Understanding and using SAT and SMT solvers

Erika Ábrahám
RWTH Aachen University, Germany

14th Summer School on Formal Techniques
May 24-30, 2025

The Xmas problem

There are three types of Xmas presents Santa Claus can make.

- Santa Claus wants to reduce the overhead by making only two types.
- He needs at least 100 presents.
- He needs at least 5 of either type 1 or type 2.
- He needs at least 10 of the third type.
- Each present of type 1, 2, and 3 need 1, 2, resp. 5 minutes to make.
- Santa Claus is late, and he has only 3 hours left.
- Each present of type 1, 2, and 3 costs 3, 2, resp. 1 EUR.
- He has 300 EUR for presents in total.

The Xmas problem

There are three types of Xmas presents Santa Claus can make.

- Santa Claus wants to reduce the overhead by making only two types.
- He needs at least 100 presents.
- He needs at least 5 of either type 1 or type 2.
- He needs at least 10 of the third type.
- Each present of type 1, 2, and 3 need 1, 2, resp. 5 minutes to make.
- Santa Claus is late, and he has only 3 hours left.
- Each present of type 1, 2, and 3 costs 3, 2, resp. 1 EUR.
- He has 300 EUR for presents in total.

$$
\begin{aligned}
(p_1 = 0 \lor p_2 = 0 \lor p_3 = 0) \land p_1 + p_2 + p_3 \geq 100 \land \\
(p_1 \geq 5 \lor p_2 \geq 5) \land p_3 \geq 10 \land p_1 + 2p_2 + 5p_3 \leq 180 \land \\
3p_1 + 2p_2 + p_3 \leq 300
\end{aligned}
$$

The Xmas problem

There are three types of Xmas presents Santa Claus can make.

- Santa Claus wants to reduce the overhead by making only two types.
- He needs at least 100 presents.
- He needs at least 5 of either type 1 or type 2.
- He needs at least 10 of the third type.
- Each present of type 1, 2, and 3 need 1, 2, resp. 5 minutes to make.
- Santa Claus is late, and he has only 3 hours left.
- Each present of type 1, 2, and 3 costs 3, 2, resp. 1 EUR.
- He has 300 EUR for presents in total.

$$
\begin{aligned}
(p_1 = 0 \vee p_2 = 0 \vee p_3 = 0) \wedge p_1 + p_2 + p_3 \geq 100 \wedge \\
(p_1 \geq 5 \vee p_2 \geq 5) \wedge p_3 \geq 10 \wedge p_1 + 2p_2 + 5p_3 \leq 180 \wedge \\
3p_1 + 2p_2 + p_3 \leq 300
\end{aligned}
$$

Logic:

The Xmas problem

There are three types of Xmas presents Santa Claus can make.

- Santa Claus wants to reduce the overhead by making only two types.
- He needs at least 100 presents.
- He needs at least 5 of either type 1 or type 2.
- He needs at least 10 of the third type.
- Each present of type 1, 2, and 3 need 1, 2, resp. 5 minutes to make.
- Santa Claus is late, and he has only 3 hours left.
- Each present of type 1, 2, and 3 costs 3, 2, resp. 1 EUR.
- He has 300 EUR for presents in total.

$$
\begin{aligned}
(p_1 = 0 \lor p_2 = 0 \lor p_3 = 0) \land p_1 + p_2 + p_3 \geq 100 \land \\
(p_1 \geq 5 \lor p_2 \geq 5) \land p_3 \geq 10 \land p_1 + 2p_2 + 5p_3 \leq 180 \land \\
3p_1 + 2p_2 + p_3 \leq 300
\end{aligned}
$$

Logic: Linear integer arithmetic.

The traveller Eve's problem

The traveller Eve's problem

Eve is eager to make scientific visits.

- She has 100 travel wishes $A_1, \ldots, A_{100}$.
- She is allowed to make only 5 travels.
- She wants to be physically at A_1.
- To coordinate a project, she needs to visit either A_2 or A_3.
- Travel A_i costs C_i EUR.
- Eve can spend up to C EUR.
- Travel A_i takes T_i days.
- Eve wants to travel at least T days.

The traveller Eve's problem

Eve is eager to make scientific visits.

- She has 100 travel wishes $A_1, \ldots, A_{100}$.
- She is allowed to make only 5 travels.
- She wants to be physically at A_1.
- To coordinate a project, she needs to visit either A_2 or A_3.
- Travel A_i costs C_i EUR.
- Eve can spend up to C EUR.
- Travel A_i takes T_i days.
- Eve wants to travel at least T days.

$$\left(\bigwedge_{i=1}^{100} \Big((a_i = 0 \wedge c_i = 0 \wedge t_i = 0) \vee (a_i = 1 \wedge c_i = C_i \wedge t_i = T_i) \Big) \right) \wedge$$

$$\left(\sum_{i=1}^{100} a_i \leq 5 \right) \wedge (a_1 = 1) \wedge (a_2 = 1 \vee a_3 = 1) \wedge \left(\sum_{i=1}^{100} c_i \leq C \right) \wedge \left(\sum_{i=1}^{100} t_i \geq T \right)$$

The traveller Eve's problem

Eve is eager to make scientific visits.

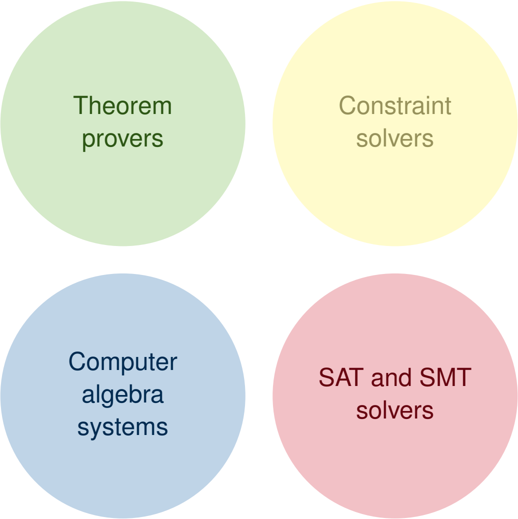
- She has 100 travel wishes $A_1, \ldots, A_{100}$.
- She is allowed to make only 5 travels.
- She wants to be physically at A_1.
- To coordinate a project, she needs to visit either A_2 or A_3.
- Travel A_i costs C_i EUR.
- Eve can spend up to C EUR.
- Travel A_i takes T_i days.
- Eve wants to travel at least T days.

$$\left(\bigwedge_{i=1}^{100} \Big((a_i = 0 \land c_i = 0 \land t_i = 0) \lor (a_i = 1 \land c_i = C_i \land t_i = T_i) \Big) \right) \land$$

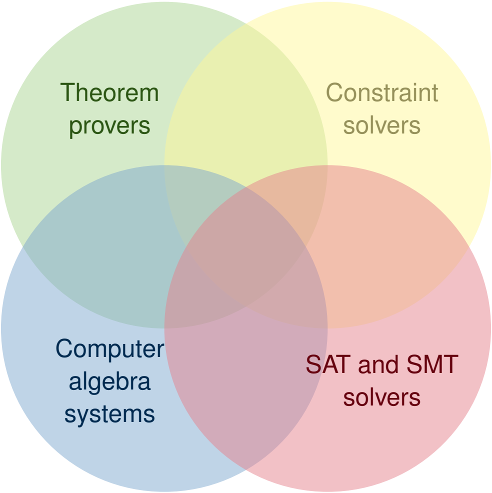
$$\left(\sum_{i=1}^{100} a_i \leq 5 \right) \land (a_1 = 1) \land (a_2 = 1 \lor a_3 = 1) \land \left(\sum_{i=1}^{100} c_i \leq C \right) \land \left(\sum_{i=1}^{100} t_i \geq T \right)$$

Logic: Mixed integer linear arithmetic.

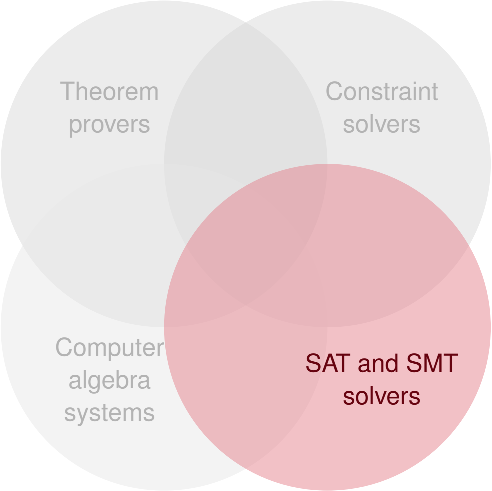
Some technologies for satisfiability checking



Some technologies for satisfiability checking

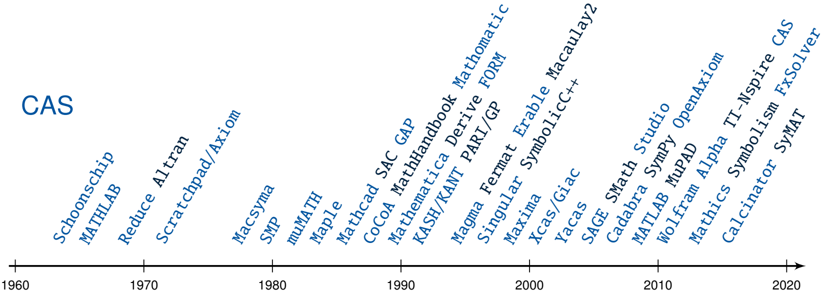


Some technologies for satisfiability checking

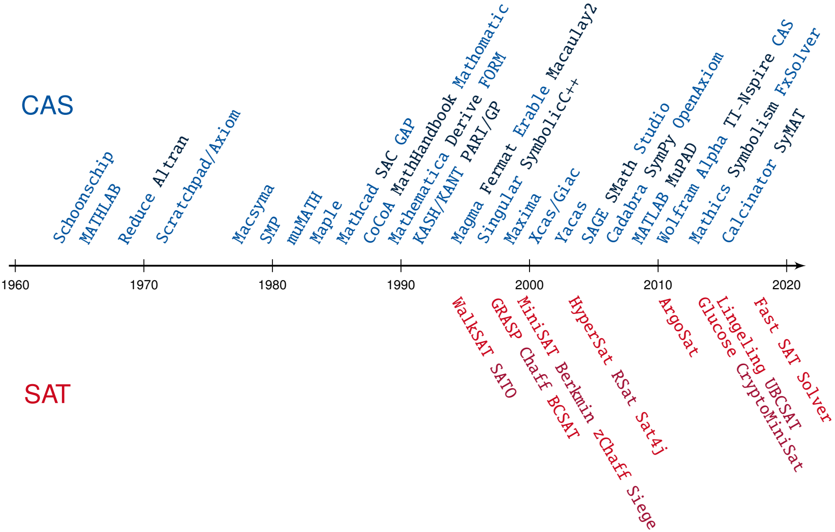


Tool development

CAS



Tool development



Satisfiability checking for propositional logic

Success story: SAT-solving

- Practical problems with millions of variables are solvable.
- A wide range of applications, e.g., verification, synthesis, combinatorial optimisation, etc.

Satisfiability checking for propositional logic

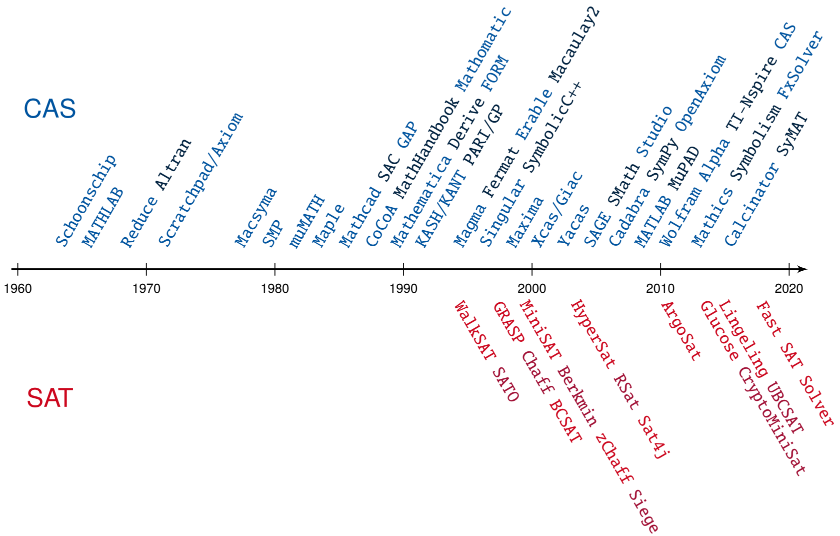
Success story: SAT-solving

- Practical problems with millions of variables are solvable.
- A wide range of applications, e.g., verification, synthesis, combinatorial optimisation, etc.

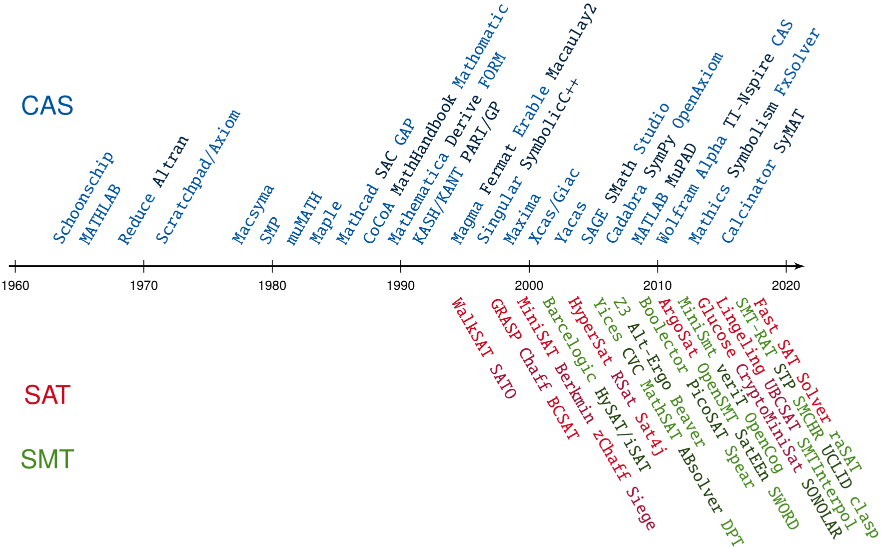
Community support:

- Standard input language.
- Large benchmark library.
- Competitions since 2002.
- SAT Live! forum as community platform, dedicated conferences, journals, etc.

Tool development



Tool development



Satisfiability modulo theories (SMT) solving:

- Propositional logic is sometimes too weak for modelling.
- Increase expressiveness: quantifier-free (QF) fragments of first-order logic over various theories.

Satisfiability modulo theories (SMT) solving

Satisfiability modulo theories (SMT) solving:

- Propositional logic is sometimes too weak for modelling.
- Increase expressiveness: quantifier-free (QF) fragments of first-order logic over various theories.

Community support:

- SMT-LIB: standard input language since 2004.
- Large (~ 250.000) benchmark library.
- Competitions since 2005.

Contents

Syntax of propositional logic

Abstract syntax of well-formed propositional logic formulae:

$$\varphi \; := \; a \; | \; (\neg\varphi) \; | \; (\varphi \wedge \varphi)$$

where *AP* is a set of (atomic) propositions (Boolean variables) and $a \in AP$.

Syntax of propositional logic

Abstract syntax of well-formed propositional logic formulae:

$$\varphi \;:=\; a \;\mid\; (\neg\varphi) \;\mid\; (\varphi \wedge \varphi)$$

where *AP* is a set of (atomic) propositions (Boolean variables) and $a \in AP$.
Syntactic sugar:

Syntax of propositional logic

Abstract syntax of well-formed propositional logic formulae:

$$\varphi := a \mid (\neg\varphi) \mid (\varphi \wedge \varphi)$$

where *AP* is a set of (atomic) propositions (Boolean variables) and $a \in AP$.

Syntactic sugar:

$$\bot \quad := $$

Syntax of propositional logic

Abstract syntax of well-formed propositional logic formulae:

$$\varphi \;:=\; a \;\mid\; (\neg\varphi) \;\mid\; (\varphi \wedge \varphi)$$

where *AP* is a set of (atomic) propositions (Boolean variables) and $a \in AP$.
Syntactic sugar:

$$\bot \quad := (a \wedge \neg a)$$

Syntax of propositional logic

Abstract syntax of well-formed propositional logic formulae:

$$\varphi \ := \ a \ | \ (\neg\varphi) \ | \ (\varphi \wedge \varphi)$$

where *AP* is a set of (atomic) propositions (Boolean variables) and $a \in AP$.

Syntactic sugar:

$$\begin{array}{ll} \bot & := (a \wedge \neg a) \\ \top & := \end{array}$$

Syntax of propositional logic

Abstract syntax of well-formed propositional logic formulae:

$$\varphi \ := \ a \ \mid \ (\neg\varphi) \ \mid \ (\varphi \wedge \varphi)$$

where *AP* is a set of (atomic) propositions (Boolean variables) and $a \in AP$.

Syntactic sugar:

$$\begin{aligned} \bot \quad &:= (a \wedge \neg a) \\ \top \quad &:= (a \vee \neg a) \end{aligned}$$

Syntax of propositional logic

Abstract syntax of well-formed propositional logic formulae:

$$\varphi \ := \ a \ | \ (\neg\varphi) \ | \ (\varphi \wedge \varphi)$$

where *AP* is a set of (atomic) propositions (Boolean variables) and $a \in AP$.

Syntactic sugar:

$$\begin{array}{rcl} \bot & := & (a \wedge \neg a) \\ \top & := & (a \vee \neg a) \\ (\ \varphi_1 \ \vee \ \varphi_2\ ) & := & \end{array}$$

Syntax of propositional logic

Abstract syntax of well-formed propositional logic formulae:

$$\varphi \; := \; a \; | \; (\neg\varphi) \; | \; (\varphi \wedge \varphi)$$

where *AP* is a set of (atomic) propositions (Boolean variables) and $a \in AP$.

Syntactic sugar:

$$\begin{aligned} \bot &:= (a \wedge \neg a) \\ \top &:= (a \vee \neg a) \\ (\varphi_1 \vee \varphi_2) &:= \neg((\neg\varphi_1) \wedge (\neg\varphi_2)) \end{aligned}$$

Syntax of propositional logic

Abstract syntax of well-formed propositional logic formulae:

$$\varphi \;:=\; a \mid (\neg\varphi) \mid (\varphi \wedge \varphi)$$

where *AP* is a set of (atomic) propositions (Boolean variables) and $a \in AP$.

Syntactic sugar:

$$\begin{aligned}
\bot &:= (a \wedge \neg a)\\
\top &:= (a \vee \neg a)\\
(\varphi_1 \vee \varphi_2) &:= \neg((\neg\varphi_1) \wedge (\neg\varphi_2))\\
(\varphi_1 \rightarrow \varphi_2) &:=
\end{aligned}$$

Syntax of propositional logic

Abstract syntax of well-formed propositional logic formulae:

$$\varphi \ := \ a \ | \ (\neg\varphi) \ | \ (\varphi \land \varphi)$$

where *AP* is a set of (atomic) propositions (Boolean variables) and $a \in AP$.

Syntactic sugar:

$$\begin{aligned}
\bot \quad &:= (a \land \neg a) \\
\top \quad &:= (a \lor \neg a) \\
(\ \varphi_1 \ \lor \ \varphi_2\ ) &:= \neg((\neg\varphi_1) \land (\neg\varphi_2)) \\
(\ \varphi_1 \ \rightarrow \ \varphi_2\ ) &:= ((\neg\varphi_1) \lor \varphi_2)
\end{aligned}$$

Syntax of propositional logic

Abstract syntax of well-formed propositional logic formulae:

$$\varphi \;:=\; a \mid (\neg\varphi) \mid (\varphi \wedge \varphi)$$

where *AP* is a set of (atomic) propositions (Boolean variables) and $a \in AP$.

Syntactic sugar:

$$\begin{aligned}
\bot &:= (a \wedge \neg a)\\
\top &:= (a \vee \neg a)\\
(\varphi_1 \vee \varphi_2) &:= \neg((\neg\varphi_1) \wedge (\neg\varphi_2))\\
(\varphi_1 \rightarrow \varphi_2) &:= ((\neg\varphi_1) \vee \varphi_2)\\
(\varphi_1 \leftrightarrow \varphi_2) &:=
\end{aligned}$$

Syntax of propositional logic

Abstract syntax of well-formed propositional logic formulae:

$$\varphi \; := \; a \;\mid\; (\neg\varphi) \;\mid\; (\varphi \land \varphi)$$

where *AP* is a set of (atomic) propositions (Boolean variables) and $a \in AP$.

Syntactic sugar:

$$\begin{aligned}
\bot &:= (a \land \neg a)\\
\top &:= (a \lor \neg a)\\
(\varphi_1 \lor \varphi_2) &:= \neg((\neg\varphi_1) \land (\neg\varphi_2))\\
(\varphi_1 \rightarrow \varphi_2) &:= ((\neg\varphi_1) \lor \varphi_2)\\
(\varphi_1 \leftrightarrow \varphi_2) &:= ((\varphi_1 \rightarrow \varphi_2) \land (\varphi_2 \rightarrow \varphi_1))
\end{aligned}$$

Syntax of propositional logic

Abstract syntax of well-formed propositional logic formulae:

$$\varphi \;:=\; a \;\mid\; (\neg\varphi) \;\mid\; (\varphi \wedge \varphi)$$

where *AP* is a set of (atomic) propositions (Boolean variables) and $a \in AP$.

Syntactic sugar:

$$\begin{array}{ccccl}
 & \bot & & & := (a \wedge \neg a) \\
 & \top & & & := (a \vee \neg a) \\
(\; \varphi_1 & \vee & \varphi_2 \;) & & := \neg((\neg\varphi_1) \wedge (\neg\varphi_2)) \\
(\; \varphi_1 & \rightarrow & \varphi_2 \;) & & := ((\neg\varphi_1) \vee \varphi_2) \\
(\; \varphi_1 & \leftrightarrow & \varphi_2 \;) & & := ((\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)) \\
(\; \varphi_1 & \oplus & \varphi_2 \;) & & :=
\end{array}$$

Syntax of propositional logic

Abstract syntax of well-formed propositional logic formulae:

$$\varphi \ := \ a \ | \ (\neg\varphi) \ | \ (\varphi \wedge \varphi)$$

where *AP* is a set of (atomic) propositions (Boolean variables) and $a \in AP$.
Syntactic sugar:

$$\begin{array}{lllll} & & \bot & & := (a \wedge \neg a) \\ & & \top & & := (a \vee \neg a) \\ (& \varphi_1 & \vee & \varphi_2 &) := \neg((\neg\varphi_1) \wedge (\neg\varphi_2)) \\ (& \varphi_1 & \rightarrow & \varphi_2 &) := ((\neg\varphi_1) \vee \varphi_2) \\ (& \varphi_1 & \leftrightarrow & \varphi_2 &) := ((\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)) \\ (& \varphi_1 & \oplus & \varphi_2 &) := (\varphi_1 \leftrightarrow (\neg\varphi_2)) \end{array}$$

Semantics of propositional logic

- Truth tables define the semantics (=meaning) of the operators.
  They can be used to define the semantics of formulae inductively over their structure.

