variable Types: **Bool**
Resolution Rules: **Boolean Resolution (BR)**
Algorithm: Input: **a set of Boolean clauses** Repeat Choose a variable X of type to eliminate Combine positive and negative occurrences of X, using

The Davis-Putnam Procedure [DP60]

Variable Types: **Bool**
Resolution Rules: **Boolean Resolution (BR)**
Algorithm: Input: **a set of Boolean clauses** Repeat Choose a variable X of type **Bool** to eliminate Combine positive and negative occurrences of X, using

opensmt

The Davis-Putnam Procedure [DP60]

Variable Types: **Bool**
Resolution Rules: **Boolean Resolution (BR)**
Algorithm: Input: **a set of Boolean clauses** Repeat Choose a variable X of type **Bool** to eliminate Combine positive and negative occurrences of X, using **BR**

opensmt

Boolean Resolution

- Clauses are expressions like $(a \vee \neg b \vee c)$, i.e., disjunctions of literals (Boolean variables or negated Boolean variables)

opensmt

Boolean Resolution

- Clauses are expressions like $(a \vee \neg b \vee c)$, i.e., disjunctions of literals (Boolean variables or negated Boolean variables)
- In the following C_1, C_2, D_1, D_2 are disjunctions of literals

Boolean Resolution

- Clauses are expressions like $(a \vee \neg b \vee c)$, i.e., disjunctions of literals (Boolean variables or negated Boolean variables)
- In the following C_1, C_2, D_1, D_2 are disjunctions of literals

Boolean Resolution for two clauses

$$(C_1 \vee \mathbf{a} \vee C_2) \otimes_a (D_1 \vee \neg \mathbf{a} \vee D_2) := (C_1 \vee C_2 \vee D_1 \vee D_2)$$

opensmt

Boolean Resolution

- Clauses are expressions like $(a \vee \neg b \vee c)$, i.e., disjunctions of literals (Boolean variables or negated Boolean variables)
- In the following C_1, C_2, D_1, D_2 are disjunctions of literals

Boolean Resolution for two clauses

$$(C_1 \vee \mathbf{a} \vee C_2) \otimes_a (D_1 \vee \neg \mathbf{a} \vee D_2) := (C_1 \vee C_2 \vee D_1 \vee D_2)$$

- Let S_a, $S_{\neg a}$ be the set of clauses with positive resp. negative occurrences of a

opensmt

Boolean Resolution

- Clauses are expressions like $(a \vee \neg b \vee c)$, i.e., disjunctions of literals (Boolean variables or negated Boolean variables)
- In the following C_1, C_2, D_1, D_2 are disjunctions of literals

Boolean Resolution for two clauses

$$(C_1 \vee \mathbf{a} \vee C_2) \otimes_a (D_1 \vee \neg\mathbf{a} \vee D_2) := (C_1 \vee C_2 \vee D_1 \vee D_2)$$

- Let S_a, $S_{\neg a}$ be the set of clauses with positive resp. negative occurrences of a

Boolean Resolution for sets of clauses

$$S_a \otimes_a S_{\neg a} := \{ C_1 \otimes_a C_2 \mid C_1 \in S_a, C_2 \in S_{\neg a} \}$$

opensmt

Boolean Resolution

- Let S_a, $S_{\neg a}$ be the set of clauses with positive resp. negative occurrences of a

Boolean Resolution for sets of clauses

$$S_a \otimes_a S_{\neg a} := \{ C_1 \otimes_a C_2 \mid C_1 \in S_a, C_2 \in S_{\neg a} \}$$

Theorem [DP60]

$S_a \cup S_{\neg a}$ is equisatisfiable with $S_a \otimes_a S_{\neg a}$

opensmt

DP - Example (on var a)

	OLD	NEW
	$(a \vee b \vee c)$	
	$(a \vee \neg b \vee \neg c)$	
	$(\neg a \vee \neg b \vee \neg c)$	
	$(\neg a \vee \neg b \vee c)$	

DP - Example (on var a)

	OLD	NEW
S_a	$(a \vee b \vee c)$	
	$(a \vee \neg b \vee \neg c)$	
$S_{\neg a}$	$(\neg a \vee \neg b \vee \neg c)$	
	$(\neg a \vee \neg b \vee c)$	

opensmt

DP - Example (on var a)

	OLD	NEW
S_a	$(a \vee b \vee c)$ $(a \vee \neg b \vee \neg c)$	
$S_{\neg a}$	$(\neg a \vee \neg b \vee \neg c)$ $(\neg a \vee \neg b \vee c)$	

opensmt

DP - Example (on var a)

	OLD	NEW
S_a	$(a \vee b \vee c)$ $(a \vee \neg b \vee \neg c)$	$(b \vee c \vee \neg b \vee \neg c)$
$S_{\neg a}$	$(\neg a \vee \neg b \vee \neg c)$ $(\neg a \vee \neg b \vee c)$	

DP - Example (on var a)

	OLD	NEW
S_a	$(a \vee b \vee c)$ $(a \vee \neg b \vee \neg c)$	$(b \vee c \vee \neg b \vee \neg c)$
$S_{\neg a}$	$(\neg a \vee \neg b \vee \neg c)$ $(\neg a \vee \neg b \vee c)$	

opensmt

DP - Example (on var a)

	OLD	NEW
S_a	$(a \vee b \vee c)$	$(b \vee c \vee \neg b \vee \neg c)$
	$(a \vee \neg b \vee \neg c)$	$(b \vee c \vee \neg b \vee c)$
$S_{\neg a}$	$(\neg a \vee \neg b \vee \neg c)$	
	$(\neg a \vee \neg b \vee c)$	

opensmt

DP - Example (on var a)

	OLD	NEW
S_a	$(a \vee b \vee c)$ $(a \vee \neg b \vee \neg c)$	$(b \vee c \vee \neg b \vee \neg c)$ $(b \vee c \vee \neg b \vee c)$
$S_{\neg a}$	$(\neg a \vee \neg b \vee \neg c)$ $(\neg a \vee \neg b \vee c)$	

opensmt

DP - Example (on var a)

	OLD	NEW
S_a	$(a \vee b \vee c)$ $(a \vee \neg b \vee \neg c)$	$(b \vee c \vee \neg b \vee \neg c)$ $(b \vee c \vee \neg b \vee c)$
$S_{\neg a}$	$(\neg a \vee \neg b \vee \neg c)$ $(\neg a \vee \neg b \vee c)$	$(\neg b \vee \neg c)$

opensmt

DP - Example (on var a)

	OLD	NEW
S_a	$(a \vee b \vee c)$ $(a \vee \neg b \vee \neg c)$	$(b \vee c \vee \neg b \vee \neg c)$ $(b \vee c \vee \neg b \vee c)$ $(\neg b \vee \neg c)$
$S_{\neg a}$	$(\neg a \vee \neg b \vee \neg c)$ $(\neg a \vee \neg b \vee c)$	

opensmt

DP - Example (on var a)

	OLD	NEW
S_a	$(a \vee b \vee c)$	$(b \vee c \vee \neg b \vee \neg c)$
	$(a \vee \neg b \vee \neg c)$	$(b \vee c \vee \neg b \vee c)$
$S_{\neg a}$	$(\neg a \vee \neg b \vee \neg c)$	$(\neg b \vee \neg c)$
	$(\neg a \vee \neg b \vee c)$	$(\neg b \vee \neg c \vee c)$

opensmt

DP - Example (on var a)

OLD	NEW
$(a \vee b \vee c)$	$(b \vee c \vee \neg b \vee \neg c)$
$(a \vee \neg b \vee \neg c)$	$(b \vee c \vee \neg b \vee c)$
$(\neg a \vee \neg b \vee \neg c)$	$(\neg b \vee \neg c)$
$(\neg a \vee \neg b \vee c)$	$(\neg b \vee \neg c \vee c)$