Semantics of propositional logic

- Truth tables define the semantics (=meaning) of the operators.
  They can be used to define the semantics of formulae inductively over their structure.
- Convention: 0= false, 1= true

Semantics of propositional logic

- Truth tables define the semantics (=meaning) of the operators. They can be used to define the semantics of formulae inductively over their structure.
- Convention: 0= false, 1= true

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$	$p \oplus q$
0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	0	0	1
1	1	0	1	1	1	1	0

Semantics of propositional logic

- Truth tables define the semantics (=meaning) of the operators. They can be used to define the semantics of formulae inductively over their structure.
- Convention: 0= false, 1= true

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$	$p \oplus q$
0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	0	0	1
1	1	0	1	1	1	1	0

Each possible assignment is covered by a line of the truth table.
α satisfies φ iff in the line for α and the column for φ the entry is 1.

Conjunctive normal form

- A literal is either a variable or the negation of a variable.
- A clause is a disjunction of literals.
- A formula in Conjunctive Normal Form (CNF) is a conjunction of clauses.

Conjunctive normal form

- A literal is either a variable or the negation of a variable.
- A clause is a disjunction of literals.
- A formula in Conjunctive Normal Form (CNF) is a conjunction of clauses.
- Every propositional logic formula can be converted to an equi-satisfiable CNF in linear time and space on the cost of (linearly many) new variables.

Tseitin's CNF encoding

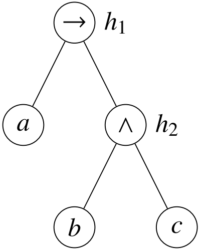
Consider the formula $\varphi = (a \rightarrow (b \wedge c))$.

Tseitin's encoding:

$(h_1 \leftrightarrow (a \rightarrow h_2))\wedge$

$(h_2 \leftrightarrow (b \wedge c))\wedge$

(h_1)



Tseitin's CNF encoding

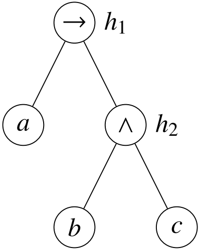
Consider the formula $\varphi = (a \rightarrow (b \wedge c))$.

Tseitin's encoding:

$(h_1 \leftrightarrow (a \rightarrow h_2))\wedge$

$(h_2 \leftrightarrow (b \wedge c))\wedge$

(h_1)



- Each node's encoding has a CNF representation with 3 or 4 clauses.

  $h_1 \leftrightarrow (a \rightarrow h_2)$ in CNF: $(h_1 \vee a) \wedge (h_1 \vee \neg h_2) \wedge (\neg h_1 \vee \neg a \vee h_2)$
  $h_2 \leftrightarrow (b \wedge c)$ in CNF: $(\neg h_2 \vee b) \wedge (\neg h_2 \vee c) \wedge (h_2 \vee \neg b \vee \neg c)$

Contents

Satisfiability problem

Given:

- Propositional logic formula φ in CNF.

Question:

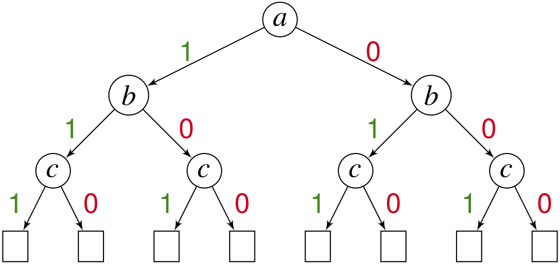
- Is φ satisfiable?

  (Is there a model for φ?)

1. Exploration: Exploration

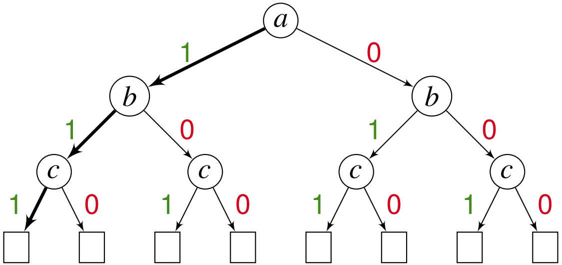
$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$

$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$

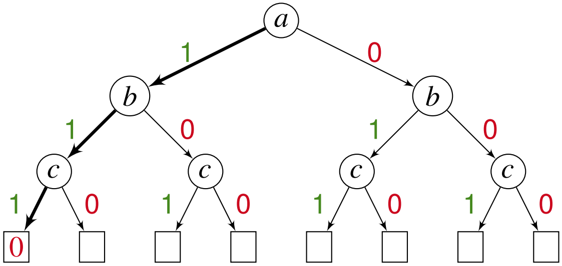


1. Exploration: Exploration

$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$

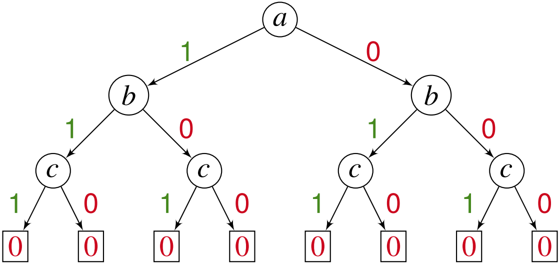


$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$

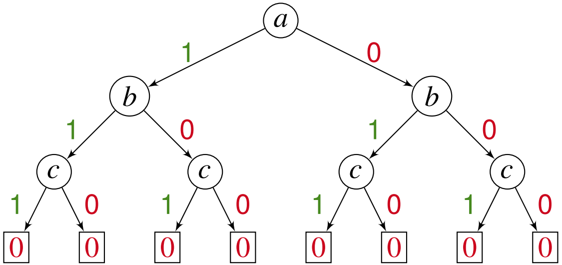


1. Exploration: Exploration

$(a \lor c) \land (\neg a \lor c) \land (a \lor \neg c) \land (\neg a \lor \neg c) \land (a \lor b) \land (\neg a \lor b) \land (a \lor \neg b \lor \neg c)$



$$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$$



unsatisfiable problem in n variables $\rightsquigarrow$ ALWAYS 2^n assignments need to be tested

2. Proof system: Boolean resolution

$$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$$

2. Proof system: Boolean resolution

$$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$$

$$\frac{(x \vee \ell_1 \vee \ldots \vee \ell_n) \qquad (\neg x \vee \ell'_1 \vee \ldots \vee \ell'_m)}{(\ell_1 \vee \ldots \vee \ell_n \vee \ell'_1 \vee \ldots \vee \ell'_m)} \; \texttt{Rule}_{\texttt{res}}$$

2. Proof system: Boolean resolution

$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$

$(b \vee c)$

$$\frac{(x \vee \ell_1 \vee \ldots \vee \ell_n) \qquad (\neg x \vee \ell'_1 \vee \ldots \vee \ell'_m)}{(\ell_1 \vee \ldots \vee \ell_n \vee \ell'_1 \vee \ldots \vee \ell'_m)} \; \texttt{Rule}_{\texttt{res}}$$

2. Proof system: Boolean resolution

$$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$$

$$(c)$$

$$\frac{(x \vee \ell_1 \vee \ldots \vee \ell_n) \qquad (\neg x \vee \ell'_1 \vee \ldots \vee \ell'_m)}{(\ell_1 \vee \ldots \vee \ell_n \vee \ell'_1 \vee \ldots \vee \ell'_m)} \ \mathtt{Rule_{res}}$$

$$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$$

(true)

$$\frac{(\, x \vee \ell_1 \vee \ldots \vee \ell_n \,) \qquad (\, \neg x \vee \ell'_1 \vee \ldots \vee \ell'_m \,)}{(\, \ell_1 \vee \ldots \vee \ell_n \;\vee\; \ell'_1 \vee \ldots \vee \ell'_m \,)} \; \mathtt{Rule}_{\mathtt{res}}$$

2. Proof system: Boolean resolution

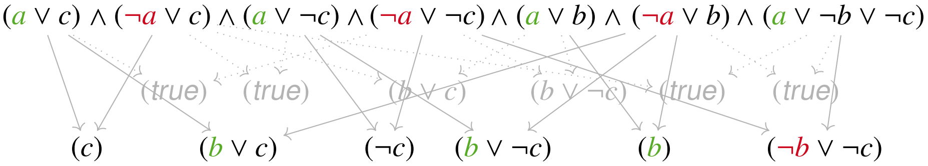
$$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$$

$$\frac{(\, x \vee \ell_1 \vee \ldots \vee \ell_n \,) \qquad (\, \neg x \vee \ell'_1 \vee \ldots \vee \ell'_m \,)}{(\, \ell_1 \vee \ldots \vee \ell_n \vee \ell'_1 \vee \ldots \vee \ell'_m \,)} \; \texttt{Rule}_{\texttt{res}}$$

$$\exists x. \quad C \wedge C_x \wedge C_{\neg x}$$

$$\equiv$$

$$C \wedge \bigwedge_{c_x \in C_x} \bigwedge_{c_{\neg x} \in C_{\neg x}} \textit{resolvent}(c_x, c_{\neg x}, x)$$



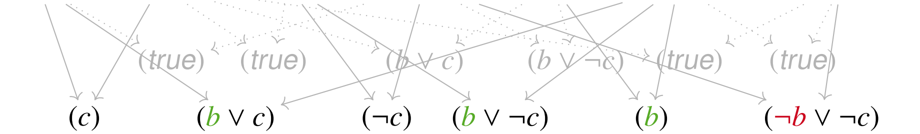
$$\frac{(\,x \lor \ell_1 \lor \ldots \lor \ell_n\,) \qquad (\,\neg x \lor \ell'_1 \lor \ldots \lor \ell'_m\,)}{(\,\ell_1 \lor \ldots \lor \ell_n \;\lor\; \ell'_1 \lor \ldots \lor \ell'_m\,)}\ \texttt{Rule}_{\texttt{res}}$$

$$\exists x.\quad C \;\land\; C_x \;\land\; C_{\neg x}$$

$$\equiv$$

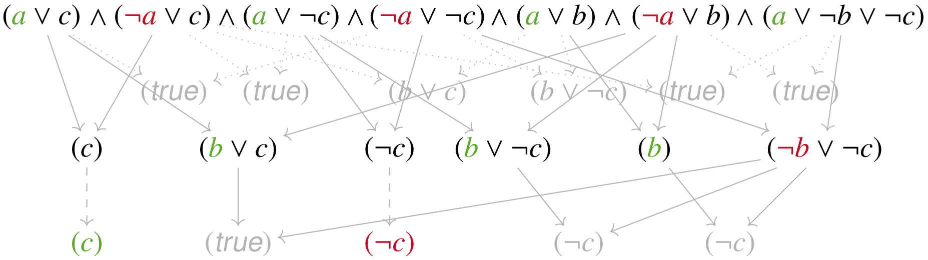
$$C \;\land\; \bigwedge_{c_x \in C_x} \;\bigwedge_{c_{\neg x} \in C_{\neg x}} \textit{resolvent}(c_x, c_{\neg x}, x)$$

$$(a \lor c) \land (\neg a \lor c) \land (a \lor \neg c) \land (\neg a \lor \neg c) \land (a \lor b) \land (\neg a \lor b) \land (a \lor \neg b \lor \neg c)$$

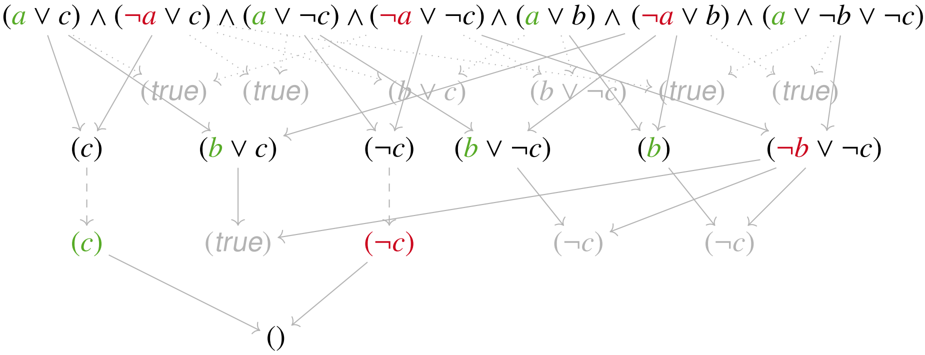


$$\exists x. \quad C \land C_x \land C_{\neg x}$$
$$\equiv$$
$$C \land \bigwedge_{c_x \in C_x} \bigwedge_{c_{\neg x} \in C_{\neg x}} \textit{resolvent}(c_x, c_{\neg x}, x)$$

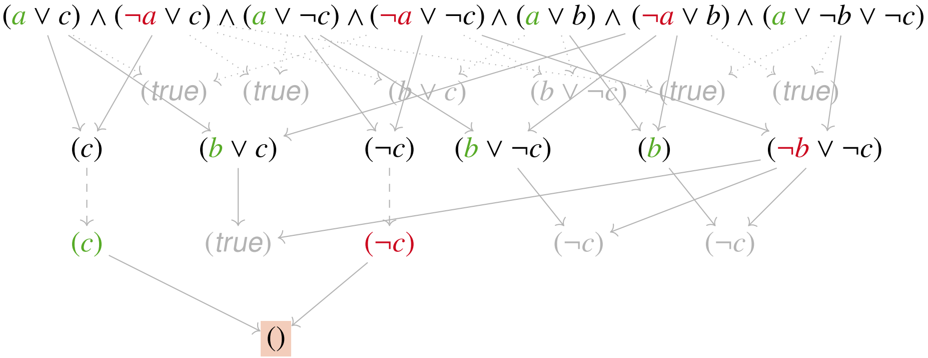
2. Proof system: Boolean resolution



$$\exists x.\quad C \;\wedge\; C_x \;\wedge\; C_{\neg x}$$
$$\equiv$$
$$C \;\wedge \bigwedge_{c_x \in C_x} \; \bigwedge_{c_{\neg x} \in C_{\neg x}} \textit{resolvent}(c_x, c_{\neg x}, x)$$

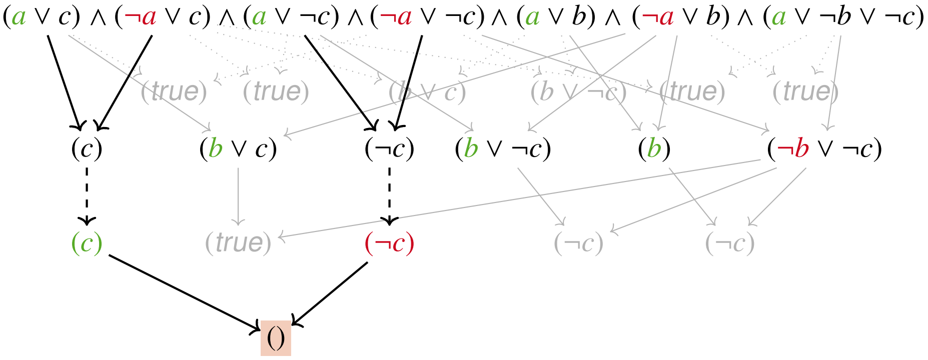


$$\exists x.\quad C \wedge C_x \wedge C_{\neg x} \;\equiv\; C \wedge \bigwedge_{c_x \in C_x} \bigwedge_{c_{\neg x} \in C_{\neg x}} \textit{resolvent}(c_x, c_{\neg x}, x)$$



$$\exists x.\quad C \wedge C_x \wedge C_{\neg x}$$
$$\equiv$$
$$C \wedge \bigwedge_{c_x \in C_x} \bigwedge_{c_{\neg x} \in C_{\neg x}} \textit{resolvent}(c_x, c_{\neg x}, x)$$

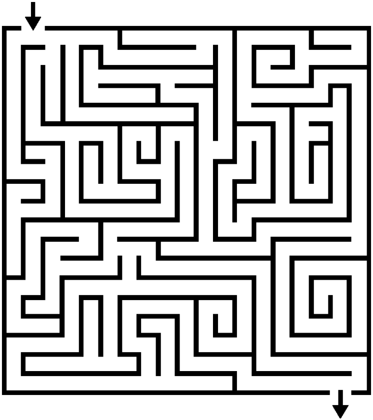
2. Proof system: Boolean resolution



$$\exists x. \quad C \wedge C_x \wedge C_{\neg x}$$
$$\equiv$$
$$C \wedge \bigwedge_{c_x \in C_x} \bigwedge_{c_{\neg x} \in C_{\neg x}} \textit{resolvent}(c_x, c_{\neg x}, x)$$

SAT solving: The DPLL+CDCL idea

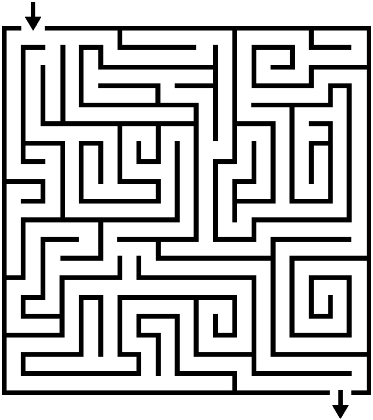
[Davis et al., '60/61] [Marques-Silva et al., '96]



SAT solving: The DPLL+CDCL idea

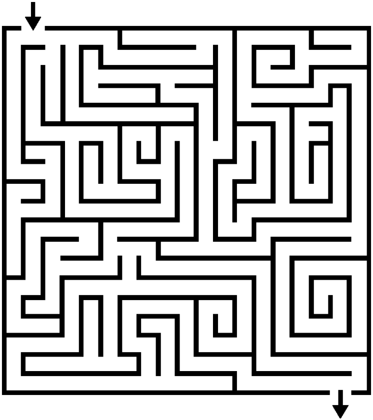
[Davis et al., '60/61] [Marques-Silva et al., '96]

Proof system



SAT solving: The DPLL+CDCL idea

[Davis et al., '60/61] [Marques-Silva et al., '96]



SAT solving: The DPLL+CDCL idea

[Davis et al., '60/61] [Marques-Silva et al., '96]

Exploration

SAT solving: The DPLL+CDCL idea

[Davis et al., ’60/61] [Marques-Silva et al., ’96]

Exploration

Look-ahead

SAT solving: The DPLL+CDCL idea

[Davis et al., ’60/61] [Marques-Silva et al., ’96]

Exploration

Look-ahead

Proof system

The DPLL+CDCL algorithm

```
if (!BCP()) return UNSAT;
while (true)
{
        if (!decide()) return SAT;
        while (!BCP())
                if (!resolve_conflict()) return UNSAT;
}
```

The DPLL+CDCL algorithm

```
if (!BCP()) return UNSAT;
while (true)
{
        if (!decide()) return SAT;
        while (!BCP())
                if (!resolve_conflict()) return UNSAT;
}
```

Boolean constraint propagation.
Return false if reached a conflict.

The DPLL+CDCL algorithm

Choose the next variable and value.
Return false if all variables are assigned.

```
if (!BCP()) return UNSAT;
while (true)
{
        if (!decide()) return SAT;
        while (!BCP())
                if (!resolve_conflict()) return UNSAT;
}
```

Boolean constraint propagation.
Return false if reached a conflict.

The DPLL+CDCL algorithm

```
if (!BCP()) return UNSAT;
while (true)
{
    if (!decide()) return SAT;
    while (!BCP())
        if (!resolve_conflict()) return UNSAT;
}
```

Choose the next variable and value.
Return false if all variables are assigned.

Boolean constraint propagation.
Return false if reached a conflict.

Conflict resolution and backtracking. Return false if impossible.

Status of a clause

- Assume in the following: all literals in a clause have different variables

Status of a clause

- Assume in the following: all literals in a clause have different variables
- Given a (partial) assignment, a clause can be

satisfied:	at least one literal is satisfied
unsatisfied:	all literals are assigned but none are statisfied
unit:	all but one literals are assigned but none are satisfied
unresolved:	all other cases

Example :

x_1	x_2	x_3	$c = (x_1 \vee x_2 \vee x_3)$
1	0		satisfied
0	0	0	unsatisfied
0	0		unit
	0		unresolved

BCP: Unit clauses are used to imply consequences of decisions.

Status of a clause

- Assume in the following: all literals in a clause have different variables
- Given a (partial) assignment, a clause can be

satisfied:	at least one literal is satisfied
unsatisfied:	all literals are assigned but none are statisfied
unit:	all but one literals are assigned but none are satisfied
unresolved:	all other cases

Example :

x_1	x_2	x_3	$c = (x_1 \lor x_2 \lor x_3)$
1	0		satisfied
0	0	0	unsatisfied
0	0		unit
	0		unresolved

BCP: Unit clauses are used to imply consequences of decisions.

Some notations:

Decision Level (DL) is a counter for decisions

Antecedent(ℓ): unit clause implying the value of literal ℓ (nil if decision)

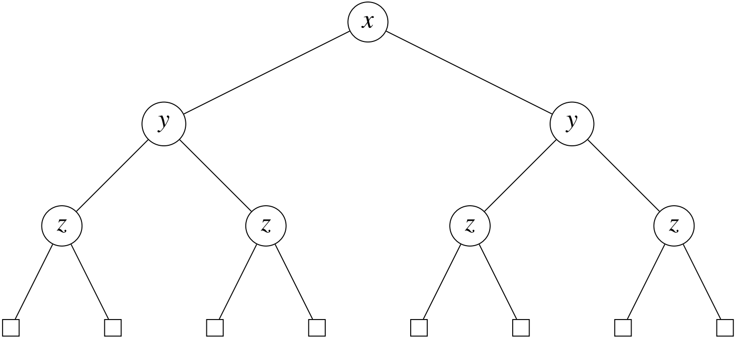
DPLL: Exploration with propagation

$$\underbrace{(\neg x \vee y \vee z)}_{c_1} \wedge \underbrace{(y \vee \neg z)}_{c_2} \wedge \underbrace{(\neg x \vee \neg y)}_{c_3}$$

DPLL: Exploration with propagation

$$\underbrace{(\neg x \vee y \vee z)}_{c_1} \wedge \underbrace{(y \vee \neg z)}_{c_2} \wedge \underbrace{(\neg x \vee \neg y)}_{c_3}$$

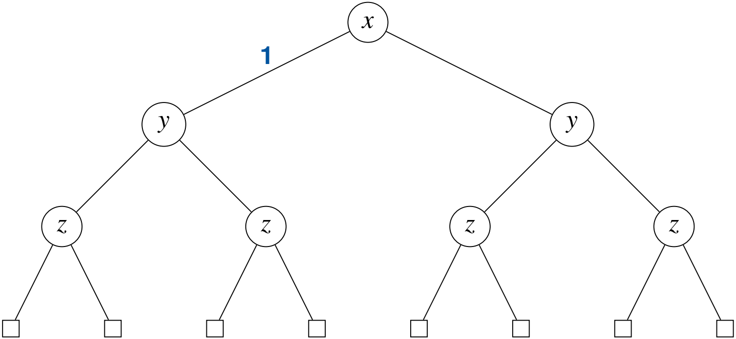
Static variable order $x < y < z$, sign: try positive first



DPLL: Exploration with propagation

$$\underbrace{(\neg x \vee y \vee z)}_{c_1} \wedge \underbrace{(y \vee \neg z)}_{c_2} \wedge \underbrace{(\neg x \vee \neg y)}_{c_3}$$

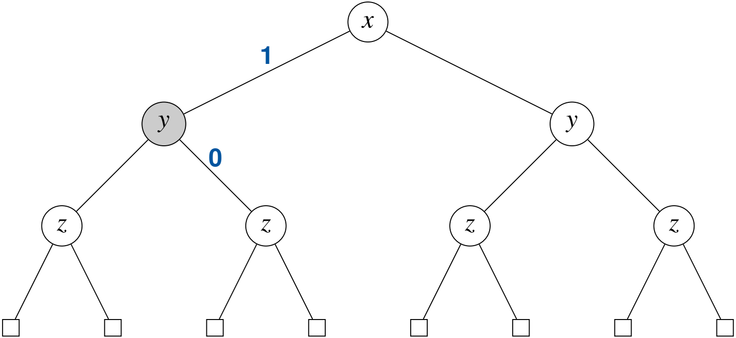
Static variable order $x < y < z$, sign: try positive first



DPLL: Exploration with propagation

$$\underbrace{(\neg x \vee y \vee z)}_{c_1} \wedge \underbrace{(y \vee \neg z)}_{c_2} \wedge \underbrace{(\neg x \vee \neg y)}_{c_3}$$

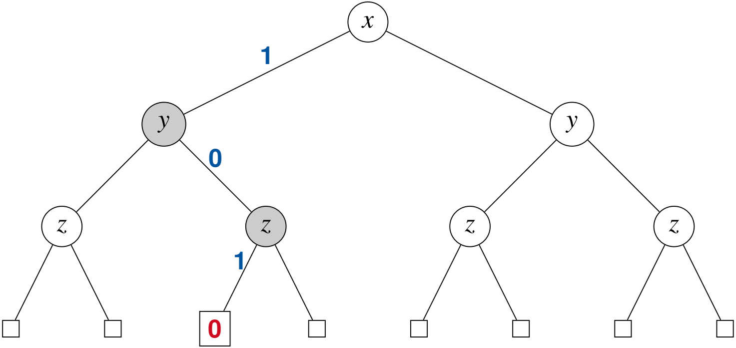
Static variable order $x < y < z$, sign: try positive first



DPLL: Exploration with propagation

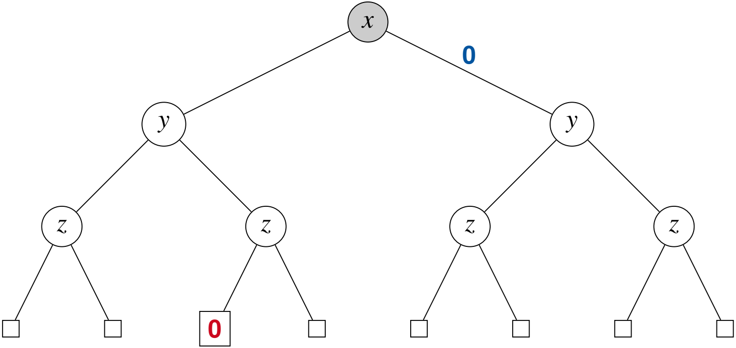
$$\underbrace{(\neg x \vee y \vee z)}_{c_1} \wedge \underbrace{(y \vee \neg z)}_{c_2} \wedge \underbrace{(\neg x \vee \neg y)}_{c_3}$$

Static variable order $x < y < z$, sign: try positive first



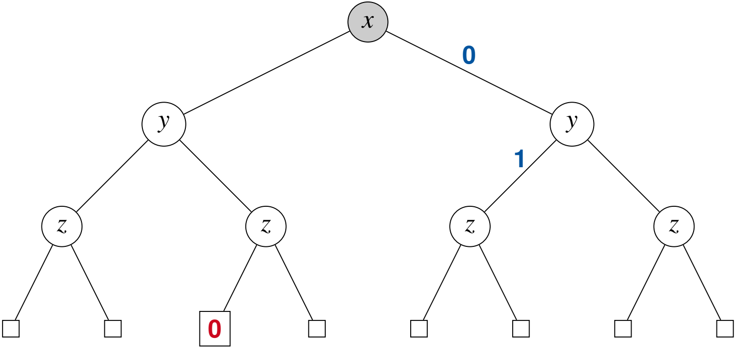
$$\underbrace{(\neg x \vee y \vee z)}_{c_1} \wedge \underbrace{(y \vee \neg z)}_{c_2} \wedge \underbrace{(\neg x \vee \neg y)}_{c_3}$$

Static variable order $x < y < z$, sign: try positive first



$$\underbrace{(\neg x \vee y \vee z)}_{c_1} \wedge \underbrace{(y \vee \neg z)}_{c_2} \wedge \underbrace{(\neg x \vee \neg y)}_{c_3}$$

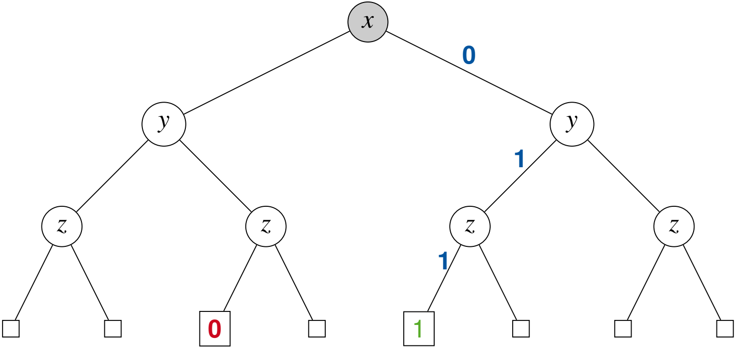
Static variable order $x < y < z$, sign: try positive first



DPLL: Exploration with propagation

$$\underbrace{(\neg x \vee y \vee z)}_{c_1} \wedge \underbrace{(y \vee \neg z)}_{c_2} \wedge \underbrace{(\neg x \vee \neg y)}_{c_3}$$

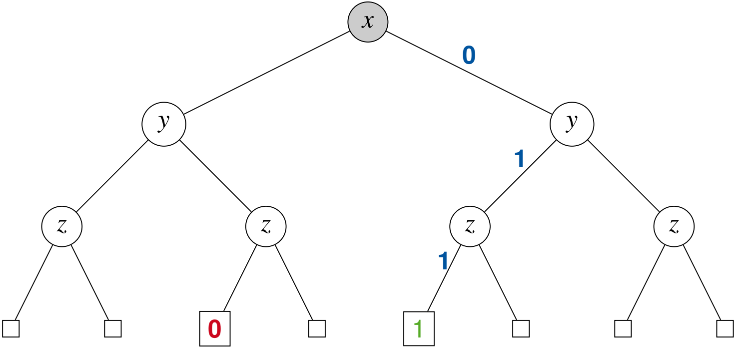
Static variable order $x < y < z$, sign: try positive first



DPLL: Exploration with propagation

$$\underbrace{(\neg x \lor y \lor z)}_{c_1} \land \underbrace{(y \lor \neg z)}_{c_2} \land \underbrace{(\neg x \lor \neg y)}_{c_3}$$

Static variable order $x < y < z$, sign: try positive first



Efficient propagation with the watched literal scheme.

The DPLL+CDCL idea [Davis et al., '60/61] [Marques-Silva et al., '96]

Exploration:	$\mathbb{B}$-decision
Look-ahead:	$\mathbb{B}$-propagation
Proof system:	$\mathbb{B}$-conflict resolution

The DPLL+CDCL idea [Davis et al., ’60/61] [Marques-Silva et al., ’96]

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

$$(a \vee b) \wedge (\neg b \vee c) \wedge (\neg b \vee \neg c)$$

The DPLL+CDCL idea [Davis et al., ’60/61] [Marques-Silva et al., ’96]

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

$$(a \vee b) \wedge (\neg b \vee c) \wedge (\neg b \vee \neg c)$$

$\mathbb{B}$-propagate -

The DPLL+CDCL idea [Davis et al., ’60/61] [Marques-Silva et al., ’96]

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

$$(a \vee b) \wedge (\neg b \vee c) \wedge (\neg b \vee \neg c)$$

$\mathbb{B}$-propagate -

$\mathbb{B}$-decision a = *false*

The DPLL+CDCL idea [Davis et al., ’60/61] [Marques-Silva et al., ’96]

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

$$(a \vee b) \wedge (\neg b \vee c) \wedge (\neg b \vee \neg c)$$

$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	a = *false*
$\mathbb{B}$-propagate	

The DPLL+CDCL idea [Davis et al., '60/61] [Marques-Silva et al., '96]

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

$$(a \vee b) \wedge (\neg b \vee c) \wedge (\neg b \vee \neg c)$$

$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	$a =$ *false*
$\mathbb{B}$-propagate	$b =$ *true*

The DPLL+CDCL idea [Davis et al., ’60/61] [Marques-Silva et al., ’96]

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

$$(a \vee b) \wedge (\neg b \vee c) \wedge (\neg b \vee \neg c)$$

$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	a = *false*
$\mathbb{B}$-propagate	b = *true*
	c = *true*

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

$$(a \vee b) \wedge (\neg b \vee c) \wedge (\neg b \vee \neg c)$$

$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	a = *false*
$\mathbb{B}$-propagate	b = *true*
	c = *true*

$\mathbb{B}$-conflict resolution

The DPLL+CDCL idea [Davis et al., '60/61] [Marques-Silva et al., '96]

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

$$(a \vee b) \wedge (\neg b \vee c) \wedge (\neg b \vee \neg c)$$

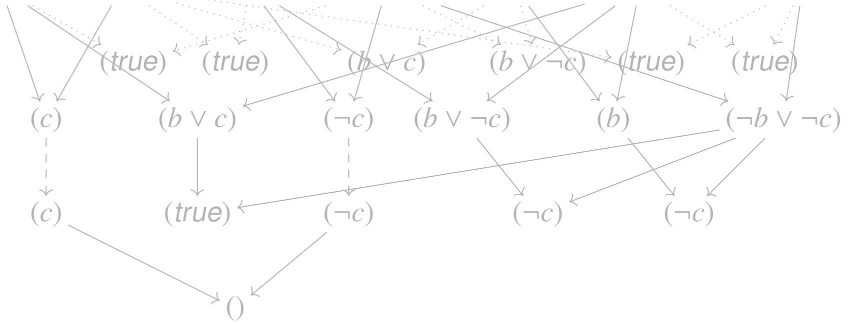
$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	$a =$ *false*
$\mathbb{B}$-propagate	$b =$ *true*
	$c =$ *true*

$\mathbb{B}$-conflict resolution

$$\frac{(\neg b \vee \neg c)\ (\neg b \vee c)}{(\neg b)}$$

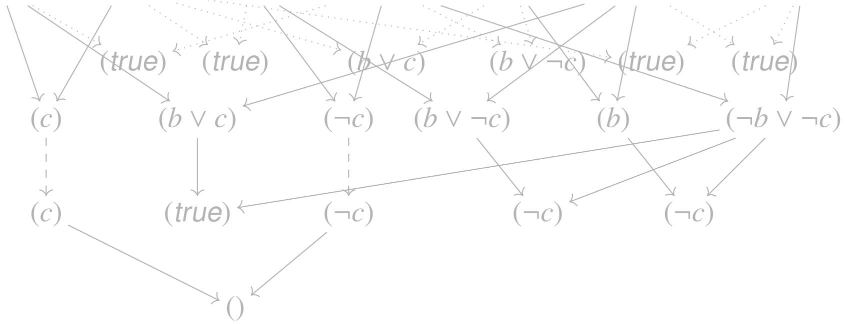
Resolution example revisited

$$(a \lor c) \land (\neg a \lor c) \land (a \lor \neg c) \land (\neg a \lor \neg c) \land (a \lor b) \land (\neg a \lor b) \land (a \lor \neg b \lor \neg c)$$



Resolution example revisited

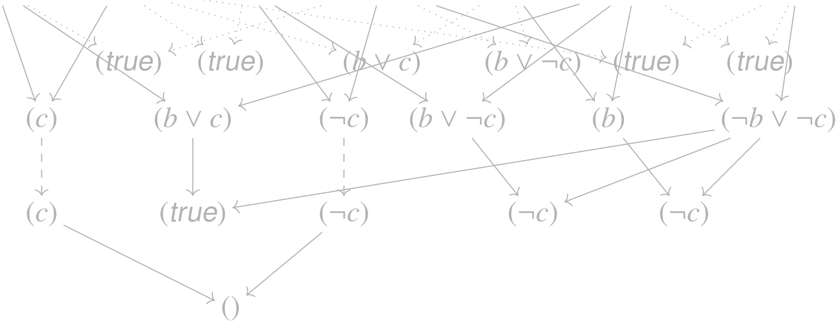
$$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$$



$\mathbb{B}$-propagate -

Resolution example revisited

$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$

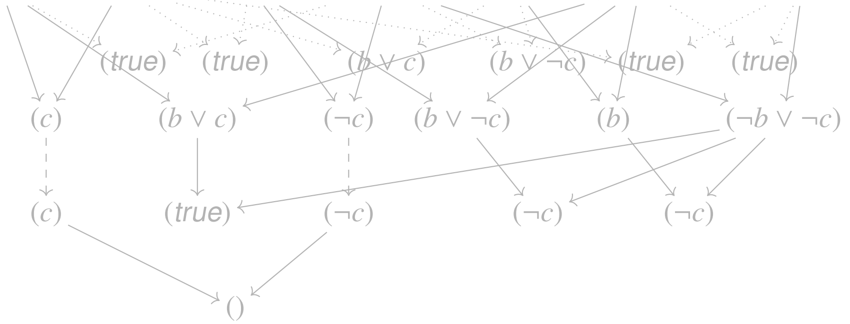


$\mathbb{B}$-propagate -

$\mathbb{B}$-decide $\neg c$

Resolution example revisited

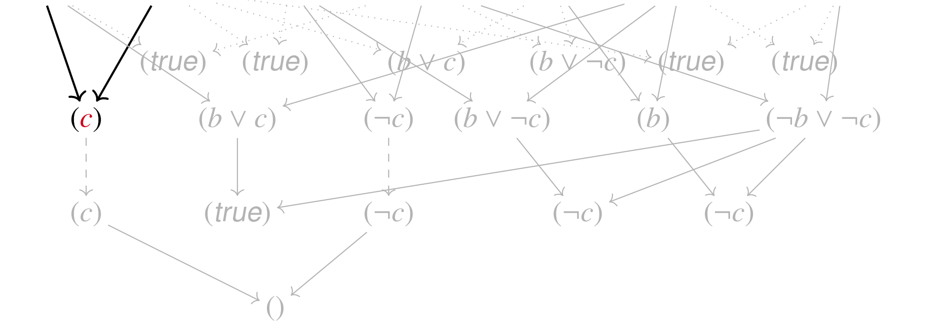
$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$



$\mathbb{B}$-propagate	-
$\mathbb{B}$-decide	$\neg c$
$\mathbb{B}$-propagate	a

Resolution example revisited

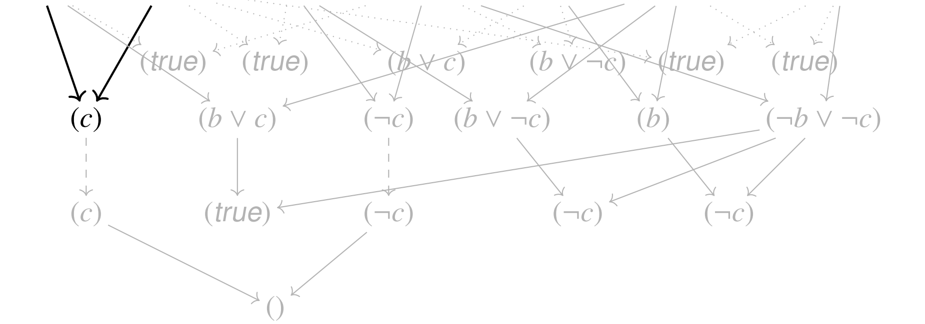
$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$



$\mathbb{B}$-propagate	-
$\mathbb{B}$-decide	$\neg c$
$\mathbb{B}$-propagate	a
$\mathbb{B}$-conflict resolution	$\dfrac{(\neg a \vee c)\ (a \vee c)}{(c)}$

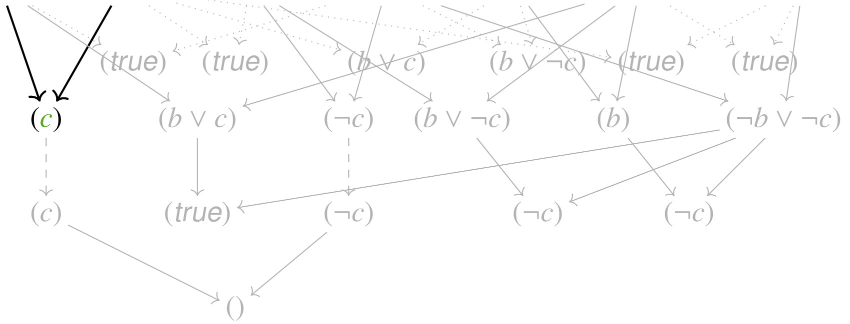
Resolution example revisited

$$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$$



Resolution example revisited

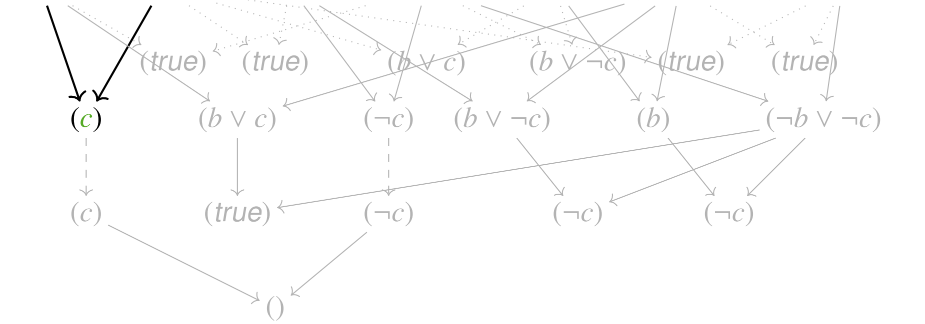
$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$



$\mathbb{B}$-propagate c

Resolution example revisited

$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$

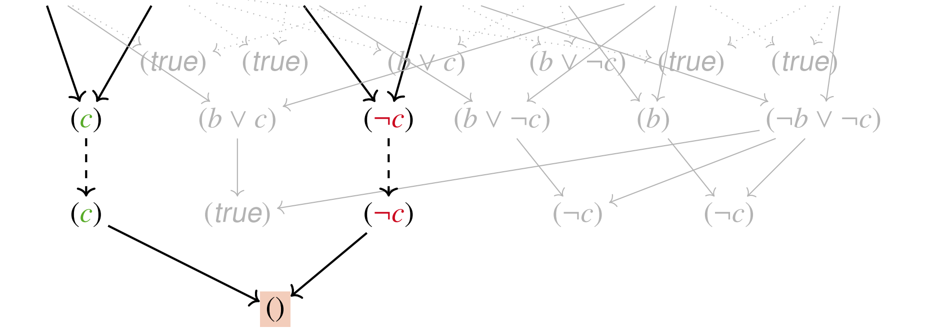


$\mathbb{B}$-propagate $\quad c$

a

Resolution example revisited

$$(a \vee c) \wedge (\neg a \vee c) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg c) \wedge (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$$



$\mathbb{B}$-propagate $\quad c$

a

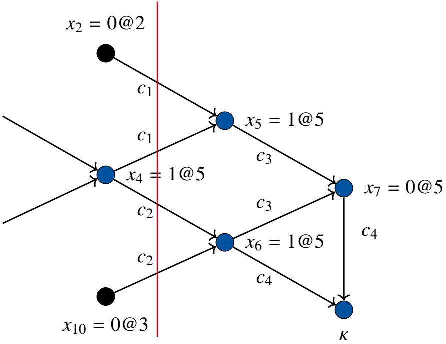
$\mathbb{B}$-conflict resolution

$$\cfrac{\cfrac{(\neg a \vee \neg c)\ (a \vee \neg c)}{(\neg c)} \quad (c)}{()}$$

Conflict clauses and (binary) resolution

- Consider the following example:

$c_1 = (\neg x_4 \vee x_2 \vee x_5)$
$c_2 = (\neg x_4 \vee x_{10} \vee x_6)$
$c_3 = (\neg x_5 \vee \neg x_6 \vee \neg x_7)$
$c_4 = (\neg x_6 \vee x_7)$
$\vdots \quad \vdots$

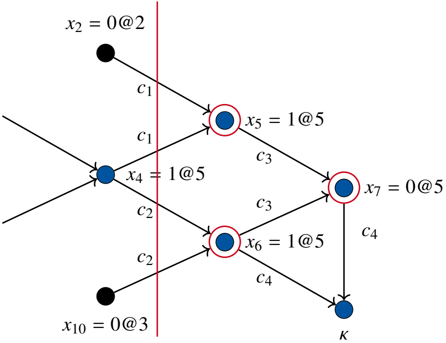


- Asserting conflict clause: $c_5 : (x_2 \vee \neg x_4 \vee x_{10})$

Conflict clauses and (binary) resolution

- Assigment order: x_4, x_5, x_6, x_7 Conflict clause: $c_5 : (x_2 \vee \neg x_4 \vee x_{10})$

$c_1 = (\neg x_4 \vee x_2 \vee x_5)$
$c_2 = (\neg x_4 \vee x_{10} \vee x_6)$
$c_3 = (\neg x_5 \vee \neg x_6 \vee \neg x_7)$
$c_4 = (\neg x_6 \vee x_7)$
$\vdots \qquad \vdots$



- Starting with the conflicting clause, apply resolution with the antecedent of the last assigned literal, until we get an asserting clause:
  - T1 = Res$(c_4, c_3, x_7) = (\neg x_5 \vee \neg x_6)$
  - T2 = Res(T1,$c_2, x_6) = (\neg x_4 \vee \neg x_5 \vee x_{10})$
  - T3 = Res(T2,$c_1, x_5) = (x_2 \vee \neg x_4 \vee x_{10})$

Unsatisfiable core

Definition

An unsatisfiable core of an unsatisfiable CNF formula is an unsatisfiable subset of the original set of clauses.