DP - Example (on var a)

OLD	NEW
$(a \vee b \vee c)$	$\cancel{(b \vee c \vee \neg b \vee \neg c)}$
$(a \vee \neg b \vee \neg c)$	$\cancel{(b \vee c \vee \neg b \vee c)}$
$(\neg a \vee \neg b \vee \neg c)$	$(\neg b \vee \neg c)$
$(\neg a \vee \neg b \vee c)$	$\cancel{(\neg b \vee \neg c \vee c)}$

opensmt

DP - Example (on var a)

OLD	NEW
$(a \vee b \vee c)$	
$(a \vee \neg b \vee \neg c)$	
$(\neg a \vee \neg b \vee \neg c)$	$(\neg b \vee \neg c)$
$(\neg a \vee \neg b \vee c)$	

opensmt

The Fourier-Motzkin Elimination [Fou26]

Variable Types:
Resolution Rules:
Algorithm:
Input:
Repeat
Choose a variable X of type to eliminate
Combine positive and negative occurrences of X, using

opensmt

The Fourier-Motzkin Elimination [Fou26]

Variable Types: **Rational**
Resolution Rules:
Algorithm: Input: Repeat Choose a variable X of type to eliminate Combine positive and negative occurrences of X, using

opensmt

The Fourier-Motzkin Elimination [Fou26]

Variable Types: **Rational**	
Resolution Rules: $\mathcal{LRA}$ **Resolution (RR)**	
Algorithm:	

Input:

Repeat

Choose a variable X of type to eliminate

Combine positive and negative occurrences of X, using

opensmt

The Fourier-Motzkin Elimination [Fou26]

Variable Types: **Rational**
Resolution Rules: $\mathcal{LRA}$ **Resolution (RR)**
Algorithm: Input: **a set of $\mathcal{LRA}$ constraints** Repeat Choose a variable X of type to eliminate Combine positive and negative occurrences of X, using

opensmt

The Fourier-Motzkin Elimination [Fou26]

Variable Types: **Rational**
Resolution Rules: $\mathcal{LRA}$ **Resolution (RR)**
Algorithm: Input: **a set of** $\mathcal{LRA}$ **constraints** Repeat Choose a variable X of type **Rational** to eliminate Combine positive and negative occurrences of X, using

The Fourier-Motzkin Elimination [Fou26]

Variable Types: **Rational**
Resolution Rules: $\mathcal{LRA}$ **Resolution (RR)**
Algorithm: Input: **a set of** $\mathcal{LRA}$ **constraints** Repeat Choose a variable X of type **Rational** to eliminate Combine positive and negative occurrences of X, using **RR**

$\mathcal{LRA}$ Resolution

- $\mathcal{LRA}$ constraints are expressions like $3x - 5y + 10z \leq 15$

opensmt

$\mathcal{LRA}$ Resolution

- $\mathcal{LRA}$ constraints are expressions like $3x - 5y + 10z \leq 15$
- Notice that $\leq$ is sufficient to represent also $\{=, <\}$ (see [DdM06])

opensmt

$\mathcal{LRA}$ Resolution

- $\mathcal{LRA}$ constraints are expressions like $3x - 5y + 10z \leq 15$
- Notice that $\leq$ is sufficient to represent also $\{=, <\}$ (see [DdM06])
- $\mathcal{LRA}$ constraints can be rewritten in terms of upper bound $x \leq p$ or lower bound $-x \leq p$ for a variable x.

opensmt

$\mathcal{LRA}$ Resolution

- $\mathcal{LRA}$ constraints are expressions like $3x - 5y + 10z \leq 15$
- Notice that $\leq$ is sufficient to represent also $\{=, <\}$ (see [DdM06])
- $\mathcal{LRA}$ constraints can be rewritten in terms of upper bound $x \leq p$ or lower bound $-x \leq p$ for a variable x.

opensmt

$\mathcal{LRA}$ Resolution

- $\mathcal{LRA}$ constraints are expressions like $3x - 5y + 10z \leq 15$
- Notice that $\leq$ is sufficient to represent also $\{=, <\}$ (see [DdM06])
- $\mathcal{LRA}$ constraints can be rewritten in terms of upper bound $x \leq p$ or lower bound $-x \leq p$ for a variable x.

$\mathcal{LRA}$ Resolution for two constraints

$$(x \leq p) \otimes_x (-x \leq q) := (-q \leq p)$$

opensmt

$\mathcal{LRA}$ Resolution

- $\mathcal{LRA}$ constraints are expressions like $3x - 5y + 10z \leq 15$
- Notice that $\leq$ is sufficient to represent also $\{=, <\}$ (see [DdM06])
- $\mathcal{LRA}$ constraints can be rewritten in terms of upper bound $x \leq p$ or lower bound $-x \leq p$ for a variable x.

$\mathcal{LRA}$ Resolution for two constraints

$$(x \leq p) \otimes_x (-x \leq q) := (-q \leq p)$$

- Let S_x, S_{-x} be the set of upper resp. lower bounds for x

opensmt

$\mathcal{LRA}$ Resolution

- $\mathcal{LRA}$ constraints are expressions like $3x - 5y + 10z \leq 15$
- Notice that $\leq$ is sufficient to represent also $\{=, <\}$ (see [DdM06])
- $\mathcal{LRA}$ constraints can be rewritten in terms of upper bound $x \leq p$ or lower bound $-x \leq p$ for a variable x.

$\mathcal{LRA}$ Resolution for two constraints

$$(x \leq p) \otimes_x (-x \leq q) := (-q \leq p)$$

- Let S_x, S_{-x} be the set of upper resp. lower bounds for x

$\mathcal{LRA}$ Resolution for sets of constraints

$$S_x \otimes_x S_{-x} := \{(x \leq p) \otimes_x (-x \leq q) \mid (x \leq p) \in S_x, (-x \leq q) \in S_{-x}\}$$

opensmt

$\mathcal{LRA}$ Resolution

- Let S_x, S_{-x} be the set of upper resp. lower bounds for x

$\mathcal{LRA}$ Resolution for sets of constraints

$$S_x \otimes_x S_{-x} := \{(x \leq p) \otimes_x (-x \leq q) \mid (x \leq p) \in S_x, (-x \leq q) \in S_{-x}\ \}$$

Theorem [Fou26]

$S_x \cup S_{-x}$ is equisatisfiable with $S_x \otimes_x S_{-x}$

opensmt

FM - Example (on var z)

	OLD	NEW
	$-x + z \leq -4$	
	$x + z \leq 18$	
	$x - z \leq 6$	
	$-x - z \leq -16$	
	$y \leq 5$	
	$-y \leq -3$	

opensmt

FM - Example (on var z)

	OLD	NEW
S_z	$-x + z \leq -4$ $x + z \leq 18$	
S_{-z}	$x - z \leq 6$ $-x - z \leq -16$	
	$y \leq 5$ $-y \leq -3$	$y \leq 5$ $-y \leq -3$

opensmt

FM - Example (on var z)

	OLD	NEW
S_z	$-x + z \leq -4$ $x + z \leq 18$	
S_{-z}	$x - z \leq 6$ $-x - z \leq -16$	
	$y \leq 5$ $-y \leq -3$	$y \leq 5$ $-y \leq -3$

opensmt

FM - Example (on var z)

	OLD	NEW
S_z	$-x + z \leq -4$	$0 \leq 2$
	$x + z \leq 18$	
S_{-z}	$x - z \leq 6$	
	$-x - z \leq -16$	
	$y \leq 5$	$y \leq 5$
	$-y \leq -3$	$-y \leq -3$

opensmt

FM - Example (on var z)