Unsatisfiable core

Definition

An unsatisfiable core of an unsatisfiable CNF formula is an unsatisfiable subset of the original set of clauses.

- The set of all original clauses is an unsatisfiable core.

Unsatisfiable core

Definition

An unsatisfiable core of an unsatisfiable CNF formula is an unsatisfiable subset of the original set of clauses.

- The set of all original clauses is an unsatisfiable core.
- The set of those original clauses that were used for resolution in conflict analysis during SAT-solving (inclusively the last conflict at decision level 0) gives us an unsatisfiable core which is in general much smaller.

Unsatisfiable core

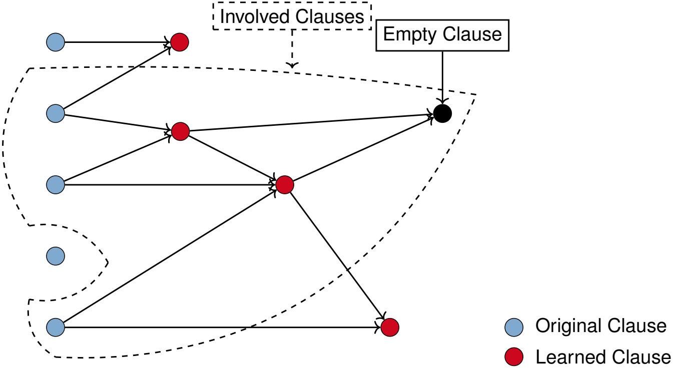
Definition

An unsatisfiable core of an unsatisfiable CNF formula is an unsatisfiable subset of the original set of clauses.

- The set of all original clauses is an unsatisfiable core.
- The set of those original clauses that were used for resolution in conflict analysis during SAT-solving (inclusively the last conflict at decision level 0) gives us an unsatisfiable core which is in general much smaller.
- However, this unsatifiable core is still not always minimal (i.e., we can remove clauses from it still having an unsatisfiable core).

The resolution graph

A resolution graph gives us more information to get a minimal unsatisfiable core.



Termination

Theorem

It is never the case that the solver enters decision level dl again with the same partial assignment.

Termination

Theorem

It is never the case that the solver enters decision level dl again with the same partial assignment.

Proof.

Define a partial order on partial assignments: $\alpha < \beta$ iff either α is an extension of β or α has more assignments at the smallest decision level at that α and β do not agree.
BCP decreases the order, conflict-driven backtracking also. Since the order always decreases during the search, the theorem holds. □

Decision heuristics: VSIDS

- VSIDS (variable state independent decaying sum)
- Gives priority to variables involved in recent conflicts.
- “Involved” can have different definitions. We take those variables that occur in clauses used for conflict resolution.

Decision heuristics: VSIDS

- VSIDS (variable state independent decaying sum)
- Gives priority to variables involved in recent conflicts.
- "Involved" can have different definitions. We take those variables that occur in clauses used for conflict resolution.

1. Each variable has a counter initialized to 0.
2. We define an increment value (e.g., 1).
3. When a conflict occurs, we increase the counter of each variable, that occurs in at least one clause used for conflict resolution, by the increment value.
   Afterwards we increase the increment value (e.g., by 1).
4. For decisions, the unassigned variable with the highest counter is chosen.
5. Periodically, all the counters and the increment value are divided by a constant.

Decision heuristics: VSIDS

VSIDS is a ’quasi-static’ strategy:

- static because it doesn’t depend on current assignment
- dynamic because it gradually changes. Variables that appear in recent conflicts have higher priority.

  This strategy is a conflict-driven decision strategy.

”...employing this strategy dramatically (i.e., an order of magnitude) improved performance...”

Contents

Example 1: Seminar topic assignment

- n participants
- n topics
- Set of preferences $E \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$

  $(p, t) \in E$ means: participant p would take topic t

Example 1: Seminar topic assignment

- n participants
- n topics
- Set of preferences $E \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$

  $(p, t) \in E$ means: participant p would take topic t

- Q: Can we assign to each participant a topic which he/she is willing to take?

Example 1: Propositional encoding

- Notation:

Example 1: Propositional encoding

- Notation: $x_{p,t}$ = “participant p is assigned topic t”

Example 1: Propositional encoding

- Notation: $x_{p,t}$ = "participant p is assigned topic t"
- Constraints:

Example 1: Propositional encoding

- Notation: $x_{p,t}$ = "participant p is assigned topic t"
- Constraints:

  Each participant is assigned at least one topic:

Example 1: Propositional encoding

- Notation: $x_{p,t}$ = "participant p is assigned topic t"
- Constraints:

  Each participant is assigned at least one topic:

$$\bigwedge_{p=1}^{n} \left(\bigvee_{t=1}^{n} x_{p,t} \right)$$

Example 1: Propositional encoding

- Notation: $x_{p,t}$ = "participant p is assigned topic t"
- Constraints:

  Each participant is assigned at least one topic:

$$\bigwedge_{p=1}^{n}\left(\bigvee_{t=1}^{n} x_{p,t}\right)$$

  Each participant is assigned at most one topic:

Example 1: Propositional encoding

- Notation: $x_{p,t}$ = "participant p is assigned topic t"
- Constraints:

  Each participant is assigned at least one topic:

$$\bigwedge_{p=1}^{n} \left(\bigvee_{t=1}^{n} x_{p,t} \right)$$

  Each participant is assigned at most one topic:

$$\bigwedge_{p=1}^{n} \bigwedge_{t_1=1}^{n-1} \bigwedge_{t_2=t_1+1}^{n} \left(\neg x_{p,t_1} \vee \neg x_{p,t_2} \right)$$

Example 1: Propositional encoding

- Notation: $x_{p,t}$ = "participant p is assigned topic t"
- Constraints:

Each participant is assigned at least one topic:

$$\bigwedge_{p=1}^{n} \left(\bigvee_{t=1}^{n} x_{p,t} \right)$$

Each participant is assigned at most one topic:

$$\bigwedge_{p=1}^{n} \bigwedge_{t_1=1}^{n-1} \bigwedge_{t_2=t_1+1}^{n} \left(\neg x_{p,t_1} \vee \neg x_{p,t_2} \right)$$

Each participant is willing to take his/her assigned topic:

Example 1: Propositional encoding

- Notation: $x_{p,t}$ = "participant p is assigned topic t"
- Constraints:

Each participant is assigned at least one topic:

$$\bigwedge_{p=1}^{n} \left(\bigvee_{t=1}^{n} x_{p,t} \right)$$

Each participant is assigned at most one topic:

$$\bigwedge_{p=1}^{n} \bigwedge_{t_1=1}^{n-1} \bigwedge_{t_2=t_1+1}^{n} \left(\neg x_{p,t_1} \vee \neg x_{p,t_2} \right)$$

Each participant is willing to take his/her assigned topic:

$$\bigwedge_{p=1}^{n} \bigwedge_{(p,t) \notin E} \neg x_{p,t}$$

Example 1: Propositional encoding

Example 1: Propositional encoding

Each topic is assigned to at most one participant:

Example 1: Propositional encoding

Each topic is assigned to at most one participant:

$$\bigwedge_{t=1}^{n} \bigwedge_{p_1=1}^{n} \bigwedge_{p_2=p_1+1}^{n} \left(\neg x_{p_1,t} \vee \neg x_{p_2,t}\right)$$

Example 2: Placement of wedding guests

- Three chairs in a row: $1, 2, 3$
- We need to place Aunt, Sister and Father.
- Constraints:
  - Aunt doesn't want to sit near Father
  - Aunt doesn't want to sit in the left chair
  - Sister doesn't want to sit to the right of Father

Example 2: Placement of wedding guests

- Three chairs in a row: $1, 2, 3$
- We need to place Aunt, Sister and Father.
- Constraints:
  - Aunt doesn't want to sit near Father
  - Aunt doesn't want to sit in the left chair
  - Sister doesn't want to sit to the right of Father

- Q: Can we satisfy these constraints?

Example 2: Propositional encoding

Example 2: Propositional encoding

- Notation:

Example 2: Propositional encoding

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$

Example 2: Propositional encoding

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$
- Constraints:

Example 2: Propositional encoding

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$
- Constraints:

  Aunt doesn't want to sit near Father:

Example 2: Propositional encoding

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$
- Constraints:

  Aunt doesn't want to sit near Father:

$$((x_{1,1} \vee x_{1,3}) \rightarrow \neg x_{3,2}) \wedge (x_{1,2} \rightarrow (\neg x_{3,1} \wedge \neg x_{3,3}))$$

Example 2: Propositional encoding

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$
- Constraints:

  Aunt doesn't want to sit near Father:

$$((x_{1,1} \vee x_{1,3}) \rightarrow \neg x_{3,2}) \wedge (x_{1,2} \rightarrow (\neg x_{3,1} \wedge \neg x_{3,3}))$$

  Aunt doesn't want to sit in the left chair:

Example 2: Propositional encoding

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$
- Constraints:

  Aunt doesn't want to sit near Father:

$$((x_{1,1} \vee x_{1,3}) \rightarrow \neg x_{3,2}) \wedge (x_{1,2} \rightarrow (\neg x_{3,1} \wedge \neg x_{3,3}))$$

  Aunt doesn't want to sit in the left chair:

$$\neg x_{1,1}$$

Example 2: Propositional encoding

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$
- Constraints:

  Aunt doesn't want to sit near Father:

$$((x_{1,1} \vee x_{1,3}) \rightarrow \neg x_{3,2}) \wedge (x_{1,2} \rightarrow (\neg x_{3,1} \wedge \neg x_{3,3}))$$

  Aunt doesn't want to sit in the left chair:

$$\neg x_{1,1}$$

  Sister doesn't want to sit to the right of Father:

Example 2: Propositional encoding

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$
- Constraints:

  Aunt doesn't want to sit near Father:

$$((x_{1,1} \vee x_{1,3}) \rightarrow \neg x_{3,2}) \wedge (x_{1,2} \rightarrow (\neg x_{3,1} \wedge \neg x_{3,3}))$$

  Aunt doesn't want to sit in the left chair:

$$\neg x_{1,1}$$

  Sister doesn't want to sit to the right of Father:

$$(x_{3,1} \rightarrow \neg x_{2,2}) \wedge (x_{3,2} \rightarrow \neg x_{2,3})$$

Example 2: Propositional encoding

Example 2: Propositional encoding

Each person is placed:

Example 2: Propositional encoding

Each person is placed:

$$(x_{1,1} \vee x_{1,2} \vee x_{1,3}) \wedge (x_{2,1} \vee x_{2,2} \vee x_{2,3}) \wedge (x_{3,1} \vee x_{3,2} \vee x_{3,3})$$

$$\bigwedge_{p=1}^{3} \bigvee_{c=1}^{3} x_{p,c}$$

Example 2: Propositional encoding

Each person is placed:

$$(x_{1,1} \vee x_{1,2} \vee x_{1,3}) \wedge (x_{2,1} \vee x_{2,2} \vee x_{2,3}) \wedge (x_{3,1} \vee x_{3,2} \vee x_{3,3})$$

$$\bigwedge_{p=1}^{3} \bigvee_{c=1}^{3} x_{p,c}$$

At most one person per chair:

Example 2: Propositional encoding

Each person is placed:

$$(x_{1,1} \vee x_{1,2} \vee x_{1,3}) \wedge (x_{2,1} \vee x_{2,2} \vee x_{2,3}) \wedge (x_{3,1} \vee x_{3,2} \vee x_{3,3})$$

$$\bigwedge_{p=1}^{3} \bigvee_{c=1}^{3} x_{p,c}$$

At most one person per chair:

$$\bigwedge_{p1=1}^{3} \bigwedge_{p2=p1+1}^{3} \bigwedge_{c=1}^{3} (\neg x_{p1,c} \vee \neg x_{p2,c})$$

Example 3: Assignment of frequencies

- n radio stations
- For each station assign one of k transmission frequencies, $k < n$.
- E – set of pairs of stations, that are too close to have the same frequency.

Example 3: Assignment of frequencies

- n radio stations
- For each station assign one of k transmission frequencies, $k < n$.
- E – set of pairs of stations, that are too close to have the same frequency.
- Q: Can we assign to each station one frequency, such that no station pairs from E have the same frequency?

Example 3: Propositional encoding

- Notation:

Example 3: Propositional encoding

- Notation:

  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$

Example 3: Propositional encoding

- Notation:

  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$
- Constraints:

Example 3: Propositional encoding

- Notation:
  $x_{s,f}$ = “station s is assigned frequency f” for $1 \leq s \leq n$, $1 \leq f \leq k$
- Constraints:
  Every station is assigned at least one frequency:

Example 3: Propositional encoding

- Notation:

  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$
- Constraints:

  Every station is assigned at least one frequency:

$$\bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right)$$

Example 3: Propositional encoding

- Notation:

  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$
- Constraints:

  Every station is assigned at least one frequency:

$$\bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right)$$

  Every station is assigned at most one frequency:

Example 3: Propositional encoding

- Notation:
  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$
- Constraints:
  Every station is assigned at least one frequency:

$$\bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right)$$

  Every station is assigned at most one frequency:

$$\bigwedge_{s=1}^{n} \bigwedge_{f1=1}^{k-1} \bigwedge_{f2=f1+1}^{k} \left(\neg x_{s,f1} \vee \neg x_{s,f2} \right)$$

Example 3: Propositional encoding

- Notation:

  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$
- Constraints:

  Every station is assigned at least one frequency:

$$\bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right)$$

  Every station is assigned at most one frequency:

$$\bigwedge_{s=1}^{n} \bigwedge_{f1=1}^{k-1} \bigwedge_{f2=f1+1}^{k} \left(\neg x_{s,f1} \vee \neg x_{s,f2} \right)$$

  Close stations are not assigned the same frequency:

Example 3: Propositional encoding

- Notation:

  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$
- Constraints:

  Every station is assigned at least one frequency:

$$\bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right)$$

  Every station is assigned at most one frequency:

$$\bigwedge_{s=1}^{n} \bigwedge_{f1=1}^{k-1} \bigwedge_{f2=f1+1}^{k} \left(\neg x_{s,f1} \vee \neg x_{s,f2} \right)$$

  Close stations are not assigned the same frequency:

  For each $(s1, s2) \in E$,

$$\bigwedge_{f=1}^{k} \left(\neg x_{s1,f} \vee \neg x_{s2,f} \right)$$

Contents

You need to have installed...

- Python
- Z3

  `https://github.com/exercism/z3/blob/main/docs/INSTALLATION.md`

SAT encodings

- Suppose we can solve the satisfiability problem... how can this help us?

SAT encodings

- Suppose we can solve the satisfiability problem... how can this help us?

- There are numerous problems in the industry that are solved via the satisfiability problem of propositional logic
  - Logistics
  - Planning
  - Electronic Design Automation industry
  - Cryptography
  - ...

DIMACS input syntax for SAT solvers

The DIMACS format for SAT solvers has three types of lines:

- header: “p cnf n m” in which
  - n denotes the highest variable index and
  - m the number of clauses.
- clauses: a sequence of integers ending with “0”
- comments: any line starting with “c “

Example:

		c example
		p cnf 2 4
$(a \vee b)$	$\wedge$	$1 \quad 2 \quad 0$
$(\neg a \vee b)$	$\wedge$	$-1 \quad 2 \quad 0$
$(a \vee \neg b)$	$\wedge$	$1 \quad -2 \quad 0$
$(\neg a \vee \neg b)$	$\wedge$	$-1 \quad -2 \quad 0$

Example 2 (wedding): DIMACS format

Notation: Aunt = 1, Sister = 2, Father = 3
Left chair = 1, Middle chair = 2, Right chair = 3
$x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$

Constraints:

(1) $((x_{1,1} \vee x_{1,3}) \rightarrow \neg x_{3,2}) \wedge (x_{1,2} \rightarrow (\neg x_{3,1} \wedge \neg x_{3,3}))$

(2) $\neg x_{1,1}$

(3) $(x_{3,1} \rightarrow \neg x_{2,2}) \wedge (x_{3,2} \rightarrow \neg x_{2,3})$

(4) $\bigwedge_{p=1}^{3} \bigvee_{c=1}^{3} x_{p,c}$

(5) $\bigwedge_{p_1=1}^{3} \bigwedge_{p_2=p_1+1}^{3} \bigwedge_{c=1}^{3} (\neg x_{p_1,c} \vee \neg x_{p_2,c})$

	c example
	p cnf 2 4
$(a \vee b) \quad \wedge$	$1 \quad 2 \qquad 0$
$\ldots$	$\ldots$

Example 3 (frequencies): DIMACS format

$$
\begin{array}{ll}
(1) & \bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right) \\
(2) & \bigwedge_{s=1}^{n} \bigwedge_{f_1=1}^{k-1} \bigwedge_{f_2=f_1+1}^{k} \left(\neg x_{s,f_1} \vee \neg x_{s,f_2} \right) \\
(3) & \forall (s_1, s_2) \in E.\ \bigwedge_{f=1}^{k} \left(\neg x_{s_1,f} \vee \neg x_{s_2,f} \right)
\end{array}
$$

Example 3 (frequencies): DIMACS format

$$\begin{array}{ll}(1) & \bigwedge_{s=1}^{n}\left(\bigvee_{f=1}^{k} x_{s,f}\right)\\ (2) & \bigwedge_{s=1}^{n}\bigwedge_{f_1=1}^{k-1}\bigwedge_{f_2=f_1+1}^{k}\left(\neg x_{s,f_1} \vee \neg x_{s,f_2}\right)\\ (3) & \forall (s_1,s_2)\in E.\ \bigwedge_{f=1}^{k}\left(\neg x_{s_1,f} \vee \neg x_{s_2,f}\right)\end{array}$$

Assume that n^2 ($n \in \mathbb{N}_{>0}$) stations are arranged in a grid with the coordinates (i,j), $1 \leq i,j \leq n$, and

$$\begin{array}{rcl} E & = & \{((i,j),(i+1,j)) \mid 1 \leq i \leq n-1 \wedge 1 \leq j \leq n\} \cup \\ & & \{((i,j),(i,j+1)) \mid 1 \leq i \leq n \wedge 1 \leq j \leq n-1\}. \end{array}$$

Example 3 (frequencies): DIMACS format

$$
\begin{array}{ll}
(1) & \bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right) \\
(2) & \bigwedge_{s=1}^{n} \bigwedge_{f_1=1}^{k-1} \bigwedge_{f_2=f_1+1}^{k} \left(\neg x_{s,f_1} \vee \neg x_{s,f_2} \right) \\
(3) & \forall (s_1, s_2) \in E. \ \bigwedge_{f=1}^{k} \left(\neg x_{s_1,f} \vee \neg x_{s_2,f} \right)
\end{array}
$$

Assume that n^2 ($n \in \mathbb{N}_{>0}$) stations are arranged in a grid with the coordinates (i,j), $1 \leq i,j \leq n$, and

$$
\begin{array}{rcl}
E & = & \{((i,j),(i+1,j)) \,|\, 1 \leq i \leq n-1 \wedge 1 \leq j \leq n\} \cup \\
& & \{((i,j),(i,j+1)) \,|\, 1 \leq i \leq n \wedge 1 \leq j \leq n-1\}.
\end{array}
$$

Write a Python program that writes for an input n the **DIMACS** encoding for $k = 1, \ldots, n$ into an external file, and check them (by manually calling z3 on them) to identify the minimal k necessary for a solution.

Example 3: DIMACS

```
import argparse
import sys
try:
    parser = argparse.ArgumentParser()
    parser.add_argument("n", help="number of stations", type=int)
    args = parser.parse_args()
    n = args.n
except:
    e = sys.exc_info()[0]
    print(e)

for k in range(n):
    names = []
    for i in range(n):
        names_i = []
        for j in range(k+1):
            name = str(i*(k+1)+j+1)
            names_i.append(name)
        names.append(names_i)
    clauses = ""
    counter = 0
    for i in range(n):
        for j in range(k+1):
            clauses += names[i][j] + " "
        clauses += "0\n"
        counter += 1
    file = open("frequencies" + str(k+1) + ".dimacs","w")
    file.write("p cnf " + str(n*(k+1)) + " " + str(counter) + "\n")
    file.write(clauses)
    file.close()
```

Solving propositional logic with SMT solvers

- SMT-LIB format:
  https://microsoft.github.io/z3guide/docs/logic/propositional-logic
- Python interface:
  https://ericpony.github.io/z3py-tutorial/guide-examples.htm
- Both:
  https://cvc5.github.io/tutorials/beginners/

SMT-LIB2 format

Boolean SMT-LIB example

```
(set-logic QF_UF)
(declare-const p Bool)
(assert (and p (not p)))
(check-sat)
```

Example 3 (frequencies): SMT-LIB format

$$
\begin{array}{ll}
(1) & \bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right) \\
(2) & \bigwedge_{s=1}^{n} \bigwedge_{f_1=1}^{k-1} \bigwedge_{f_2=f_1+1}^{k} \left(\neg x_{s,f_1} \vee \neg x_{s,f_2} \right) \\
(3) & \forall (s_1, s_2) \in E.\ \bigwedge_{f=1}^{k} \left(\neg x_{s_1,f} \vee \neg x_{s_2,f} \right)
\end{array}
$$

Assume that n^2 ($n \in \mathbb{N}_{>0}$) stations are arranged in a grid with the coordinates (i,j), $1 \leq i,j \leq n$, and

$$
\begin{array}{rcl}
E & = & \{((i,j),(i{+}1,j)) \,|\, 1 \leq i \leq n{-}1 \wedge 1 \leq j \leq n\} \cup \\
 & & \{((i,j),(i,j{+}1)) \,|\, 1 \leq i \leq n \wedge 1 \leq j \leq n{-}1\}\,.
\end{array}
$$

Example 3 (frequencies): SMT-LIB format

$$
\begin{array}{ll}
(1) & \bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right) \\
(2) & \bigwedge_{s=1}^{n} \bigwedge_{f_1=1}^{k-1} \bigwedge_{f_2=f_1+1}^{k} \left(\neg x_{s,f_1} \vee \neg x_{s,f_2} \right) \\
(3) & \forall (s_1, s_2) \in E.\ \bigwedge_{f=1}^{k} \left(\neg x_{s_1,f} \vee \neg x_{s_2,f} \right)
\end{array}
$$

Assume that n^2 ($n \in \mathbb{N}_{>0}$) stations are arranged in a grid with the coordinates (i, j), $1 \leq i, j \leq n$, and

$$
\begin{aligned}
E \quad = \quad & \{((i,j),(i+1,j)) \mid 1 \leq i \leq n-1 \wedge 1 \leq j \leq n\} \cup \\
& \{((i,j),(i,j+1)) \mid 1 \leq i \leq n \wedge 1 \leq j \leq n-1\}\,.
\end{aligned}
$$

Write a Python program that writes for an input n the **SMT-LIB2** encoding for $k = 1, \ldots, n$ into an external file, and check them (by manually calling z3 on them) to identify the minimal k necessary for a solution.

Example 3: SMT-LIB2

```
import argparse
import sys
try:
     parser = argparse.ArgumentParser()
     parser.add_argument("n", help="number of stations", type=int)
     args = parser.parse_args()
     n = args.n
except:
    e = sys.exc_info()[0]
    print(e)

names = []
for i in range(n):
    names_i = []
    for j in range(n):
        name = "a_" + str(i+1) + "_" + str(j+1);
        names_i.append(name)
    names.append(names_i)
for k in range(n):
    file = open("frequencies" + str(k+1) + ".smt2","w")
    file.write("(set-logic QF_UF)\n")
    for i in range(n):
        for j in range(k+1):
            file.write("(declare-const " + names[i][j] + " Bool)\n")
    for i in range(n):
        file.write("(assert (or")
        for j in range(k+1):
            file.write(" " + names[i][j])
        file.write("))\n")
    file.write("(check-sat)\n")
    file.write("(exit)\n")
    file.close()
```

Solving propositional logic with SMT solvers

- SMT-LIB format:
  https://microsoft.github.io/z3guide/docs/logic/propositional-logic
- Python interface:
  https://ericpony.github.io/z3py-tutorial/guide-examples.htm
- Both:
  https://cvc5.github.io/tutorials/beginners/

Example 3: Python API

$$
\begin{array}{ll}
(1) & \bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right) \\
(2) & \bigwedge_{s=1}^{n} \bigwedge_{f_1=1}^{k-1} \bigwedge_{f_2=f_1+1}^{k} \left(\neg x_{s,f_1} \vee \neg x_{s,f_2} \right) \\
(3) & \forall (s_1, s_2) \in E.\ \bigwedge_{f=1}^{k} \left(\neg x_{s_1,f} \vee \neg x_{s_2,f} \right)
\end{array}
$$

Assume that n^2 ($n \in \mathbb{N}_{>0}$) stations are arranged in a grid with the coordinates (i,j), $1 \leq i,j \leq n$, and

$$
\begin{array}{rcl}
E & = & \{((i,j),(i{+}1,j)) \mid 1 \leq i \leq n{-}1 \wedge 1 \leq j \leq n\} \cup \\
 & & \{((i,j),(i,j{+}1)) \mid 1 \leq i \leq n \wedge 1 \leq j \leq n{-}1\}\,.
\end{array}
$$

Example 3: Python API

$$
\begin{array}{ll}
(1) & \bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right) \\
(2) & \bigwedge_{s=1}^{n} \bigwedge_{f_1=1}^{k-1} \bigwedge_{f_2=f_1+1}^{k} \left(\neg x_{s,f_1} \vee \neg x_{s,f_2} \right) \\
(3) & \forall (s_1, s_2) \in E.\ \bigwedge_{f=1}^{k} \left(\neg x_{s_1,f} \vee \neg x_{s_2,f} \right)
\end{array}
$$

Assume that n^2 ($n \in \mathbb{N}_{>0}$) stations are arranged in a grid with the coordinates (i,j), $1 \leq i,j \leq n$, and

$$
\begin{array}{rcl}
E & = & \{((i,j),(i{+}1,j)) \,|\, 1 \leq i \leq n{-}1 \wedge 1 \leq j \leq n\} \cup \\
 & & \{((i,j),(i,j{+}1)) \,|\, 1 \leq i \leq n \wedge 1 \leq j \leq n{-}1\}\,.
\end{array}
$$

Write a Python program that uses for an input n the **Python API** of z3 to find the minimal k necessary for a solution.

Example 3: Python API

```
from z3 import *
import argparse
import sys
try:
     parser = argparse.ArgumentParser()
     parser.add_argument("n", help="number of stations", type=int)
     args = parser.parse_args()
     n = args.n
except:
    e = sys.exc_info()[0]
    print(e)

names = []
for i in range(n):
    names_i = []
    for j in range(n):
        name = "a_" + str(i+1) + "_" + str(j+1);
        names_i.append(Bool(name))
    names.append(names_i)
s = Solver()
for k in range(n):
    s.push()
    for i in range(n):
        params = []
        for j in range(k+1):
            params.append(names[i][j])
        s.add(Or(params))
    print(s)
    print(s.check())
    s.pop()
```

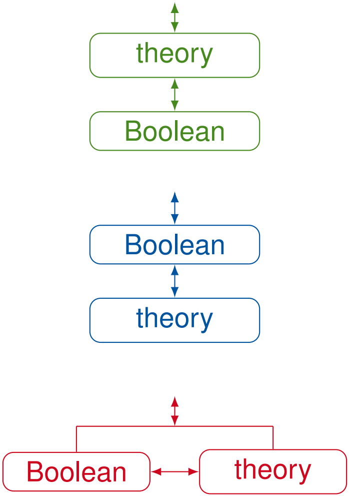
Contents

Three SMT solving approaches

Eager SMT solving

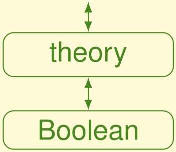
Lazy SMT solving

Model-constructing
satisfiability calculus
(MCSAT)

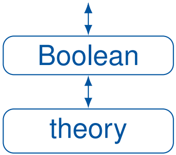


Three SMT solving approaches

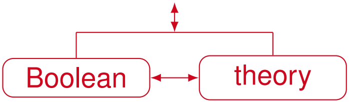
Eager SMT solving



Lazy SMT solving



Model-constructing satisfiability calculus (MCSAT)



Eager example [Bryant and Velev, 2000]

$$\varphi^E \;=\; x_1 = x_2 \;\wedge\; x_2 = x_3 \;\wedge\; x_1 \neq x_3$$

Eager example [Bryant and Velev, 2000]

$$\varphi^E \;=\; x_1 = x_2 \;\wedge\; x_2 = x_3 \;\wedge\; x_1 \neq x_3$$

$$\varphi^{prop} \;:=$$

φ^E *is satisfiable* *iff* φ^{prop} *is satisfiable*

Eager example [Bryant and Velev, 2000]

$$\varphi^E = x_1 = x_2 \wedge x_2 = x_3 \wedge x_1 \neq x_3$$

$$\varphi^{prop} := \underbrace{e_1 \wedge e_2 \wedge \neg e_3}_{\textit{Boolean abstraction}} \wedge$$

φ^E *is satisfiable* *iff* φ^{prop} *is satisfiable*

Eager example [Bryant and Velev, 2000]

$$\varphi^E \;=\; x_1 = x_2 \;\wedge\; x_2 = x_3 \;\wedge\; x_1 \neq x_3$$

$$\varphi^{prop} \;:=\; \underbrace{e_1 \;\wedge\; e_2 \;\wedge\; \neg e_3}_{\textit{Boolean abstraction}} \;\wedge\; \underbrace{((e_1 \wedge e_2) \rightarrow e_3)}_{\textit{transitivity constraint}}$$

φ^E *is satisfiable* *iff* φ^{prop} *is satisfiable*

Eager example [Bryant and Velev, 2000]

$$\varphi^E = x_1 = x_2 \wedge x_2 = x_3 \wedge x_1 \neq x_3$$

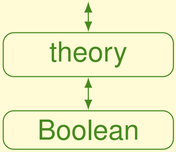
$$\varphi^{prop} := \underbrace{e_1 \wedge e_2 \wedge \neg e_3}_{\text{Boolean abstraction}} \wedge \underbrace{((e_1 \wedge e_2) \rightarrow e_3)}_{\text{transitivity constraint}}$$

φ^E *is satisfiable* *iff* φ^{prop} *is satisfiable*

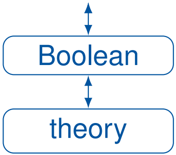
Similar approaches are available for uninterpreted functions, bit-vector arithmetic ("bit-blasting"), floating-point arithmetic and others.

Three SMT solving approaches

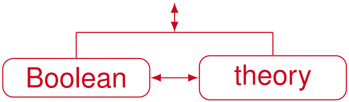
Eager SMT solving



Lazy SMT solving

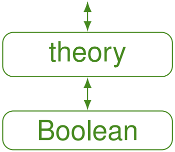


Model-constructing satisfiability calculus (MCSAT)

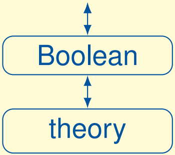


Three SMT solving approaches

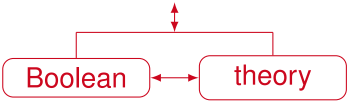
Eager SMT solving



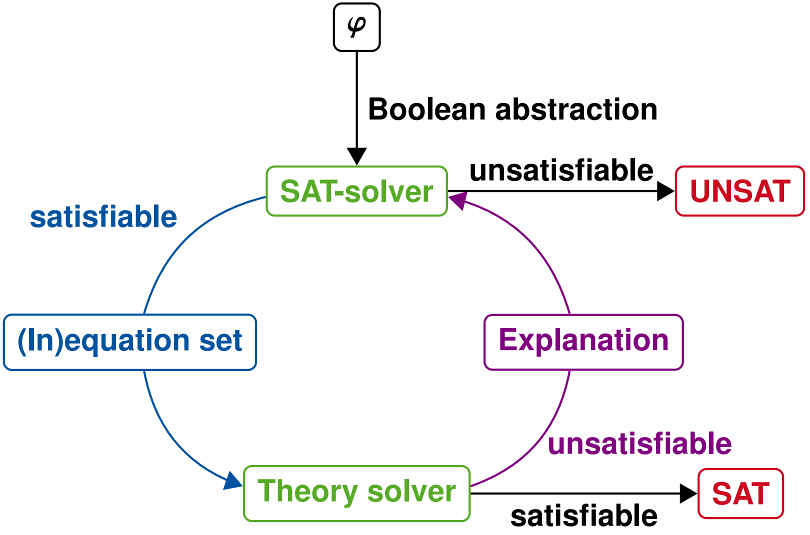
Lazy SMT solving



Model-constructing satisfiability calculus (MCSAT)



Full lazy SMT solving



Boolean abstraction

$$(\underbrace{p_1 = 0}_{a_1} \vee \underbrace{p_2 = 0}_{a_2} \vee \underbrace{p_3 = 0}_{a_3}) \wedge \underbrace{p_1 + p_2 + p_3 \geq 100}_{a_4} \wedge$$

$$(\underbrace{p_1 \geq 5}_{a_5} \vee \underbrace{p_2 \geq 5}_{a_6}) \wedge \underbrace{p_3 \geq 10}_{a_7} \wedge \underbrace{p_1 + 2p_2 + 5p_3 \leq 180}_{a_8} \wedge$$

$$\underbrace{3p_1 + 2p_2 + p_3 \leq 300}_{a_9}$$

Boolean abstraction

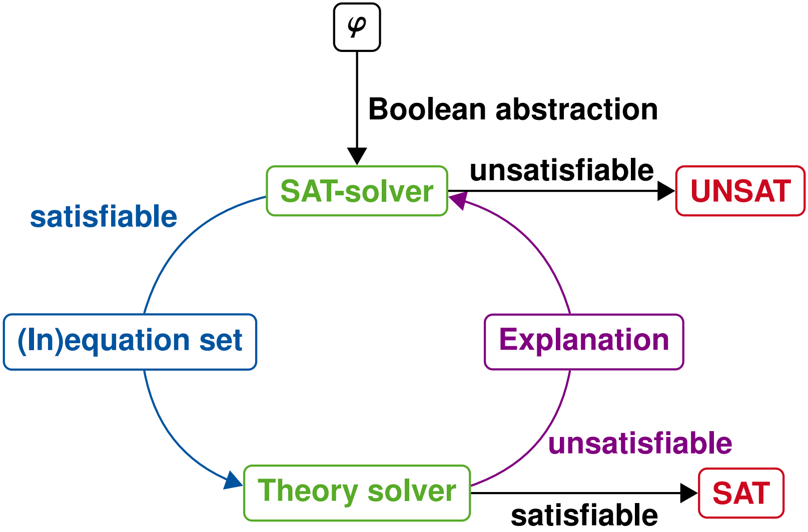
$$(\underbrace{p_1 = 0}_{a_1} \vee \underbrace{p_2 = 0}_{a_2} \vee \underbrace{p_3 = 0}_{a_3}) \wedge \underbrace{p_1 + p_2 + p_3 \geq 100}_{a_4} \wedge$$

$$(\underbrace{p_1 \geq 5}_{a_5} \vee \underbrace{p_2 \geq 5}_{a_6}) \wedge \underbrace{p_3 \geq 10}_{a_7} \wedge \underbrace{p_1 + 2p_2 + 5p_3 \leq 180}_{a_8} \wedge$$

$$\underbrace{3p_1 + 2p_2 + p_3 \leq 300}_{a_9}$$

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Full lazy SMT solving



SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order:	$a_1, \ldots, a_9$
Assignment to decision variables:	false

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order:	$a_1, \ldots, a_9$
Assignment to decision variables:	false

$DL0$:

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order:	$a_1, \ldots, a_9$
Assignment to decision variables:	false

$DL0 : a_4 : 1$

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order:	$a_1, \ldots, a_9$
Assignment to decision variables:	false

$DL0 : a_4 : 1, a_7 : 1$

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order:	$a_1, \ldots, a_9$
Assignment to decision variables:	false

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1$

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order:	$a_1, \ldots, a_9$
Assignment to decision variables:	false

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$

SAT solving

$$(a_1 \lor a_2 \lor a_3) \land a_4 \land (a_5 \lor a_6) \land a_7 \land a_8 \land a_9$$

Assume a fixed variable order: $a_1, \ldots, a_9$
Assignment to decision variables: false

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$
$DL1 :$

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order:	$a_1, \ldots, a_9$
Assignment to decision variables:	false

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$
$DL1 : a_1 : 0$

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order:	$a_1, \ldots, a_9$
Assignment to decision variables:	false

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$
$DL1 : a_1 : 0$
$DL2 :$

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order:	$a_1, \ldots, a_9$
Assignment to decision variables:	false

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$
$DL1 : a_1 : 0$
$DL2 : a_2 : 0$

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order:	$a_1, \ldots, a_9$
Assignment to decision variables:	false

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$
$DL1 : a_1 : 0$
$DL2 : a_2 : 0, a_3 : 1$

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order:	$a_1, \ldots, a_9$
Assignment to decision variables:	false

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$
$DL1 : a_1 : 0$
$DL2 : a_2 : 0, a_3 : 1$
$DL3 :$

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order:	$a_1, \ldots, a_9$
Assignment to decision variables:	false

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$
$DL1 : a_1 : 0$
$DL2 : a_2 : 0, a_3 : 1$
$DL3 : a_5 : 0$

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order:	$a_1, \ldots, a_9$
Assignment to decision variables:	false

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$
$DL1 : a_1 : 0$
$DL2 : a_2 : 0, a_3 : 1$
$DL3 : a_5 : 0, a_6 : 1$

SAT solving

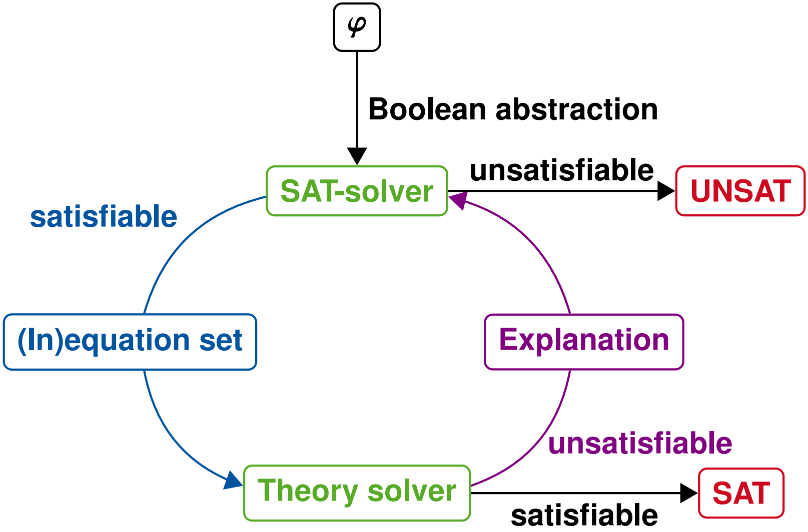
$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order:	$a_1, \ldots, a_9$
Assignment to decision variables:	false

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$
$DL1 : a_1 : 0$
$DL2 : a_2 : 0, a_3 : 1$
$DL3 : a_5 : 0, a_6 : 1$

Solution found for the Boolean abstraction.

Full lazy SMT solving



Theory solving

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$ $\quad DL1 : a_1 : 0$
$DL2 : a_2 : 0, a_3 : 1$ $\quad DL3 : a_5 : 0, a_6 : 1$

Theory solving

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$ $DL1 : a_1 : 0$
$DL2 : a_2 : 0, a_3 : 1$ $DL3 : a_5 : 0, a_6 : 1$

True theory constraints: $a_4, a_7, a_8, a_9, a_3, a_6$

Theory solving

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$ $\quad DL1 : a_1 : 0$
$DL2 : a_2 : 0, a_3 : 1$ $\quad DL3 : a_5 : 0, a_6 : 1$

True theory constraints: $a_4, a_7, a_8, a_9, a_3, a_6$

$$(\underbrace{p_1 = 0}_{a_1} \vee \underbrace{p_2 = 0}_{a_2} \vee \underbrace{p_3 = 0}_{a_3}) \wedge \underbrace{p_1 + p_2 + p_3 \geq 100}_{a_4} \wedge$$

$$(\underbrace{p_1 \geq 5}_{a_5} \vee \underbrace{p_2 \geq 5}_{a_6}) \wedge \underbrace{p_3 \geq 10}_{a_7} \wedge \underbrace{p_1 + 2p_2 + 5p_3 \leq 180}_{a_8} \wedge$$

$$\underbrace{3p_1 + 2p_2 + p_3 \leq 300}_{a_9}$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$ $DL1 : a_1 : 0$
$DL2 : a_2 : 0, a_3 : 1$ $DL3 : a_5 : 0, a_6 : 1$

True theory constraints: $a_4, a_7, a_8, a_9, a_3, a_6$

$$(\underbrace{p_1 = 0}_{a_1} \vee \underbrace{p_2 = 0}_{a_2} \vee \underbrace{p_3 = 0}_{a_3}) \wedge \underbrace{p_1 + p_2 + p_3 \geq 100}_{a_4} \wedge$$
$$(\underbrace{p_1 \geq 5}_{a_5} \vee \underbrace{p_2 \geq 5}_{a_6}) \wedge \underbrace{p_3 \geq 10}_{a_7} \wedge \underbrace{p_1 + 2p_2 + 5p_3 \leq 180}_{a_8} \wedge$$
$$\underbrace{3p_1 + 2p_2 + p_3 \leq 300}_{a_9}$$

Encoding:

$a_4 : p_1 + p_2 + p_3 \geq 100$ $a_7 : p_3 \geq 10$ $a_8 : p_1 + 2p_2 + 5p_3 \leq 180$
$a_9 : 3p_1 + 2p_2 + p_3 \leq 300$ $a_3 : p_3 = 0$ $a_6 : p_2 \geq 5$

Theory solving

Is the conjunction of the following constraints satisfiable?

$a_4 : p_1 + p_2 + p_3 \geq 100$

$a_7 : p_3 \geq 10$

$a_8 : p_1 + 2p_2 + 5p_3 \leq 180$

$a_9 : 3p_1 + 2p_2 + p_3 \leq 300$

$a_3 : p_3 = 0$

$a_6 : p_2 \geq 5$

Is the conjunction of the following constraints satisfiable?

$a_4 : p_1 + p_2 + p_3 \geq 100$

$a_7 : p_3 \geq 10$

$a_8 : p_1 + 2p_2 + 5p_3 \leq 180$

$a_9 : 3p_1 + 2p_2 + p_3 \leq 300$

$a_3 : p_3 = 0$

$a_6 : p_2 \geq 5$

No.

Is the conjunction of the following constraints satisfiable?

$a_4 : p_1 + p_2 + p_3 \geq 100$

$a_7 : p_3 \geq 10$

$a_8 : p_1 + 2p_2 + 5p_3 \leq 180$

$a_9 : 3p_1 + 2p_2 + p_3 \leq 300$

$a_3 : p_3 = 0$

$a_6 : p_2 \geq 5$

No.

Reason:

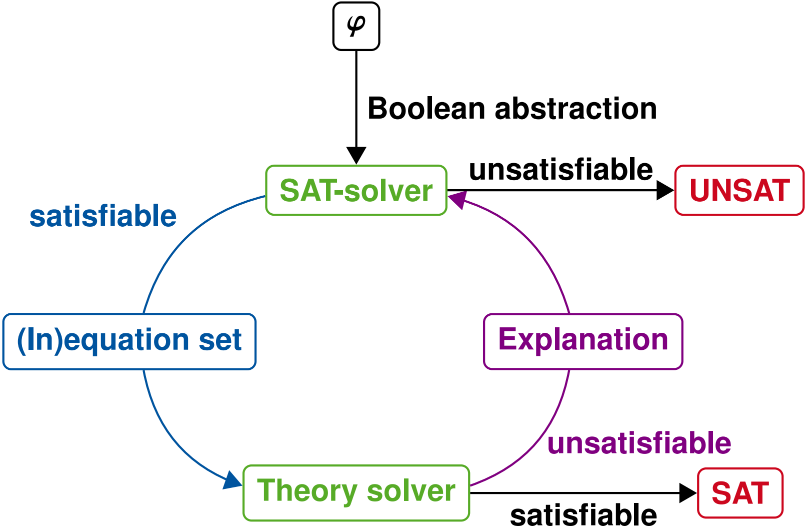
Theory solving

Is the conjunction of the following constraints satisfiable?

$a_4 : p_1 + p_2 + p_3 \geq 100$
$a_7 : p_3 \geq 10$
$a_8 : p_1 + 2p_2 + 5p_3 \leq 180$
$a_9 : 3p_1 + 2p_2 + p_3 \leq 300$
$a_3 : p_3 = 0$
$a_6 : p_2 \geq 5$

No.
Reason: $\underbrace{p_3 = 0}_{a_3} \wedge \underbrace{p_3 \geq 10}_{a_7}$ are conflicting.

Full lazy SMT solving



SAT solving

Add clause $(\neg a_3 \vee \neg a_7)$.

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$
$DL1 : a_1 : 0$
$DL2 : a_2 : 0, a_3 : 1$
$DL3 : a_5 : 0, a_6 : 1$

Add clause $(\neg a_3 \vee \neg a_7)$.

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$
$DL1 : a_1 : 0$
$DL2 : a_2 : 0, a_3 : 1$
$DL3 : a_5 : 0, a_6 : 1$

Conflict resolution is simple, since the new clause is already an asserting one.