	OLD	NEW
S_z	$-x + z \leq -4$	$0 \leq 2$
	$x + z \leq 18$	
S_{-z}	$x - z \leq 6$	
	$-x - z \leq -16$	
	$y \leq 5$	$y \leq 5$
	$-y \leq -3$	$-y \leq -3$

opensmt

FM - Example (on var z)

	OLD	NEW
S_z	$-x + z \leq -4$ $x + z \leq 18$	$0 \leq 2$ $-x \leq -10$
S_{-z}	$x - z \leq 6$ $-x - z \leq -16$	
	$y \leq 5$ $-y \leq -3$	$y \leq 5$ $-y \leq -3$

opensmt

FM - Example (on var z)

	OLD	NEW
S_z	$-x + z \leq -4$ $x + z \leq 18$	$0 \leq 2$ $-x \leq -10$
S_{-z}	$x - z \leq 6$ $-x - z \leq -16$	
	$y \leq 5$ $-y \leq -3$	$y \leq 5$ $-y \leq -3$

opensmt

FM - Example (on var z)

	OLD	NEW
S_z	$-x + z \leq -4$	$0 \leq 2$
	$x + z \leq 18$	$-x \leq -10$
S_{-z}	$x - z \leq 6$	$x \leq 12$
	$-x - z \leq -16$	
	$y \leq 5$	$y \leq 5$
	$-y \leq -3$	$-y \leq -3$

opensmt

FM - Example (on var z)

	OLD	NEW
S_z	$-x + z \leq -4$	$0 \leq 2$
	$x + z \leq 18$	$-x \leq -10$
S_{-z}	$x - z \leq 6$	$x \leq 12$
	$-x - z \leq -16$	
	$y \leq 5$	$y \leq 5$
	$-y \leq -3$	$-y \leq -3$

opensmt

FM - Example (on var z)

	OLD	NEW
S_z	$-x + z \leq -4$	$0 \leq 2$
	$x + z \leq 18$	$-x \leq -10$
S_{-z}	$x - z \leq 6$	$x \leq 12$
	$-x - z \leq -16$	$0 \leq 2$
	$y \leq 5$	$y \leq 5$
	$-y \leq -3$	$-y \leq -3$

opensmt

FM - Example (on var z)

	OLD	NEW
S_z	$-x + z \leq -4$	$\cancel{0 \leq 2}$
	$x + z \leq 18$	$-x \leq -10$
S_{-z}	$x - z \leq 6$	$x \leq 12$
	$-x - z \leq -16$	$\cancel{0 \leq 2}$
	$y \leq 5$	$y \leq 5$
	$-y \leq -3$	$-y \leq -3$

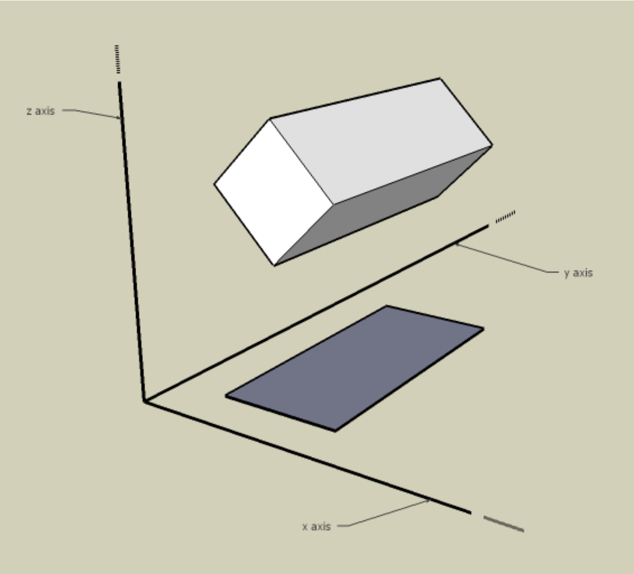
opensmt

FM - Example (on var z)

OLD	NEW
$-x + z \leq -4$	
$x + z \leq 18$	$-x \leq -10$
$x - z \leq 6$	$x \leq 12$
$-x - z \leq -16$	
$y \leq 5$	$y \leq 5$
$-y \leq -3$	$-y \leq -3$

opensmt

FM - Example (on var z)



DP + FM = DPFM

Variable Types:
Resolution Rules:
Algorithm: Input: Repeat Choose a variable X of type to eliminate Combine positive and negative occurrences of X, using

opensmt

DP + FM = DPFM

Variable Types: **Bool**, **Rational**
Resolution Rules:
Algorithm: Input: Repeat Choose a variable X of type to eliminate Combine positive and negative occurrences of X, using

opensmt

DP + FM = DPFM

Variable Types: **Bool**, **Rational**
Resolution Rules: **BR**, **SMT($\mathcal{LRA}$) Resolution (SR)**
Algorithm: Input: Repeat Choose a variable X of type to eliminate Combine positive and negative occurrences of X, using

opensmt

DP + FM = DPFM

Variable Types: **Bool**, **Rational**
Resolution Rules: **BR**, **SMT($\mathcal{LRA}$) Resolution (SR)**
Algorithm: Input: **a set of SMT($\mathcal{LRA}$) clauses** in OCCF Repeat Choose a variable X of type to eliminate Combine positive and negative occurrences of X, using

DP + FM = DPFM

Variable Types: **Bool**, **Rational**
Resolution Rules: **BR**, **SMT($\mathcal{LRA}$) Resolution (SR)**
Algorithm: Input: **a set of SMT($\mathcal{LRA}$) clauses** in OCCF Repeat Choose a variable X of type **Bool** (**Rational**) to eliminate Combine positive and negative occurrences of X, using

opensmt

DP + FM = DPFM

Variable Types: **Bool**, **Rational**
Resolution Rules: **BR**, **SMT($\mathcal{LRA}$) Resolution (SR)**
Algorithm: Input: **a set of SMT($\mathcal{LRA}$) clauses** in OCCF Repeat Choose a variable X of type **Bool** (**Rational**) to eliminate Combine positive and negative occurrences of X, using **BR** (**SR**)

One Constraint per Clause Form (OCCF)

- DPFM requires clauses in OCCF, clauses that contain **at most one $\mathcal{LRA}$ constraint**

One Constraint per Clause Form (OCCF)

- DPFM requires clauses in OCCF, clauses that contain **at most one $\mathcal{LRA}$ constraint**
- it is easy to transform clauses in OCCF, by means of auxiliary Boolean variables

opensmt

One Constraint per Clause Form (OCCF)

- DPFM requires clauses in OCCF, clauses that contain **at most one $\mathcal{LRA}$ constraint**

- it is easy to transform clauses in OCCF, by means of auxiliary Boolean variables

  - E.g. $(a \vee (x \leq 3) \vee b \vee (x + y \leq 10))$ can be rewritten as $(a \vee (x \leq 3) \vee b \vee c)$ and $(\neg c \vee (x + y \leq 10))$

opensmt

SMT($\mathcal{LRA}$) Resolution

- negated $\mathcal{LRA}$ constr. can be expressed in terms of $\leq$
  - e.g. $\neg(x \leq 10)$ is equiv. to $-x \leq -10 - \delta, \quad (\delta > 0)$ (see [DdM06])

opensmt

SMT($\mathcal{LRA}$) Resolution

- negated $\mathcal{LRA}$ constr. can be expressed in terms of $\leq$
  - e.g. $\neg(x \leq 10)$ is equiv. to $-x \leq -10 - \delta, \quad (\delta > 0)$ (see [DdM06])
- $\mathcal{LRA}$ constraints in SMT($\mathcal{LRA}$) clauses can be rewritten in terms of upper bound $x \leq p$ or lower bound $-x \leq p$ for a variable x.

opensmt

SMT($\mathcal{LRA}$) Resolution

- negated $\mathcal{LRA}$ constr. can be expressed in terms of $\leq$
  - e.g. $\neg(x \leq 10)$ is equiv. to $-x \leq -10 - \delta, \quad (\delta > 0)$ (see [DdM06])
- $\mathcal{LRA}$ constraints in SMT($\mathcal{LRA}$) clauses can be rewritten in terms of upper bound $x \leq p$ or lower bound $-x \leq p$ for a variable x.