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$
$DL1 :$

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$
$DL1 : a_1 : 0$

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$
$DL1 : a_1 : 0, a_2 : 1$

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$
$DL1 : a_1 : 0, a_2 : 1$
$DL2 :$

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$
$DL1 : a_1 : 0, a_2 : 1$
$DL2 : a_5 : 0$

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7)$$

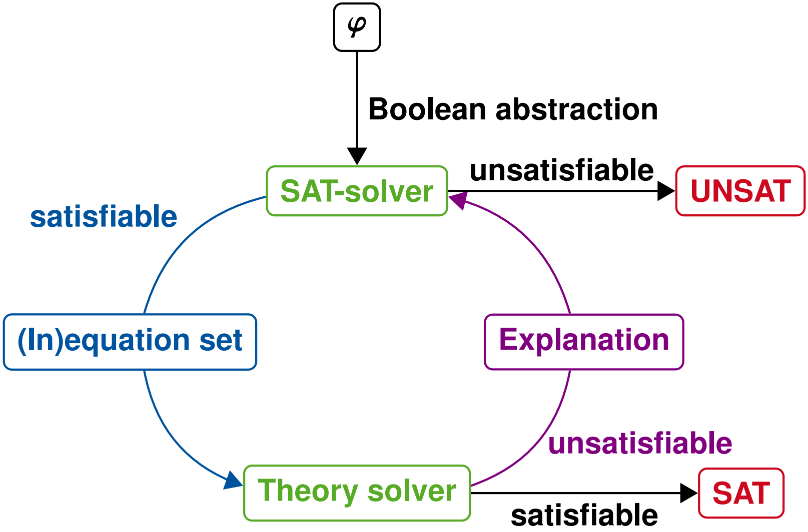
$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$
$DL1 : a_1 : 0, a_2 : 1$
$DL2 : a_5 : 0, a_6 : 1$

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$
$DL1 : a_1 : 0, a_2 : 1$
$DL2 : a_5 : 0, a_6 : 1$

Solution found for the Boolean abstraction.

Full lazy SMT solving



Theory solving

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0 \quad DL1 : a_1 : 0, a_2 : 1$
$DL2 : a_5 : 0, a_6 : 1$

Theory solving

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0 \quad DL1 : a_1 : 0, a_2 : 1$
$DL2 : a_5 : 0, a_6 : 1$

True theory constraints: $a_4, a_7, a_8, a_9, a_2, a_6$

Theory solving

$DL0: a_4:1, a_7:1, a_8:1, a_9:1, a_3:0 \quad DL1: a_1:0, a_2:1$
$DL2: a_5:0, a_6:1$

True theory constraints: $a_4, a_7, a_8, a_9, a_2, a_6$

$$(\underbrace{p_1 = 0}_{a_1} \vee \underbrace{p_2 = 0}_{a_2} \vee \underbrace{p_3 = 0}_{a_3}) \wedge \underbrace{p_1 + p_2 + p_3 \geq 100}_{a_4} \wedge$$
$$(\underbrace{p_1 \geq 5}_{a_5} \vee \underbrace{p_2 \geq 5}_{a_6}) \wedge \underbrace{p_3 \geq 10}_{a_7} \wedge \underbrace{p_1 + 2p_2 + 5p_3 \leq 180}_{a_8} \wedge$$
$$\underbrace{3p_1 + 2p_2 + p_3 \leq 300}_{a_9} \wedge (\neg a_3 \vee \neg a_7)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0 \quad DL1 : a_1 : 0, a_2 : 1$
$DL2 : a_5 : 0, a_6 : 1$

True theory constraints: $a_4, a_7, a_8, a_9, a_2, a_6$

$$(\underbrace{p_1 = 0}_{a_1} \vee \underbrace{p_2 = 0}_{a_2} \vee \underbrace{p_3 = 0}_{a_3}) \wedge \underbrace{p_1 + p_2 + p_3 \geq 100}_{a_4} \wedge$$
$$(\underbrace{p_1 \geq 5}_{a_5} \vee \underbrace{p_2 \geq 5}_{a_6}) \wedge \underbrace{p_3 \geq 10}_{a_7} \wedge \underbrace{p_1 + 2p_2 + 5p_3 \leq 180}_{a_8} \wedge$$
$$\underbrace{3p_1 + 2p_2 + p_3 \leq 300}_{a_9} \wedge (\neg a_3 \vee \neg a_7)$$

Encoding:

$a_4 : p_1 + p_2 + p_3 \geq 100 \quad a_7 : p_3 \geq 10 \quad a_8 : p_1 + 2p_2 + 5p_3 \leq 180$
$a_9 : 3p_1 + 2p_2 + p_3 \leq 300 \quad a_2 : p_2 = 0 \quad a_6 : p_2 \geq 5$

Theory solving

Is the conjunction of the following constraints satisfiable?

a_4 : $p_1 + p_2 + p_3 \geq 100$

a_7 : $p_3 \geq 10$

a_8 : $p_1 + 2p_2 + 5p_3 \leq 180$

a_9 : $3p_1 + 2p_2 + p_3 \leq 300$

a_2 : $p_2 = 0$

a_6 : $p_2 \geq 5$

Theory solving

Is the conjunction of the following constraints satisfiable?

$a_4 : p_1 + p_2 + p_3 \geq 100$

$a_7 : p_3 \geq 10$

$a_8 : p_1 + 2p_2 + 5p_3 \leq 180$

$a_9 : 3p_1 + 2p_2 + p_3 \leq 300$

$a_2 : p_2 = 0$

$a_6 : p_2 \geq 5$

No.

Theory solving

Is the conjunction of the following constraints satisfiable?

$a_4 : p_1 + p_2 + p_3 \geq 100$

$a_7 : p_3 \geq 10$

$a_8 : p_1 + 2p_2 + 5p_3 \leq 180$

$a_9 : 3p_1 + 2p_2 + p_3 \leq 300$

$a_2 : p_2 = 0$

$a_6 : p_2 \geq 5$

No.

Reason:

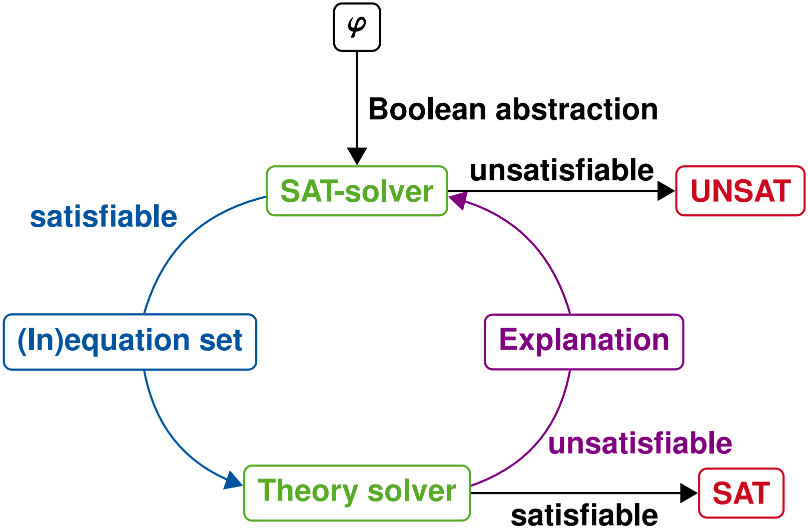
Theory solving

Is the conjunction of the following constraints satisfiable?

$a_4 : p_1 + p_2 + p_3 \geq 100$
$a_7 : p_3 \geq 10$
$a_8 : p_1 + 2p_2 + 5p_3 \leq 180$
$a_9 : 3p_1 + 2p_2 + p_3 \leq 300$
$a_2 : p_2 = 0$
$a_6 : p_2 \geq 5$

No.
Reason: $\underbrace{p_2 = 0}_{a_2} \wedge \underbrace{p_2 \geq 5}_{a_6}$ are conflicting.

Full lazy SMT solving



Add clause $(\neg a_2 \vee \neg a_6)$.

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7) \wedge (\neg a_2 \vee \neg a_6)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$
$DL1 : a_1 : 0, a_2 : 1$
$DL2 : a_5 : 0, a_6 : 1$

Add clause $(\neg a_2 \vee \neg a_6)$.

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7) \wedge (\neg a_2 \vee \neg a_6)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$
$DL1 : a_1 : 0, a_2 : 1$
$DL2 : a_5 : 0, a_6 : 1$

Conflict resolution is simple, since the new clause is already an asserting one.

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7) \wedge$$
$$(\neg a_2 \vee \neg a_6)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$
$DL1 : a_1 : 0, a_2 : 1$

SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7) \wedge$$
$$(\neg a_2 \vee \neg a_6)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$
$DL1 : a_1 : 0, a_2 : 1, a_6 : 0$

SAT solving

$$(a_1 \lor a_2 \lor a_3) \land a_4 \land (a_5 \lor a_6) \land a_7 \land a_8 \land a_9 \land (\neg a_3 \lor \neg a_7) \land$$
$$(\neg a_2 \lor \neg a_6)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$
$DL1 : a_1 : 0, a_2 : 1, a_6 : 0, a_5 : 1$

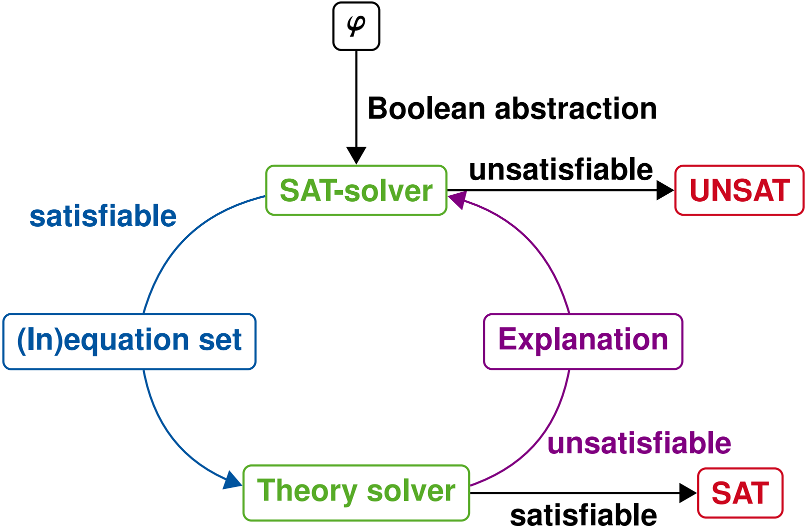
SAT solving

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7) \wedge$$
$$(\neg a_2 \vee \neg a_6)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$
$DL1 : a_1 : 0, a_2 : 1, a_6 : 0, a_5 : 1$

Solution found for the Boolean abstraction.

Full lazy SMT solving



Theory solving

$DL0: a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0 \quad DL1: a_1 : 0, a_2 : 1, a_6 : 0, a_5 : 1$

Theory solving

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0 \quad DL1 : a_1 : 0, a_2 : 1, a_6 : 0, a_5 : 1$

True theory constraints: $a_4, a_7, a_8, a_9, a_2, a_5$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0 \quad DL1 : a_1 : 0, a_2 : 1, a_6 : 0, a_5 : 1$

True theory constraints: $a_4, a_7, a_8, a_9, a_2, a_5$

$$(\underbrace{p_1 = 0}_{a_1} \vee \underbrace{p_2 = 0}_{a_2} \vee \underbrace{p_3 = 0}_{a_3}) \wedge \underbrace{p_1 + p_2 + p_3 \geq 100}_{a_4} \wedge$$

$$(\underbrace{p_1 \geq 5}_{a_5} \vee \underbrace{p_2 \geq 5}_{a_6}) \wedge \underbrace{p_3 \geq 10}_{a_7} \wedge \underbrace{p_1 + 2p_2 + 5p_3 \leq 180}_{a_8} \wedge$$

$$\underbrace{3p_1 + 2p_2 + p_3 \leq 300}_{a_9} \wedge (\neg a_3 \vee \neg a_7) \wedge (\neg a_2 \vee \neg a_6)$$

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0 \quad DL1 : a_1 : 0, a_2 : 1, a_6 : 0, a_5 : 1$

True theory constraints: $a_4, a_7, a_8, a_9, a_2, a_5$

$$(\underbrace{p_1 = 0}_{a_1} \vee \underbrace{p_2 = 0}_{a_2} \vee \underbrace{p_3 = 0}_{a_3}) \wedge \underbrace{p_1 + p_2 + p_3 \geq 100}_{a_4} \wedge$$

$$(\underbrace{p_1 \geq 5}_{a_5} \vee \underbrace{p_2 \geq 5}_{a_6}) \wedge \underbrace{p_3 \geq 10}_{a_7} \wedge \underbrace{p_1 + 2p_2 + 5p_3 \leq 180}_{a_8} \wedge$$

$$\underbrace{3p_1 + 2p_2 + p_3 \leq 300}_{a_9} \wedge (\neg a_3 \vee \neg a_7) \wedge (\neg a_2 \vee \neg a_6)$$

Encoding:

$a_4 : p_1 + p_2 + p_3 \geq 100 \quad a_7 : p_3 \geq 10 \quad a_8 : p_1 + 2p_2 + 5p_3 \leq 180$

$a_9 : 3p_1 + 2p_2 + p_3 \leq 300 \quad a_2 : p_2 = 0 \quad a_5 : p_1 \geq 5$

Theory solving

Is the conjunction of the following constraints satisfiable?

$a_4 : p_1 + p_2 + p_3 \geq 100$

$a_7 : p_3 \geq 10$

$a_8 : p_1 + 2p_2 + 5p_3 \leq 180$

$a_9 : 3p_1 + 2p_2 + p_3 \leq 300$

$a_2 : p_2 = 0$

$a_5 : p_1 \geq 5$

Theory solving

Is the conjunction of the following constraints satisfiable?

$a_4 : p_1 + p_2 + p_3 \geq 100$

$a_7 : p_3 \geq 10$

$a_8 : p_1 + 2p_2 + 5p_3 \leq 180$

$a_9 : 3p_1 + 2p_2 + p_3 \leq 300$

$a_2 : p_2 = 0$

$a_5 : p_1 \geq 5$

Yes.

Is the conjunction of the following constraints satisfiable?

a_4 : $p_1 + p_2 + p_3 \geq 100$
a_7 : $p_3 \geq 10$
a_8 : $p_1 + 2p_2 + 5p_3 \leq 180$
a_9 : $3p_1 + 2p_2 + p_3 \leq 300$
a_2 : $p_2 = 0$
a_5 : $p_1 \geq 5$

Yes. E.g.,

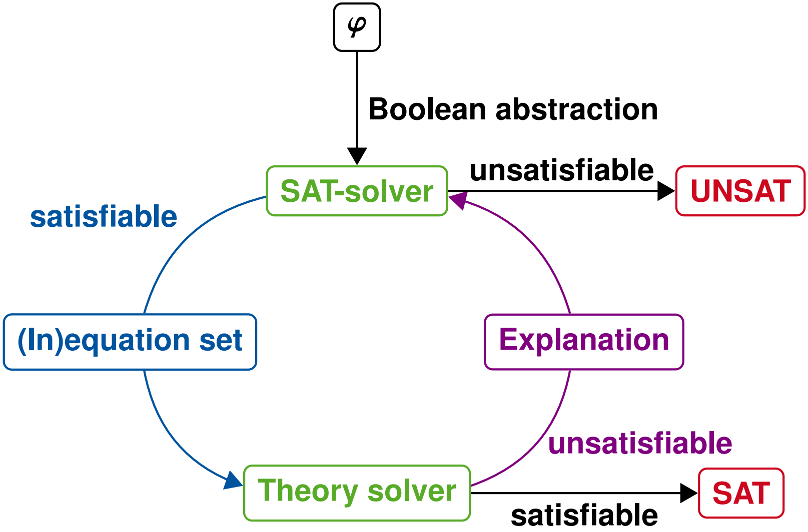
Theory solving

Is the conjunction of the following constraints satisfiable?

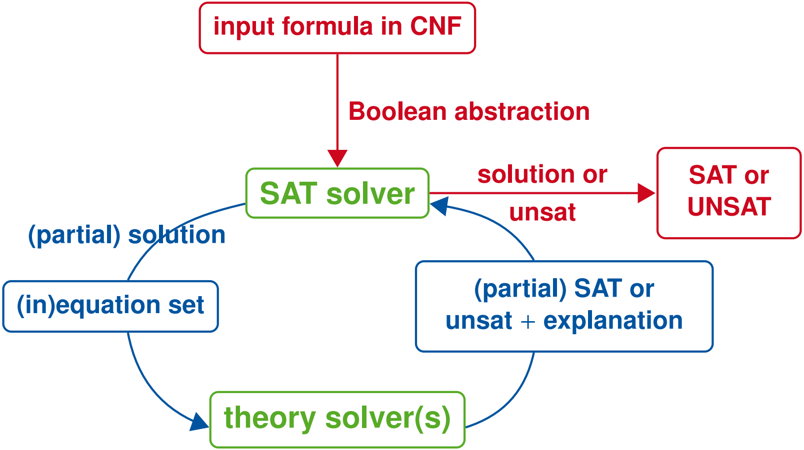
a_4 : $p_1 + p_2 + p_3 \geq 100$
a_7 : $p_3 \geq 10$
a_8 : $p_1 + 2p_2 + 5p_3 \leq 180$
a_9 : $3p_1 + 2p_2 + p_3 \leq 300$
a_2 : $p_2 = 0$
a_5 : $p_1 \geq 5$

Yes. E.g., $p_1 = 90$, $p_2 = 0$, $p_3 = 10$ is a solution.

Full lazy SMT solving



Less lazy SMT solving

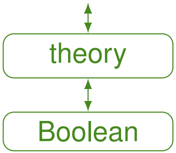


Requirements on the theory solver

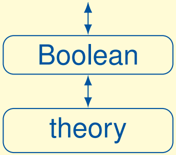
1. Incrementality: In less lazy solving we extend the set of constraints. The solver should make use of the previous satisfiability check for the check of the extended set.
2. (Preferably minimal) infeasible subsets: Compute a reason for unsatisfaction
3. Backtracking: The theory solver should be able to remove constraints in inverse chronological order.

Three SMT solving approaches

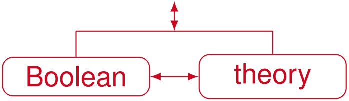
Eager SMT solving



Lazy SMT solving

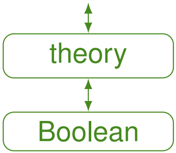


Model-constructing satisfiability calculus (MCSAT)

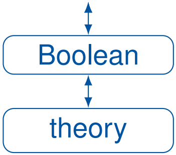


Three SMT solving approaches

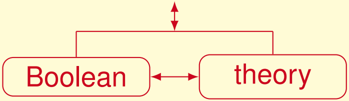
Eager SMT solving



Lazy SMT solving



Model-constructing satisfiability calculus (MCSAT)



The DPLL+CDCL idea [Davis et al., ’60/61] [Marques-Silva et al., ’96]

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

The DPLL+CDCL idea [Davis et al., '60/61] [Marques-Silva et al., '96]

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

$$(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$$

The DPLL+CDCL idea [Davis et al., '60/61] [Marques-Silva et al., '96]

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

$$(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$$

$\mathbb{B}$-propagate -

The DPLL+CDCL idea [Davis et al., ’60/61] [Marques-Silva et al., ’96]

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

$$(a \lor b \lor c) \land (a \lor b \lor \neg c)$$

$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	$a = $ *false*

The DPLL+CDCL idea [Davis et al., '60/61] [Marques-Silva et al., '96]

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

$$(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$$

$\mathbb{B}$-propagate -

$\mathbb{B}$-decision $a =$ *false*

$\mathbb{B}$-propagate -

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

$$(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$$

$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	a = *false*
$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	b = *false*

The DPLL+CDCL idea [Davis et al., '60/61] [Marques-Silva et al., '96]

Exploration:	$\mathbb{B}$-decision
Look-ahead:	$\mathbb{B}$-propagation
Proof system:	$\mathbb{B}$-conflict resolution

$$(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$$

$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	a = *false*
$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	b = *false*
$\mathbb{B}$-propagate	c = *true* $\lightning$

The DPLL+CDCL idea [Davis et al., ’60/61] [Marques-Silva et al., ’96]

Exploration: $\mathbb{B}$-decision

Look-ahead: $\mathbb{B}$-propagation

Proof system: $\mathbb{B}$-conflict resolution

$$(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$$

$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	a = *false*
$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	b = *false*
$\mathbb{B}$-propagate	c = *true* ⚡
$\mathbb{B}$-conflict resolution	$(a \vee b)$

The MCSAT idea [de Moura, Jovanović, VMCAI'13]

Exploration:	$\mathbb{B}$-decision	$\mathbb{T}$-decision
Look-ahead:	$\mathbb{B}$-propagation	$\mathbb{T}$-propagation
Proof system:	$\mathbb{B}$-conflict resolution	$\mathbb{T}$-conflict resolution

$$(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$$

$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	a = *false*
$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	b = *false*
$\mathbb{B}$-propagate	c = *true* ⚡
$\mathbb{B}$-conflict resolution	$(a \vee b)$

The MCSAT idea [de Moura, Jovanović, VMCAI'13]

Exploration:	$\mathbb{B}$-decision	$\mathbb{T}$-decision
Look-ahead:	$\mathbb{B}$-propagation	$\mathbb{T}$-propagation
Proof system:	$\mathbb{B}$-conflict resolution	$\mathbb{T}$-conflict resolution

$(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$ $\qquad \ldots\ x \cdot y^2 < 0\ \ldots$

$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	a = *false*
$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	b = *false*
$\mathbb{B}$-propagate	c = *true* ↯
$\mathbb{B}$-conflict resolution	$(a \vee b)$

The MCSAT idea [de Moura, Jovanović, VMCAI'13]

Exploration:	$\mathbb{B}$-decision	$\mathbb{T}$-decision
Look-ahead:	$\mathbb{B}$-propagation	$\mathbb{T}$-propagation
Proof system:	$\mathbb{B}$-conflict resolution	$\mathbb{T}$-conflict resolution

$$(a \vee b \vee c) \wedge (a \vee b \vee \neg c) \qquad \ldots\ x \cdot y^2 < 0\ \ldots$$

$\mathbb{B}$-propagate	-	$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	a = *false*		
$\mathbb{B}$-propagate	-		
$\mathbb{B}$-decision	b = *false*		
$\mathbb{B}$-propagate	c = *true* ⚡		
$\mathbb{B}$-conflict resolution	$(a \vee b)$		

The MCSAT idea [de Moura, Jovanović, VMCAI'13]

Exploration:	$\mathbb{B}$-decision	$\mathbb{T}$-decision
Look-ahead:	$\mathbb{B}$-propagation	$\mathbb{T}$-propagation
Proof system:	$\mathbb{B}$-conflict resolution	$\mathbb{T}$-conflict resolution

$(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$ $\quad \ldots\ x \cdot y^2 < 0\ \ldots$