SMT($\mathcal{LRA}$) Resolution for two clauses in OCCF

$$(C_1 \vee (x \leq p) \vee C_2) \otimes_x (D_1 \vee (-x \leq q) \vee D_2) := C_1 \vee C_2 \vee (-q \leq p) \vee D_1 \vee D_2$$

opensmt

SMT($\mathcal{LRA}$) Resolution

- negated $\mathcal{LRA}$ constr. can be expressed in terms of $\leq$
  - e.g. $\neg(x \leq 10)$ is equiv. to $-x \leq -10 - \delta, \quad (\delta > 0)$ (see [DdM06])
- $\mathcal{LRA}$ constraints in SMT($\mathcal{LRA}$) clauses can be rewritten in terms of upper bound $x \leq p$ or lower bound $-x \leq p$ for a variable x.

SMT($\mathcal{LRA}$) Resolution for two clauses in OCCF

$$(C_1 \vee (x \leq p) \vee C_2) \otimes_x (D_1 \vee (-x \leq q) \vee D_2) := C_1 \vee C_2 \vee (-q \leq p) \vee D_1 \vee D_2$$

- Let S_x, S_{-x} be the set of SMT($\mathcal{LRA}$) clauses where x appears upper bounded resp. lower bounded

opensmt

SMT($\mathcal{LRA}$) Resolution

- negated $\mathcal{LRA}$ constr. can be expressed in terms of $\leq$
  - e.g. $\neg(x \leq 10)$ is equiv. to $-x \leq -10 - \delta, \quad (\delta > 0)$ (see [DdM06])
- $\mathcal{LRA}$ constraints in SMT($\mathcal{LRA}$) clauses can be rewritten in terms of upper bound $x \leq p$ or lower bound $-x \leq p$ for a variable x.

SMT($\mathcal{LRA}$) Resolution for two clauses in OCCF

$$(C_1 \vee (x \leq p) \vee C_2) \otimes_x (D_1 \vee (-x \leq q) \vee D_2) := C_1 \vee C_2 \vee (-q \leq p) \vee D_1 \vee D_2$$

- Let S_x, S_{-x} be the set of SMT($\mathcal{LRA}$) clauses where x appears upper bounded resp. lower bounded

SMT($\mathcal{LRA}$) Resolution for sets of clauses in OCCF

$$S_x \otimes_x S_{-x} := \{ C_1 \otimes_x C_2 \mid C_1 \in S_x, C_2 \in S_{-x} \}$$

opensmt

SMT($\mathcal{LRA}$) Resolution

- Let S_x, S_{-x} be the set of SMT($\mathcal{LRA}$) clauses where x appears upper bounded resp. lower bounded

SMT($\mathcal{LRA}$) Resolution for sets of clauses in OCCF

$$S_x \otimes_x S_{-x} := \{ C_1 \otimes_x C_2 \mid C_1 \in S_x, C_2 \in S_{-x} \}$$

Theorem

$S_x \cup S_{-x}$ is equisatisfiable with $S_x \otimes_x S_{-x}$

opensmt

DPFM - Example (on var z)

$\neg a_1 \vee (-z \leq -3)$ $\quad$ $a_1 \vee (z \leq 3 - \delta) \vee a_2$
$\neg a_1 \vee (-x \leq -3)$ $\quad$ $\neg a_2 \vee (x \leq 3 - \delta) \vee a_3$
$\neg a_1 \vee (-y \leq -3)$ $\quad$ $\neg a_3 \vee (y \leq 3 - \delta) \vee a_4$
$\neg a_1 \vee (y \leq 5)$ $\quad$ $\neg a_4 \vee (-y \leq 5 - \delta) \vee a_5$
$\neg a_1 \vee (x \leq 5)$ $\quad$ $\neg a_5 \vee (-x \leq 5 - \delta) \vee a_6$
$\neg a_1 \vee (z \leq 5)$ $\quad$ $\neg a_6 \vee (-z \leq 5 - \delta)$
$\neg b_1 \vee (-z \leq -2)$ $\quad$ $b_1 \vee (z \leq 2 - \delta) \vee b_2$
$\neg b_1 \vee (-x \leq -2)$ $\quad$ $\neg b_2 \vee (x \leq 2 - \delta) \vee b_3$
$\neg b_1 \vee (-y \leq -2)$ $\quad$ $\neg b_3 \vee (y \leq 2 - \delta) \vee b_4$
$\neg b_1 \vee (y \leq 4)$ $\quad$ $\neg b_4 \vee (-y \leq 4 - \delta) \vee b_5$
$\neg b_1 \vee (x \leq 4)$ $\quad$ $\neg b_5 \vee (-x \leq 4 - \delta) \vee b_6$
$\neg b_1 \vee (z \leq 4)$ $\quad$ $\neg b_6 \vee (-z \leq 4 - \delta)$
$a_1 \vee b_1$

opensmt

DPFM - Example (on var z)

$\neg a_1 \vee (-z \leq -3)$	$a_1 \vee (z \leq 3 - \delta) \vee a_2$
$\neg a_1 \vee (-x \leq -3)$	$\neg a_2 \vee (x \leq 3 - \delta) \vee a_3$
$\neg a_1 \vee (-y \leq -3)$	$\neg a_3 \vee (y \leq 3 - \delta) \vee a_4$
$\neg a_1 \vee (y \leq 5)$	$\neg a_4 \vee (-y \leq -5 - \delta) \vee a_5$
$\neg a_1 \vee (x \leq 5)$	$\neg a_5 \vee (-x \leq -5 - \delta) \vee a_6$
$\neg a_1 \vee (z \leq 5)$	$\neg a_6 \vee (-z \leq -5 - \delta)$
$\neg b_1 \vee (-z \leq -2)$	$b_1 \vee (z \leq 2 - \delta) \vee b_2$
$\neg b_1 \vee (-x \leq -2)$	$\neg b_2 \vee (x \leq 2 - \delta) \vee b_3$
$\neg b_1 \vee (-y \leq -2)$	$\neg b_3 \vee (y \leq 2 - \delta) \vee b_4$
$\neg b_1 \vee (y \leq 4)$	$\neg b_4 \vee (-y \leq -4 - \delta) \vee b_5$
$\neg b_1 \vee (x \leq 4)$	$\neg b_5 \vee (-x \leq -4 - \delta) \vee b_6$
$\neg b_1 \vee (z \leq 4)$	$\neg b_6 \vee (-z \leq -4 - \delta)$
$a_1 \vee b_1$	

DPFM - Example (on var z)

	OLD	NEW
	$\neg a_1 \vee (z \leq 5)$	
	$\neg b_1 \vee (z \leq 4)$	
	$a_1 \vee (z \leq 3 - \delta) \vee a_2$	
	$b_1 \vee (z \leq 2 - \delta) \vee b_2$	
	$\neg a_6 \vee (-z \leq -5 - \delta)$	
	$\neg b_6 \vee (-z \leq -4 - \delta)$	
	$\neg a_1 \vee (-z \leq -3)$	
	$\neg b_1 \vee (-z \leq -2)$	

opensmt

DPFM - Example (on var z)

	OLD	NEW
S_z	$\neg a_1 \vee (z \leq 5)$ $\neg b_1 \vee (z \leq 4)$ $a_1 \vee (z \leq 3 - \delta) \vee a_2$ $b_1 \vee (z \leq 2 - \delta) \vee b_2$	
S_{-z}	$\neg a_6 \vee (-z \leq -5 - \delta)$ $\neg b_6 \vee (-z \leq -4 - \delta)$ $\neg a_1 \vee (-z \leq -3)$ $\neg b_1 \vee (-z \leq -2)$	

opensmt

DPFM - Example (on var z)

	OLD	NEW
S_z	$\neg a_1 \vee (z \leq 5)$ $\neg b_1 \vee (z \leq 4)$ $a_1 \vee (z \leq 3 - \delta) \vee a_2$ $b_1 \vee (z \leq 2 - \delta) \vee b_2$	
S_{-z}	$\neg a_6 \vee (-z \leq -5 - \delta)$ $\neg b_6 \vee (-z \leq -4 - \delta)$ $\neg a_1 \vee (-z \leq -3)$ $\neg b_1 \vee (-z \leq -2)$	

opensmt

DPFM - Example (on var z)

	OLD	NEW
S_z	$\neg a_1 \vee (z \leq 5)$ $\neg b_1 \vee (z \leq 4)$ $a_1 \vee (z \leq 3 - \delta) \vee a_2$ $b_1 \vee (z \leq 2 - \delta) \vee b_2$	$\neg a_1 \vee (0 \leq -\delta) \vee \neg a_6$
S_{-z}	$\neg a_6 \vee (-z \leq -5 - \delta)$ $\neg b_6 \vee (-z \leq -4 - \delta)$ $\neg a_1 \vee (-z \leq -3)$ $\neg b_1 \vee (-z \leq -2)$	

opensmt

DPFM - Example (on var z)

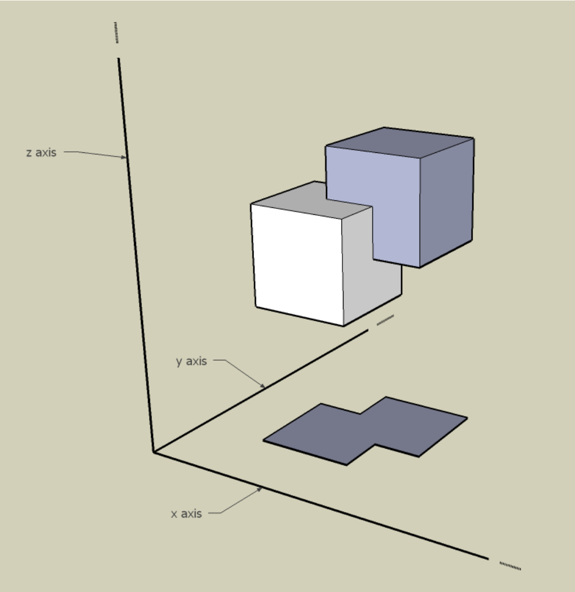
	OLD	NEW
S_z	$\neg a_1 \lor (z \leq 5)$	$\neg a_1 \lor (0 \leq -\delta) \lor \neg a_6$
	$\neg b_1 \lor (z \leq 4)$	$\neg a_1 \lor (0 \leq 1 - \delta) \lor \neg b_6$
	$a_1 \lor (z \leq 3 - \delta) \lor a_2$	$\neg a_1 \lor (0 \leq 2)$
	$b_1 \lor (z \leq 2 - \delta) \lor b_2$	$\neg a_1 \lor (0 \leq 3) \lor \neg b_1$
S_{-z}	$\neg a_6 \lor (-z \leq -5 - \delta)$	$\neg b_1 \lor (0 \leq -1 - \delta) \lor \neg a_6$
	$\neg b_6 \lor (-z \leq -4 - \delta)$	$\neg b_1 \lor (0 \leq -\delta) \lor \neg b_6$
	$\neg a_1 \lor (-z \leq -3)$	$\neg b_1 \lor (0 \leq 1) \lor \neg a_1$
	$\neg b_1 \lor (-z \leq -2)$	$\neg b_1 \lor (0 \leq 2)$
		$a_1 \lor (0 \leq -2 - \delta) \lor a_2 \lor \neg a_6$
		$a_1 \lor (0 \leq -1 - \delta) \lor a_2 \lor \neg b_6$
		$a_1 \lor (0 \leq -\delta) \lor \neg a_1 \lor a_2$
		$a_1 \lor (0 \leq 1 - \delta) \lor \neg b_1 \lor a_2$
		$b_1 \lor (0 \leq -3 - \delta) \lor b_2 \lor \neg a_6$
		$b_1 \lor (0 \leq -2 - \delta) \lor b_2 \lor \neg b_6$
		$b_1 \lor (0 \leq -1 - \delta) \lor \neg a_1 \lor b_2$
		$b_1 \lor (0 \leq -\delta) \lor \neg b_1 \lor b_2$

DPFM - Example (on var z)

OLD	NEW
$\neg a_1 \vee (z \leq 5)$	$\neg a_1 \vee \neg a_6$
$\neg b_1 \vee (z \leq 4)$	
$a_1 \vee (z \leq 3 - \delta) \vee a_2$	
$b_1 \vee (z \leq 2 - \delta) \vee b_2$	
$\neg a_6 \vee (-z \leq -5 - \delta)$	$\neg b_1 \vee \neg a_6$
$\neg b_6 \vee (-z \leq -4 - \delta)$	$\neg b_1 \vee \neg b_6$
$\neg a_1 \vee (-z \leq -3)$	
$\neg b_1 \vee (-z \leq -2)$	
	$a_1 \vee a_2 \vee \neg a_6$
	$a_1 \vee a_2 \vee \neg b_6$
	$b_1 \vee b_2 \vee \neg a_6$
	$b_1 \vee b_2 \vee \neg b_6$
	$b_1 \vee \neg a_1 \vee b_2$

opensmt

DPFM - Example (on var z)



Variable Elimination - Complexity

- High worst-case complexity

Variable Elimination - Complexity

- High worst-case complexity
- Suppose we have n variables and m initial clauses

Variable Elimination - Complexity

- High worst-case complexity
- Suppose we have n variables and m initial clauses
- Suppose we eliminate a_1: in the worst case $|S_{a_1}| = |S_{\neg a_1}| = \frac{m}{2}$

opensmt

Variable Elimination - Complexity

- High worst-case complexity
- Suppose we have n variables and m initial clauses
- Suppose we eliminate a_1: in the worst case $|S_{a_1}| = |S_{\neg a_1}| = \frac{m}{2}$
- After elim. a_1 we have $\frac{m}{2} \times \frac{m}{2} = \frac{m^2}{4}$ clauses

opensmt

Variable Elimination - Complexity

- High worst-case complexity
- Suppose we have n variables and m initial clauses
- Suppose we eliminate a_1: in the worst case $|S_{a_1}| = |S_{\neg a_1}| = \frac{m}{2}$
- After elim. a_1 we have $\frac{m}{2} \times \frac{m}{2} = \frac{m^2}{4}$ clauses
- After elim. a_2 we have $\frac{m^2}{8} \times \frac{m^2}{8} = \frac{m^4}{4^3}$ clauses