$\mathbb{B}$-propagate	-	$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	a = *false*	$\mathbb{B}$-decision	$x \cdot y^2 < 0$
$\mathbb{B}$-propagate	-		
$\mathbb{B}$-decision	b = *false*		
$\mathbb{B}$-propagate	c = *true* ↯		
$\mathbb{B}$-conflict resolution	$(a \vee b)$		

The MCSAT idea [de Moura, Jovanović, VMCAI'13]

Exploration:	$\mathbb{B}$-decision	$\mathbb{T}$-decision
Look-ahead:	$\mathbb{B}$-propagation	$\mathbb{T}$-propagation
Proof system:	$\mathbb{B}$-conflict resolution	$\mathbb{T}$-conflict resolution

$(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$ $\quad\ldots\ x \cdot y^2 < 0\ \ldots$

$\mathbb{B}$-propagate	-	$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	a = *false*	$\mathbb{B}$-decision	$x \cdot y^2 < 0$
$\mathbb{B}$-propagate	-	$\mathbb{T}$-propagate	$x \in (-\infty, \infty)$
$\mathbb{B}$-decision	b = *false*		
$\mathbb{B}$-propagate	c = *true* ↯		
$\mathbb{B}$-conflict resolution	$(a \vee b)$		

The MCSAT idea [de Moura, Jovanović, VMCAI'13]

Exploration:	$\mathbb{B}$-decision	$\mathbb{T}$-decision
Look-ahead:	$\mathbb{B}$-propagation	$\mathbb{T}$-propagation
Proof system:	$\mathbb{B}$-conflict resolution	$\mathbb{T}$-conflict resolution

$$(a \vee b \vee c) \wedge (a \vee b \vee \neg c) \qquad \ldots\ x \cdot y^2 < 0\ \ldots$$

$\mathbb{B}$-propagate	-	$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	a = *false*	$\mathbb{B}$-decision	$x \cdot y^2 < 0$
$\mathbb{B}$-propagate	-	$\mathbb{T}$-propagate	$x \in (-\infty, \infty)$
$\mathbb{B}$-decision	b = *false*	$\mathbb{T}$-decision	$x = 1$
$\mathbb{B}$-propagate	c = *true* ↯		
$\mathbb{B}$-conflict resolution	$(a \vee b)$		

The MCSAT idea [de Moura, Jovanović, VMCAI'13]

Exploration:	$\mathbb{B}$-decision	$\mathbb{T}$-decision
Look-ahead:	$\mathbb{B}$-propagation	$\mathbb{T}$-propagation
Proof system:	$\mathbb{B}$-conflict resolution	$\mathbb{T}$-conflict resolution

$(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$ $\quad\quad \ldots\ x \cdot y^2 < 0\ \ldots$

$\mathbb{B}$-propagate	-	$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	a = *false*	$\mathbb{B}$-decision	$x \cdot y^2 < 0$
$\mathbb{B}$-propagate	-	$\mathbb{T}$-propagate	$x \in (-\infty, \infty)$
$\mathbb{B}$-decision	b = *false*	$\mathbb{T}$-decision	$x = 1$
$\mathbb{B}$-propagate	c = *true* $\lightning$	$\mathbb{T}$-propagate	$y \in \emptyset$ $\lightning$
$\mathbb{B}$-conflict resolution	$(a \vee b)$		

The MCSAT idea [de Moura, Jovanović, VMCAI'13]

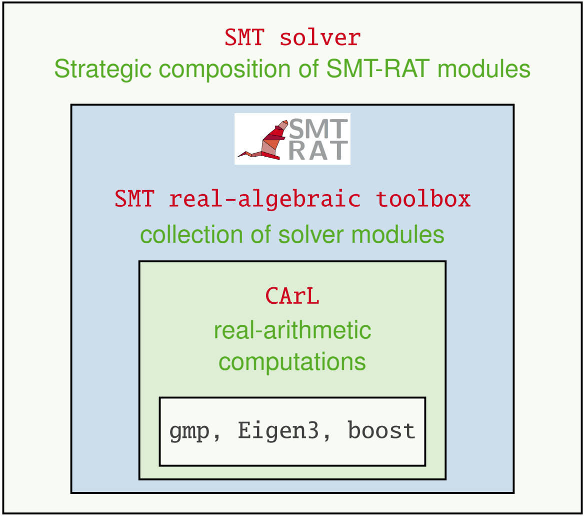
Exploration:	$\mathbb{B}$-decision	$\mathbb{T}$-decision
Look-ahead:	$\mathbb{B}$-propagation	$\mathbb{T}$-propagation
Proof system:	$\mathbb{B}$-conflict resolution	$\mathbb{T}$-conflict resolution

$(a \vee b \vee c) \wedge (a \vee b \vee \neg c)$ $\qquad$ $\ldots\ x \cdot y^2 < 0\ \ldots$

$\mathbb{B}$-propagate	-	$\mathbb{B}$-propagate	-
$\mathbb{B}$-decision	$a = false$	$\mathbb{B}$-decision	$x \cdot y^2 < 0$
$\mathbb{B}$-propagate	-	$\mathbb{T}$-propagate	$x \in (-\infty, \infty)$
$\mathbb{B}$-decision	$b = false$	$\mathbb{T}$-decision	$x = 1$
$\mathbb{B}$-propagate	$c = true$ ⚡	$\mathbb{T}$-propagate	$y \in \emptyset$ ⚡
$\mathbb{B}$-conflict resolution	$(a \vee b)$	$\mathbb{T}$-conflict resolution	$(x \cdot y^2 < 0 \rightarrow x < 0)$

Contents

Our SMT-RAT library [SAT'12, SAT'15]



- MIT licensed source code: `github.com/smtrat/smtrat`
- Documentation: `smtrat.github.io`

Solver modules in SMT-RAT [SAT'12, SAT'15]

CArL library: basic arithmetic datatypes and computations [Sapientia'18, NFM'11, CAI'11]

Basic modules

- SAT solver
- CNF converter
- Preprocessing/simplifying modules

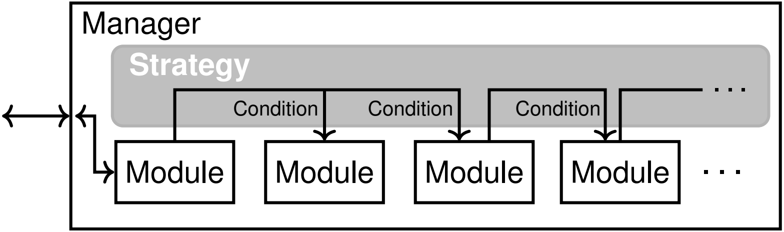
Non-algebraic decision procedures

- Equalities and uninterpreted functions
- Bit-vectors
- Bit-blasting
- Interval constraint propagation
- Pseudo-Boolean formulas

Algebraic decision procedures

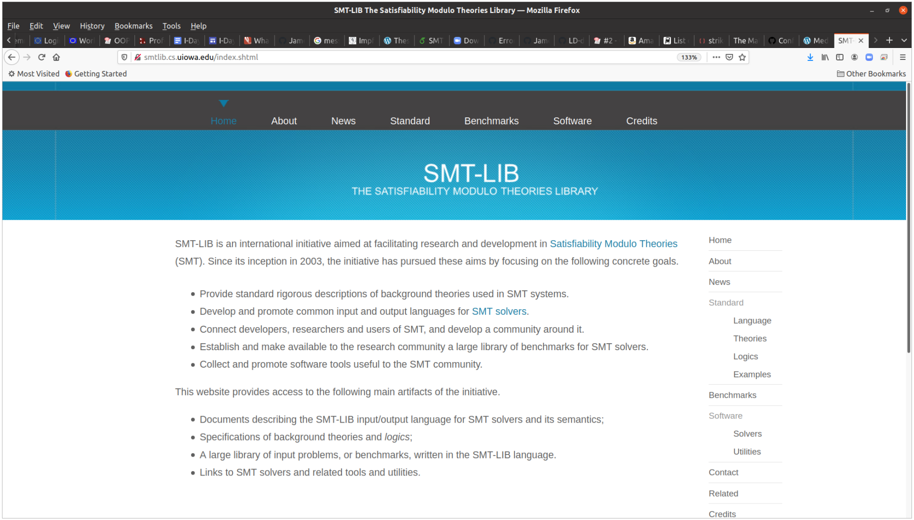
- Gauß+Fourier-Motzkin, FMplex [GandALF'23]
- Gröbner bases [CAI'13]
- MCSAT (FM,VS,CAD) [2xSC^2'19]
- Simplex [ISSAC'21]
- Cylindrical algebraic decomposition [SC^2'21, CADE-24, JSC'19, SC^2'17, 3 PhDs]
- Cylindrical algebraic covering [SMT'23, JLAMP'21, SYNASC'21, PhD Kremer]
- Virtual substitution [FCT'11, SC^2'17, 1 PhD]
- Subtropical satisfiability [NFM'23]
- Generalized branch-and-bound [CASC'16]
- Cube tests
- Linearization

Strategic composition of solver modules in SMT-RAT

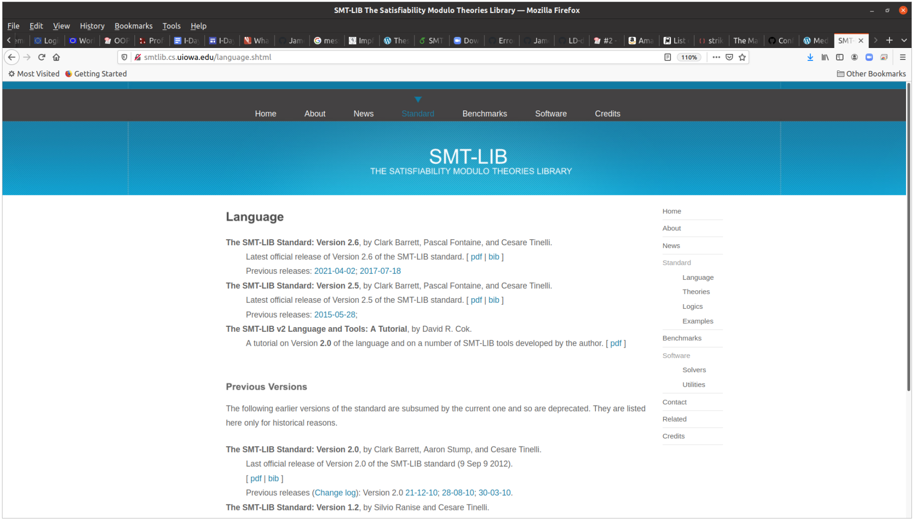


Contents

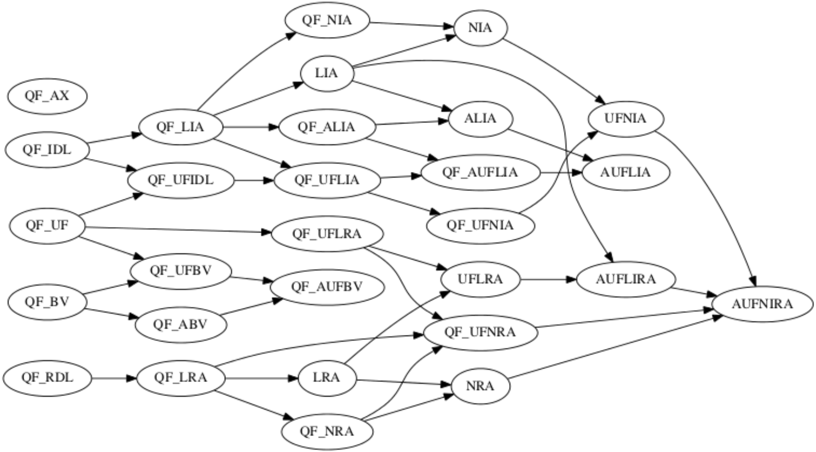
The Satisfiability Modulo Theories Library



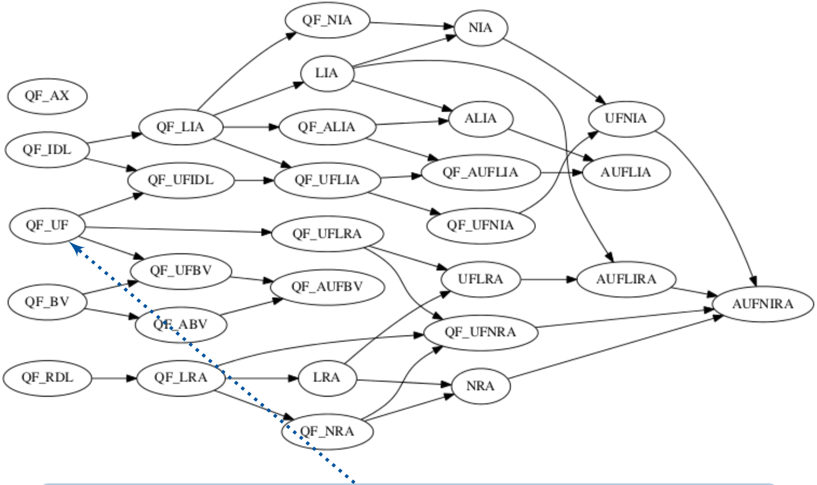
The Satisfiability Modulo Theories Library



SMT-LIB theories



Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

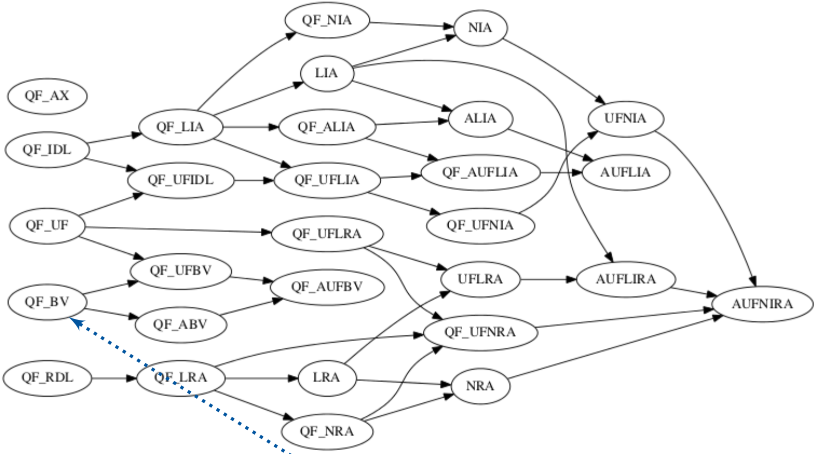


Quantifier-free equality logic with uninterpreted functions

$$(a = c \wedge b = d) \rightarrow f(a, b) = f(c, d)$$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

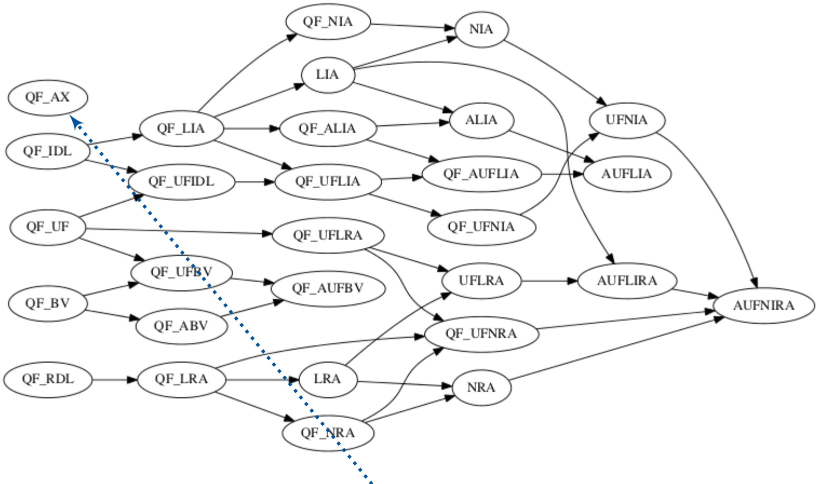


Quantifier-free bit-vector arithmetic

$(a|b) \leq (a\&b)$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

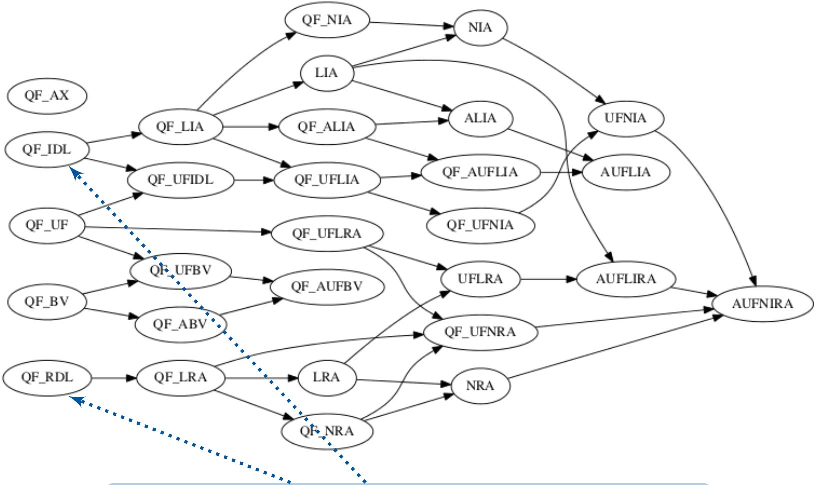


Quantifier-free array theory

$i = j \rightarrow \mathit{read}(\mathit{write}(a, i, v), j) = v$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

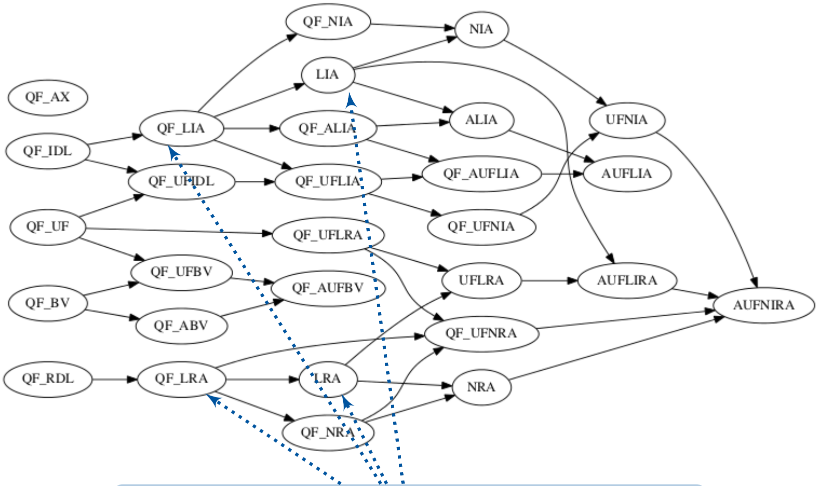
SMT-LIB theories



Quantifier-free integer/rational difference logic

$x - y \sim 0, \sim \in \{<, \leq, =, \geq, >\}$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

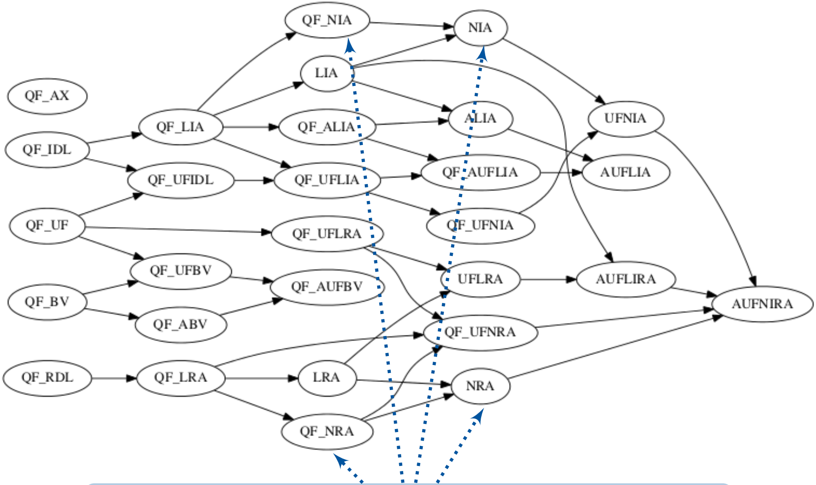


(Quantifier-free) real/integer linear arithmetic

$3x + 7y = 8$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

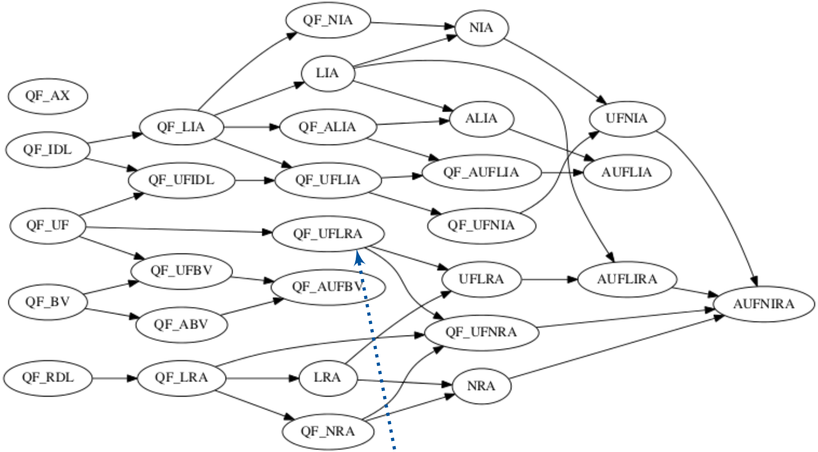


(Quantifier-free) real/integer non-linear arithmetic

$$x^2 + 2xy + y^2 \geq 0$$

Source: http://smtlib.cs.uiowa.edu/logics.shtml

SMT-LIB theories

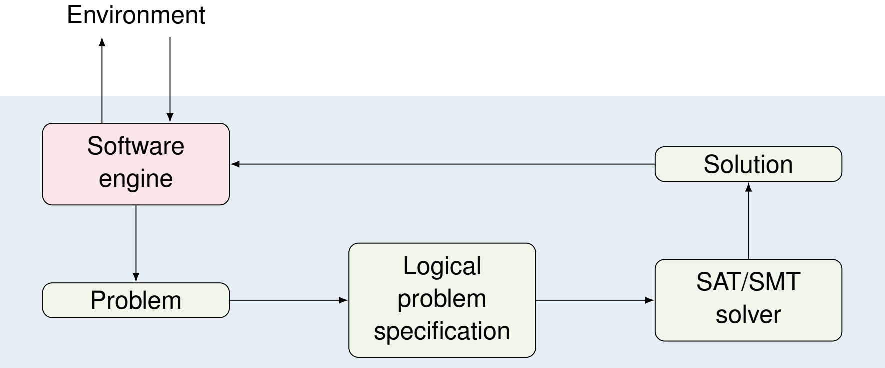


Combined theories
$2f(x) + 5y > 0$

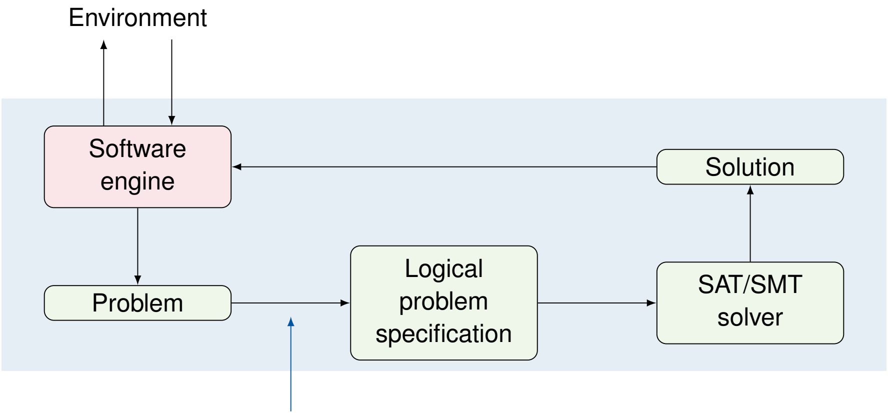
Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

Contents

Embedding SAT/SMT solvers

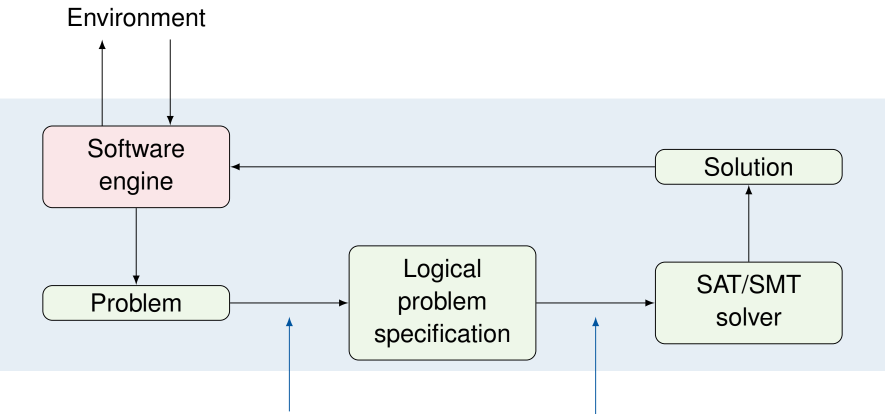


Embedding SAT/SMT solvers



Encoding: SAT/SMT-LIB standard
elaborate encoding is extremely important!