opensmt

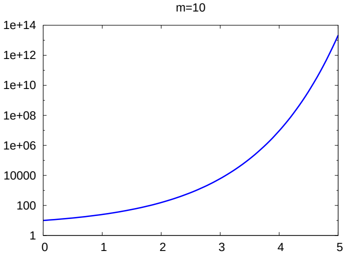
Variable Elimination - Complexity

- High worst-case complexity
- Suppose we have n variables and m initial clauses
- Suppose we eliminate a_1: in the worst case $|S_{a_1}| = |S_{\neg a_1}| = \frac{m}{2}$
- After elim. a_1 we have $\frac{m}{2} \times \frac{m}{2} = \frac{m^2}{4}$ clauses
- After elim. a_2 we have $\frac{m^2}{8} \times \frac{m^2}{8} = \frac{m^4}{4^3}$ clauses
- After elim. a_n we have $\frac{m^{2^{n-1}}}{4^{(2^{n-1}-\frac{1}{2})}} \times \frac{m^{2^{n-1}}}{4^{(2^{n-1}-\frac{1}{2})}} = \frac{m^{2^n}}{4^{2^n-1}}$ clauses

opensmt

Variable Elimination - Complexity

- High worst-case complexity
- Suppose we have n variables and m initial clauses
- Suppose we eliminate a_1: in the worst case $|S_{a_1}| = |S_{\neg a_1}| = \frac{m}{2}$
- After elim. a_1 we have $\frac{m}{2} \times \frac{m}{2} = \frac{m^2}{4}$ clauses
- After elim. a_2 we have $\frac{m^2}{8} \times \frac{m^2}{8} = \frac{m^4}{4^3}$ clauses
- After elim. a_n we have $\frac{m^{2^{n-1}}}{4^{(2^{n-1}-\frac{1}{2})}} \times \frac{m^{2^{n-1}}}{4^{(2^{n-1}-\frac{1}{2})}} = \frac{m^{2^n}}{4^{2^n-1}}$ clauses



opensmt

Formula Simplification and Preprocessing

- Variable Elimination technique for SMT($\mathcal{LRA}$)

opensmt

Formula Simplification and Preprocessing

- Variable Elimination technique for SMT($\mathcal{LRA}$)
- We apply it for preprocessing, similarly to the SATElite [EB05] preprocessor
  - Variable elimination rule (among other rules) is applied in a controlled manner so that it does not produce too many clauses

Formula Simplification and Preprocessing

- Variable Elimination technique for SMT($\mathcal{LRA}$)
- We apply it for preprocessing, similarly to the SATElite [EB05] preprocessor
  - Variable elimination rule (among other rules) is applied in a controlled manner so that it does not produce too many clauses
- In our case we use two upper bounds for **two parameters** concerning Rational variables

opensmt

Formula Simplification and Preprocessing

- Variable Elimination technique for SMT($\mathcal{LRA}$)
- We apply it for preprocessing, similarly to the SATElite [EB05] preprocessor
    - Variable elimination rule (among other rules) is applied in a controlled manner so that it does not produce too many clauses
- In our case we use two upper bounds for **two parameters** concerning Rational variables
- **Centrality (for x)**: number of distinct variables that appear in some constraint with x

opensmt

Formula Simplification and Preprocessing

- Variable Elimination technique for SMT($\mathcal{LRA}$)
- We apply it for preprocessing, similarly to the SATElite [EB05] preprocessor
  - Variable elimination rule (among other rules) is applied in a controlled manner so that it does not produce too many clauses
- In our case we use two upper bounds for **two parameters** concerning Rational variables
- **Centrality (for x)**: number of distinct variables that appear in some constraint with x
- **Trade-off (for x)**: amount of new clauses that we want to "trade" for eliminating x

opensmt

Formula Simplification - (Centrality 2, Trade-off 128)

OpenSMT on QF_IDL/qlock Benchmarks - Structural Data								
	P.Time (s)		Clauses		TAtoms		TVars	
Bench	WO	W	WO	W	WO	W	WO	W
Ind 37	1.08	6.57	41137	35299	6129	5285	829	185
Ind 38	1.16	6.62	42265	36244	6299	5423	851	188
Ind 39	1.19	7.02	43381	37150	6467	5562	873	189
Ind 40	1.17	7.05	44457	38114	6619	5702	895	203
Base 18	0.80	1.87	18630	16314	2867	2559	375	137
Base 19	0.82	2.31	19780	17269	3045	2702	397	150
Base 20	0.95	2.47	20914	18246	3215	2851	419	151
Base 21	0.94	2.54	22052	19193	3389	2995	441	155

Formula Simplification - (Centrality 2, Trade-off 128)

OpenSMT on QF_IDL/qlock Benchmarks - Structural Data								
	P.Time (s)		Clauses		TAtoms		TVars	
Bench	WO	W	WO	W	WO	W	WO	W
Ind 37	1.08	6.57	41137	35299	6129	5285	829	185
Ind 38	1.16	6.62	42265	36244	6299	5423	851	188
Ind 39	1.19	7.02	43381	37150	6467	5562	873	189
Ind 40	1.17	7.05	44457	38114	6619	5702	895	203
Base 18	0.80	1.87	18630	16314	2867	2559	375	137
Base 19	0.82	2.31	19780	17269	3045	2702	397	150
Base 20	0.95	2.47	20914	18246	3215	2851	419	151
Base 21	0.94	2.54	22052	19193	3389	2995	441	155

OpenSMT on QF_IDL/qlock Benchmarks - Solving Time					
Bench	Time WO (s)	Time W (s)	Bench	Time WO (s)	Time W (s)
Base 18	61.3	**59.0**	Ind 37	90.5	**18.0**
Base 19	146.1	**138.4**	Ind 38	105.7	**54.6**
Base 20	> 1800	**940.1**	Ind 39	64.4	**46.7**
Base 21	1367.9	**765.0**	Ind 40	98.3	**37.3**

opensmt

Mixed Boolean-Theory Static Learning

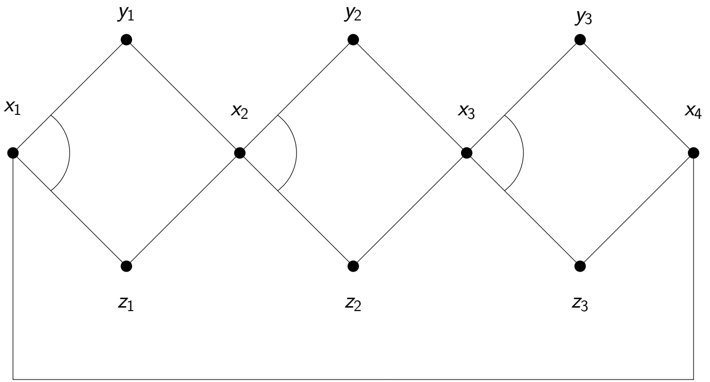
OpenSMT on QF_IDL/job_shop/jobshop12-2-6-6-2-4-9.smt							
Centr.	Trade-Off	VE	P.Time	Clauses	TAtoms	BAtoms	T.Time (s)
-	-	0	0.05	216	612	0	> 1800
12	64	0	0.05	216	612	0	> 1800
12	256	2	0.06	458	832	22	180.0
12	1024	4	0.04	1094	968	42	91.4
12	4096	6	0.09	3076	1032	60	67.2
12	16384	6	0.10	3076	1032	60	67.1
18	64	0	0.02	216	612	0	> 1800
18	256	4	0.02	714	1054	56	192.3
18	1024	8	0.07	2005	1566	109	105.6
18	4096	12	0.15	5702	2254	156	125.6
18	16384	12	0.16	5702	2254	156	125.9
24	64	0	0.02	216	612	0	> 1800
24	256	4	0.03	781	1108	66	193.2
24	1024	8	0.07	1978	1638	117	157.1
24	4096	11	0.19	5005	2198	153	89.4
24	16384	12	0.32	5519	2294	163	92.2

opensmt

A crazy benchmark suite

Fractal Diamonds

Our preprocessor is effective for those formulæ that are difficult to solve with the initial fixed set of theory atoms

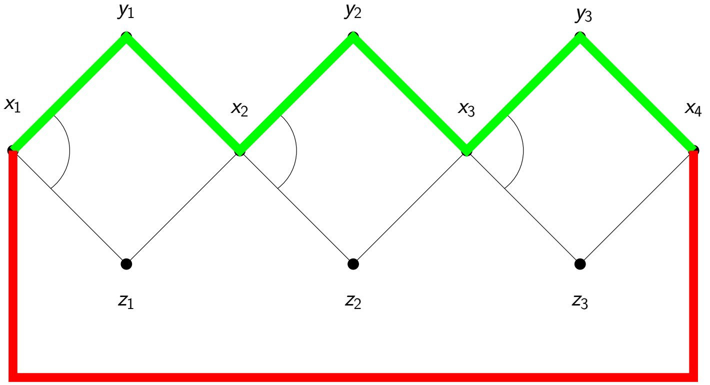


opensmt

A crazy benchmark suite

Fractal Diamonds

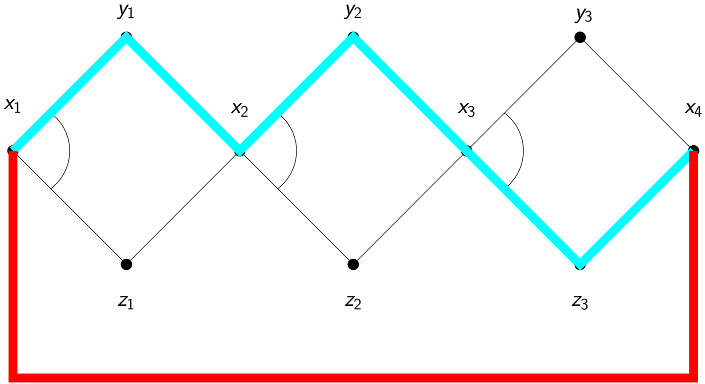
Our preprocessor is effective for those formulæ that are difficult to solve with the initial fixed set of theory atoms



opensmt

A crazy benchmark suite
Fractal Diamonds

Our preprocessor is effective for those formulæ that are difficult to solve with the initial fixed set of theory atoms

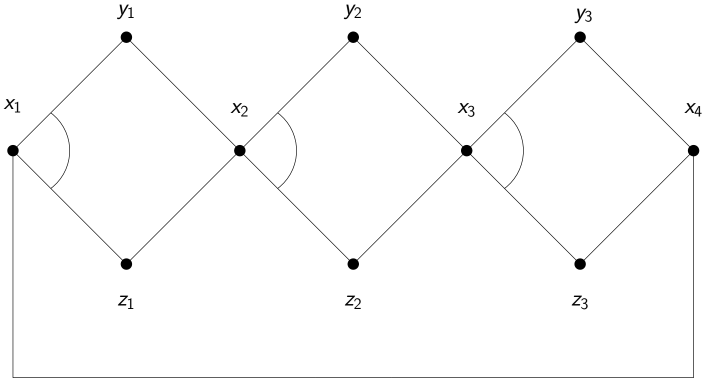


opensmt

A crazy benchmark suite

Fractal Diamonds

Our preprocessor is effective for those formulæ that are difficult to solve with the initial fixed set of theory atoms

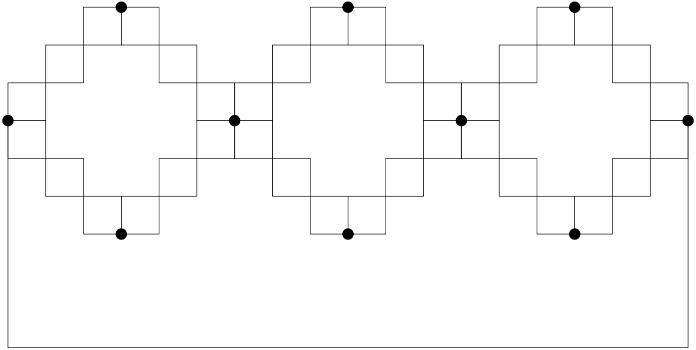


opensmt

A crazy benchmark suite

Fractal Diamonds

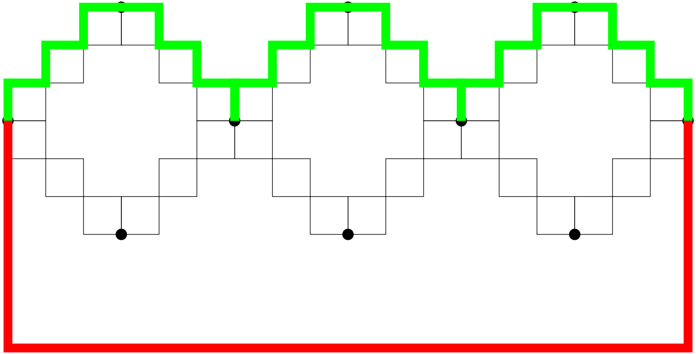
Our preprocessor is effective for those formulæ that are difficult to solve with the initial fixed set of theory atoms



opensmt

A crazy benchmark suite

Our preprocessor is effective for those formulæ that are difficult to solve with the initial fixed set of theory atoms



opensmt

A crazy benchmark suite
Fractal Diamonds (Centrality 18, Trade-off 8192)

B = BarceLogic (SMTCOMP'08 1st place for IDL)
Z = Z3 (SMTCOMP'08 2nd place for IDL)
O = OpenSMT (with DPFM based preprocessor)

Fractal Diamonds - Solving time (s) - TO = 1200 s															
	1			2			3			4			5		
Or.	B	Z	O	B	Z	O	B	Z	O	B	Z	O	B	Z	O

opensmt

B = BarceLogic (SMTCOMP'08 1st place for IDL)
Z = Z3 (SMTCOMP'08 2nd place for IDL)
O = OpenSMT (with DPFM based preprocessor)

Fractal Diamonds - Solving time (s) - TO = 1200 s															
	1			2			3			4			5		
Or.	B	Z	O	B	Z	O	B	Z	O	B	Z	O	B	Z	O
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A crazy benchmark suite

Fractal Diamonds (Centrality 18, Trade-off 8192)

B = BarceLogic (SMTCOMP'08 1st place for IDL)
Z = Z3 (SMTCOMP'08 2nd place for IDL)
O = OpenSMT (with DPFM based preprocessor)

Fractal Diamonds - Solving time (s) - TO = 1200 s															
	1			2			3			4			5		
Or.	B	Z	O	B	Z	O	B	Z	O	B	Z	O	B	Z	O
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	118	13	**1**	T	T	**3**	T	T	**7**

A crazy benchmark suite

Fractal Diamonds (Centrality 18, Trade-off 8192)

B = BarceLogic (SMTCOMP'08 1st place for IDL)
Z = Z3 (SMTCOMP'08 2nd place for IDL)
O = OpenSMT (with DPFM based preprocessor)

Fractal Diamonds - Solving time (s) - TO = 1200 s															
	1			2			3			4			5		
Or.	B	Z	O	B	Z	O	B	Z	O	B	Z	O	B	Z	O
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	118	13	**1**	T	T	**3**	T	T	**7**
3	0	0	0	**0**	T	2	T	T	**153**	M	T	T	T	T	T

opensmt

Related Work

- O. Strichman: "Deciding Disjunctive Linear Arithmetic with SAT" [Str04]
  - Eager reduction to SAT, using a procedure based on FM

Related Work

- O. Strichman: "Deciding Disjunctive Linear Arithmetic with SAT" [Str04]
  - Eager reduction to SAT, using a procedure based on FM
- N. Eén, A. Biere: "Effective Preprocessing in SAT Through Variable and Clause Elimination" [EB05]
  - SATElite algorithm for SAT preprocessing

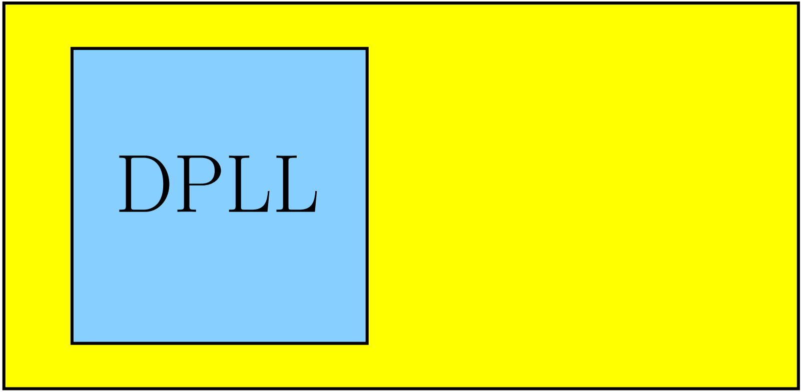
Related Work

- O. Strichman: "Deciding Disjunctive Linear Arithmetic with SAT" [Str04]
  - Eager reduction to SAT, using a procedure based on FM
- N. Eén, A. Biere: "Effective Preprocessing in SAT Through Variable and Clause Elimination" [EB05]
  - SATElite algorithm for SAT preprocessing
- K. McMillan et al.: "Generalizing DPLL to Richer Logics" [MKS09]
  - "Shadow Rule" similar to our notion of SMT($\mathcal{LRA}$) resolution: one application of the shadow rule is equiv. to many applications of SMT($\mathcal{LRA}$) resolution

opensmt

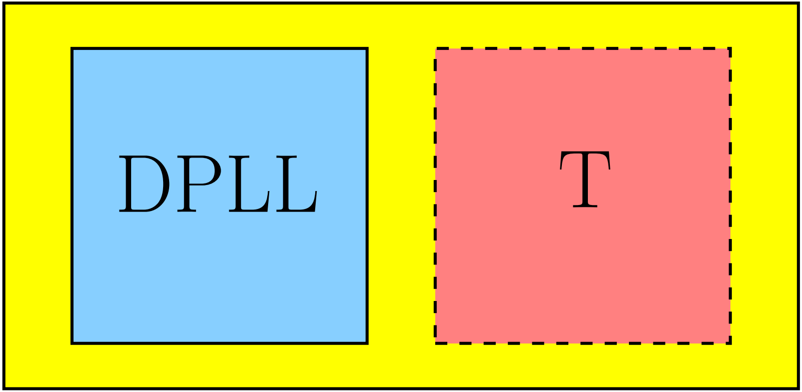
Outline

$$\mathrm{e(DPLL(T))} \approx \boxed{\mathrm{e(T)}}$$



Extending OpenSMT

$$e(DPLL(T)) \approx \boxed{e(T)}$$



- To create an empty template for a new theory solver use script `create_tsolver.sh`

- To create an empty template for a new theory solver use script `create_tsolver.sh`
  - Creates a new directory with basic class files

Extending OpenSMT

- To create an empty template for a new theory solver use script `create_tsolver.sh`
  - Creates a new directory with basic class files
  - Creates/Sets up Makefile for compilation

opensmt

Extending OpenSMT

- To create an empty template for a new theory solver use script `create_tsolver.sh`
  - Creates a new directory with basic class files
  - Creates/Sets up Makefile for compilation
  - Adds a new logic

Extending OpenSMT

- To create an empty template for a new theory solver use script `create_tsolver.sh`
  - Creates a new directory with basic class files
  - Creates/Sets up Makefile for compilation
  - Adds a new logic
  - Integrates the new solver with the core

Extending OpenSMT

- To create an empty template for a new theory solver use script `create_tsolver.sh`
  - Creates a new directory with basic class files
  - Creates/Sets up Makefile for compilation
  - Adds a new logic
  - Integrates the new solver with the core
  - Basically, it creates an incomplete solver

Extending OpenSMT

```
class TSolver
{
  void inform        ( Enode * );
  bool assertLit     ( Enode * );
  bool check         ( bool );
  void pushBktPoint  ( );
  void popBktPoint   ( );
  bool belongsToT    ( Enode * );
  void computeModel  ( );

  vector< Enode * > & explanation;
  vector< Enode * > & deductions;
  vector< Enode * > & suggestions;
}
```

opensmt

Other Features

- C API: you can compile a library and call OpenSMT via the C API

Other Features

- C API: you can compile a library and call OpenSMT via the C API
- Interpolation: given two mutually unsatisfiable formulæ A and B, an interpolant [Cra57] is a formula I such that
  - $A \to I$
  - $B \wedge I$ is unsatisfiable
  - I is defined on the variables that are common to A and B

opensmt

Other Features

- C API: you can compile a library and call OpenSMT via the C API
- Interpolation: given two mutually unsatisfiable formulæ A and B, an interpolant [Cra57] is a formula I such that
    - $A \rightarrow I$
    - $B \wedge I$ is unsatisfiable
    - I is defined on the variables that are common to A and B
- Basically I is an "overapproximation" of A, that is still unsatisfiable with B

opensmt

Other Features

- C API: you can compile a library and call OpenSMT via the C API
- Interpolation: given two mutually unsatisfiable formulæ A and B, an interpolant [Cra57] is a formula I such that
  - $A \rightarrow I$
  - $B \wedge I$ is unsatisfiable
  - I is defined on the variables that are common to A and B
- Basically I is an "overapproximation" of A, that is still unsatisfiable with B
- It has applications in Model Checking [McM04]

opensmt

Other Features

- C API: you can compile a library and call OpenSMT via the C API
- Interpolation: given two mutually unsatisfiable formulæ A and B, an interpolant [Cra57] is a formula I such that
  - $A \to I$
  - $B \wedge I$ is unsatisfiable
  - I is defined on the variables that are common to A and B
- Basically I is an "overapproximation" of A, that is still unsatisfiable with B
- It has applications in Model Checking [McM04]
- OpenSMT can compute interpolants for propositional formulæ and some arithmetic fragments

Conclusion

- OpenSMT website
  http://www.verify.inf.unisi.ch/opensmt
- Source repository
  http://code.google.com/p/opensmt
- Discussion group
  http://groups.google.com/group/opensmt

W. Craig.
Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory.
J. Symb. Log., pages 269–285, 1957.

B. Dutertre and L. M. de Moura.
A Fast Linear-Arithmetic Solver for DPLL(T).
In CAV'06, pages 81–94, 2006.

Martin Davis and Hilary Putnam.
A Computing Procedure for Quantification Theory.
J. ACM, 7(3):201–215, 1960.

N. Eén and A. Biere.
Effective Preprocessing in SAT Through Variable and Clause Elimination.
In SAT, pages 61–75, 2005.

J.B.J. Fourier.
Solution d'une question particulire du calcul des angalits.
Oevres, II:314–328, 1826.

K. L. McMillan.
Applications of Craig Interpolation to Model Checking.
In CSL, pages 22–23, 2004.

K. L. McMillan, A. Kuehlmann, and M. Sagiv.
Generalizing dpll to richer logics.
In CAV, pages 462–476, 2009.

O. Strichman.

opensmt

Deciding Disjunctive Linear Arithmetic with SAT.
CoRR, 2004.