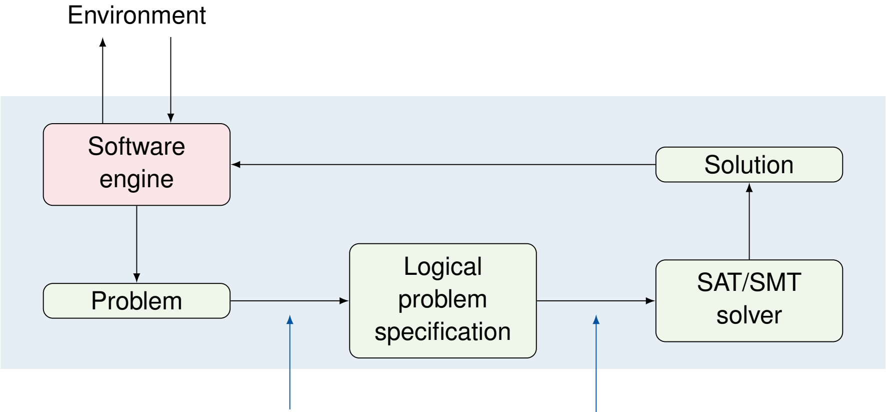
Embedding SAT/SMT solvers



Encoding: SAT/SMT-LIB standard
elaborate encoding is extremely important!

standard input syntax → free solver choice

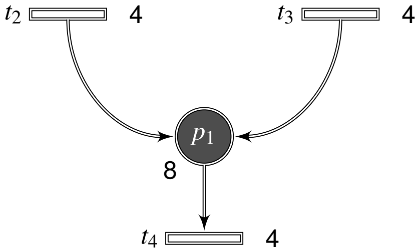
Embedding SAT/SMT solvers



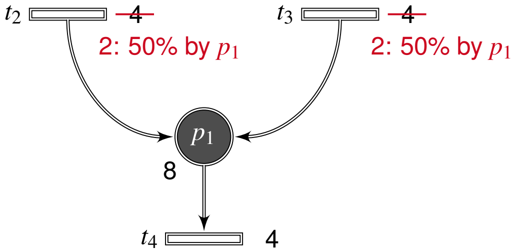
Encoding: SAT/SMT-LIB standard
elaborate encoding is extremely important!

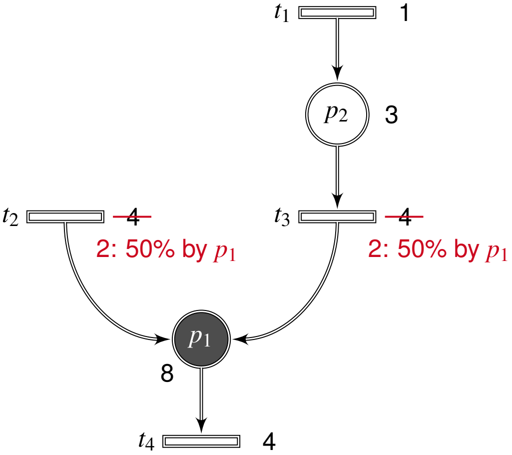
standard input syntax → free solver choice

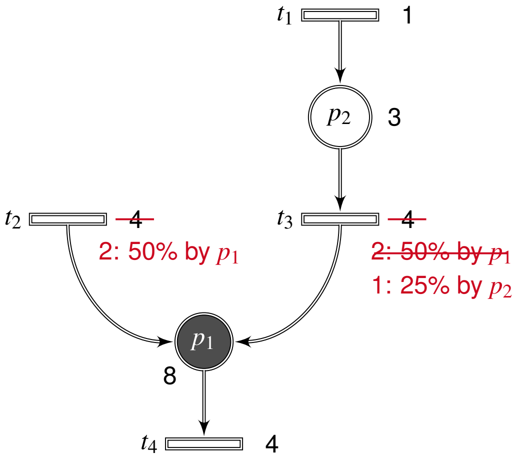
Next: some applications of SMT solvers


t_2
4
t_3
4
p_1
8
t_4
4

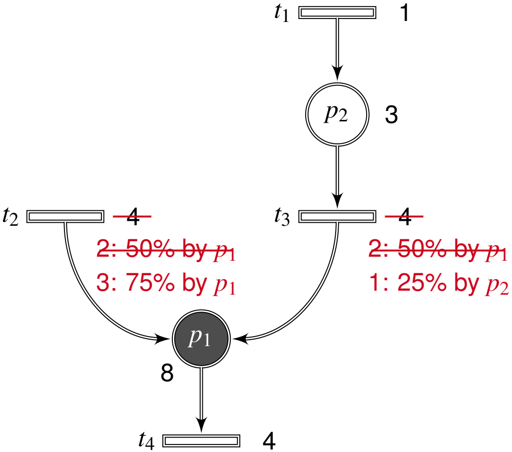
1. Rate adaption in hybrid Petri nets




t_1
1
p_2
3
t_2
~~4~~
2: 50% by p_1
t_3
~~4~~
2: 50% by p_1
p_1
8
t_4
4


t_1
1
p_2
3
t_2
~~4~~
2: 50% by p_1
t_3
~~4~~
~~2: 50% by p_1~~
1: 25% by p_2
p_1
8
t_4
4

1. Rate adaption in hybrid Petri nets



1. SMT encoding of rate adaption fixedpoint

$$(1) \quad \Big[\bigwedge_{p\in P} 0 \le \mathtt{factor}_p \le 1\Big] \wedge \Big[\bigwedge_{t\in T_a} 0 \le \mathtt{factor}_t \le 1\Big] \wedge$$

$$(2) \quad \Big[\bigwedge_{t\in T_a} ((\mathtt{owner}_t = \mathit{source}(t) \wedge \mathtt{owner}_t \in P_{\mathit{empty}}) \vee (\mathtt{owner}_t = \mathit{target}(t) \wedge \mathtt{owner}_t \in P_{\mathit{full}}))\Big] \wedge$$

$$(3) \quad \Big[\bigwedge_{p\in P} \mathtt{in}_p = (\textstyle\sum_{t\in \mathit{In}(p)\cap T_a} \mathtt{factor}_t \cdot \mathit{nominal\_rate}(t)) + (\sum_{t\in \mathit{In}(p)\cap T_{na}} \mathit{nominal\_rate}(t)) \wedge$$

$$\mathtt{out}_p = (\textstyle\sum_{t\in \mathit{Out}(p)\cap T_a} \mathtt{factor}_t \cdot \mathit{nominal\_rate}(t)) + (\sum_{t\in \mathit{Out}(p)\cap T_{na}} \mathit{nominal\_rate}(t))\Big] \wedge$$

$$(4) \quad \Big[\bigwedge_{p\in P_{\mathit{empty}}} \Big((\mathtt{factor}_p = 1 \vee \bigvee_{t\in \mathit{Out}(p)} \mathtt{owner}_t = p) \wedge$$

$$(\bigwedge_{t\in \mathit{Out}(p)} (\mathtt{owner}_t = p \rightarrow \mathtt{factor}_t = \mathtt{factor}_p) \wedge$$

$$(\mathtt{owner}_t \ne p \rightarrow \mathtt{factor}_t < \mathtt{factor}_p) \quad) \wedge$$

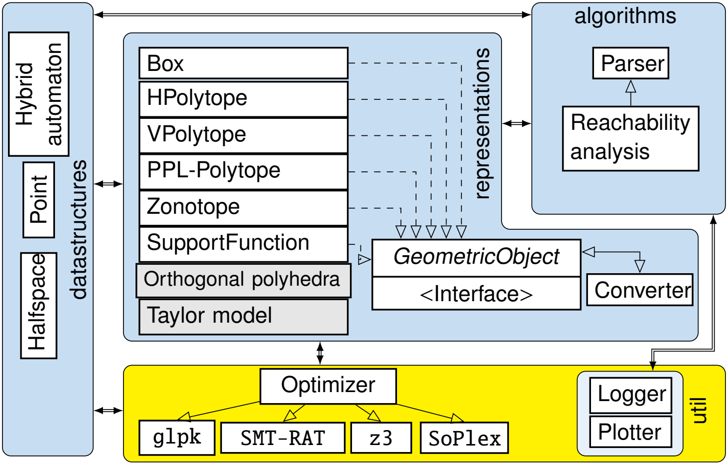
$$\mathtt{in}_p \ge \mathtt{out}_p \wedge (\mathtt{factor}_p < 1 \rightarrow \mathtt{in}_p = \mathtt{out}_p) \quad \Big)\Big] \wedge$$

$$(5) \quad \Big[\bigwedge_{p\in P_{\mathit{full}}} \Big((\mathtt{factor}_p = 1 \vee \bigvee_{t\in \mathit{In}(p)} \mathtt{owner}_t = p) \wedge$$

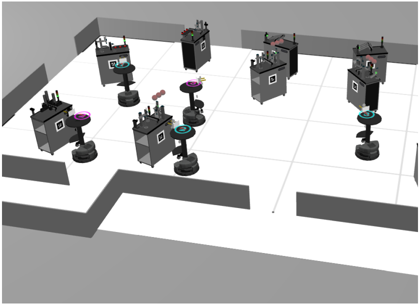
$$(\bigwedge_{t\in \mathit{In}(p)} (\mathtt{owner}_t = p \rightarrow \mathtt{factor}_t = \mathtt{factor}_p) \wedge$$

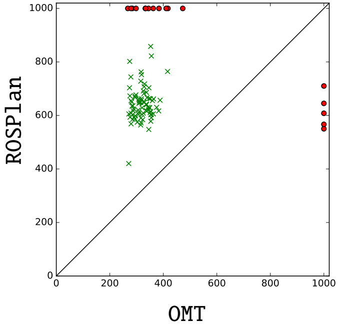
$$(\mathtt{owner}_t \ne p \rightarrow \mathtt{factor}_t \le \mathtt{factor}_p) \quad) \wedge$$

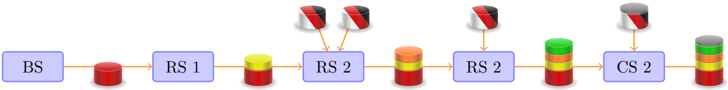
$$\mathtt{in}_p \le \mathtt{out}_p \wedge (\mathtt{factor}_p < 1 \rightarrow \mathtt{in}_p = \mathtt{out}_p) \quad \Big)\Big]$$



Source: E. Ábrahám, X. Chen, S. Sankaranarayanan, S. Schupp. PhD Chen, PhD Schupp, Information and Computation'22, IRI'18, SEFM'18, TACAS'18, NFM'17, QAPL'17, ARCH'15, CyPhy'15, NFM'15, FMCAD'14, CAV'13, FTSCS'13, NOLCOS'13, RTSS'12, EUROCAST'11, RP'11.

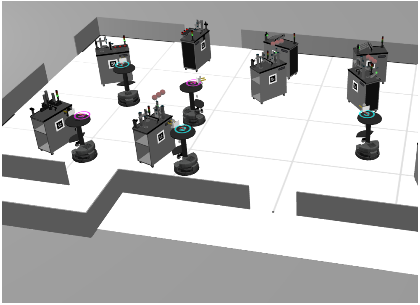


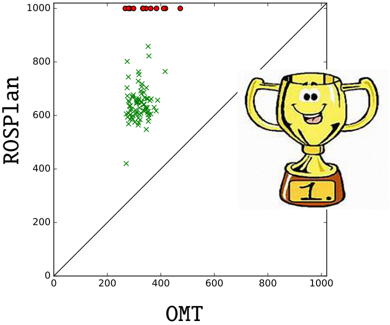


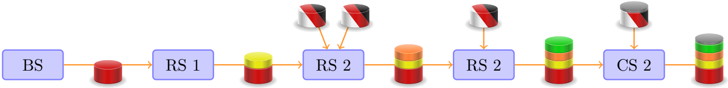


Source: E. Ábrahám, G. Lakemeyer, F. Leofante, T. D. Niemüller, A. Tacchella.

PhD Leofante, IJCAI'20, Information Systems Frontiers 2019, ECMS'19, AAAI'18, iFM'18, ICAPS'17, PlanRob'17, IRI'17.







Source: E. Ábrahám, G. Lakemeyer, F. Leofante, T. D. Niemüller, A. Tacchella.

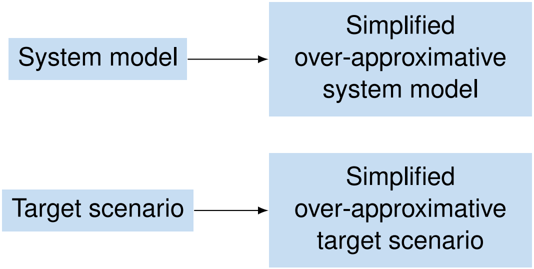
PhD Leofante, IJCAI'20, Information Systems Frontiers 2019, ECMS'19, AAAI'18, iFM'18, ICAPS'17, PlanRob'17, IRI'17.

4. Relevant domains for testing (Siemens)

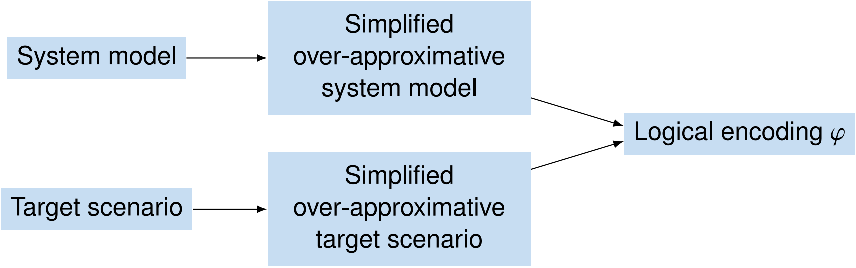
System model

Target scenario

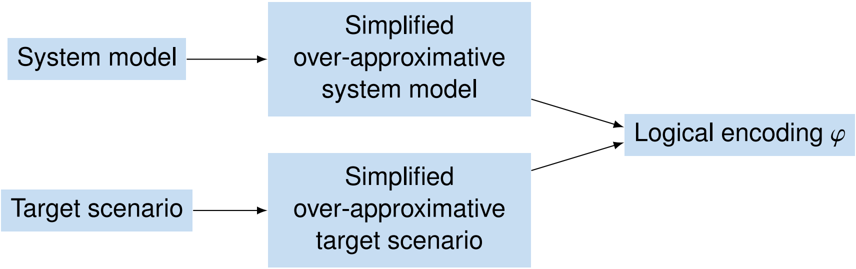
4. Relevant domains for testing (Siemens)



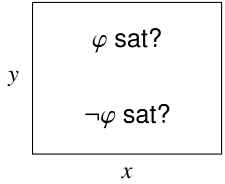
4. Relevant domains for testing (Siemens)

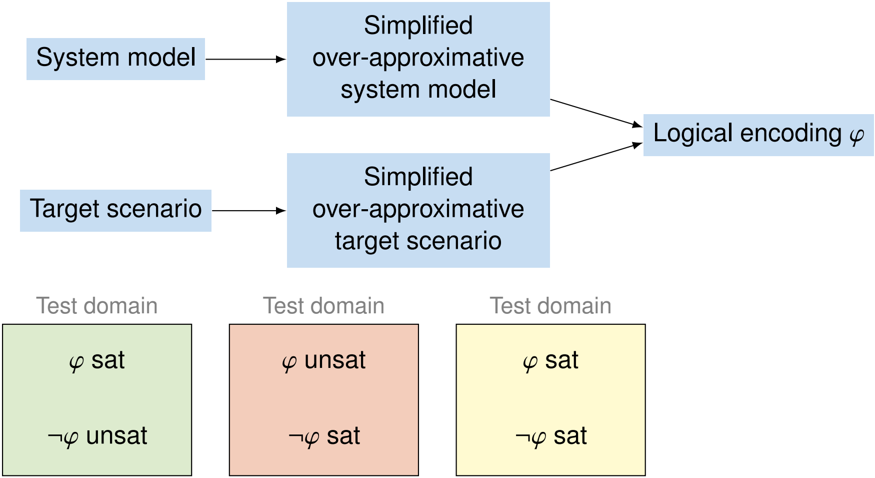


4. Relevant domains for testing (Siemens)

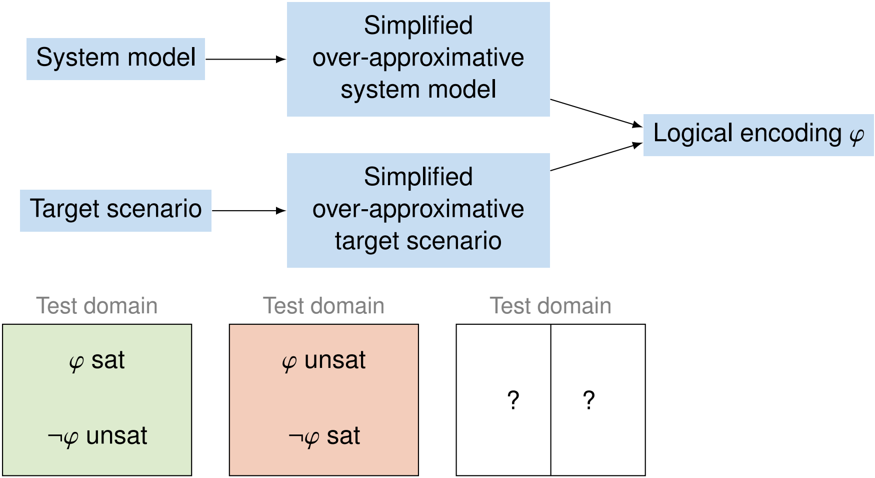


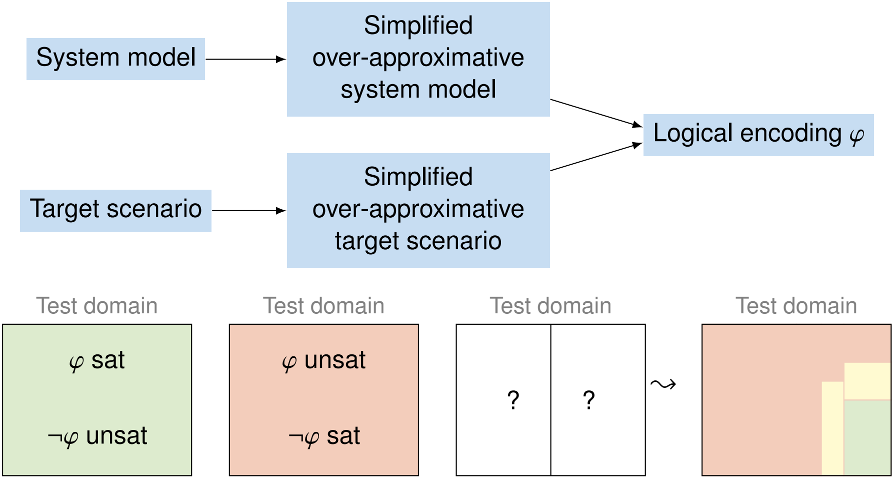
Test domain

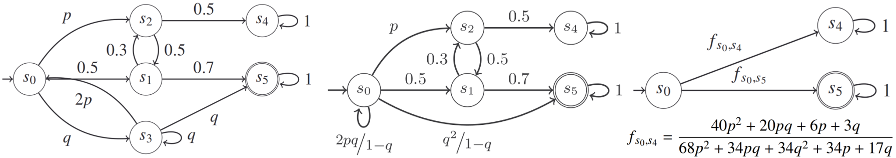


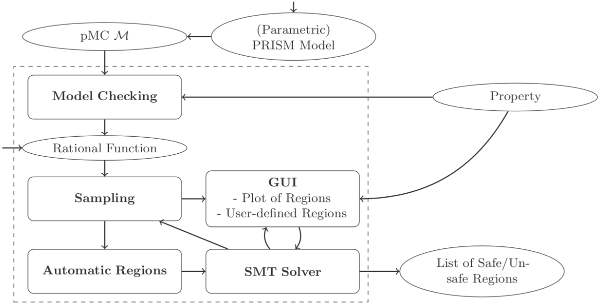

System model
Simplified
over-approximative
system model
Target scenario
Simplified
over-approximative
target scenario
Logical encoding φ
Test domain
φ sat
$\neg\varphi$ unsat
Test domain
φ unsat
$\neg\varphi$ sat
Test domain
φ sat
$\neg\varphi$ sat

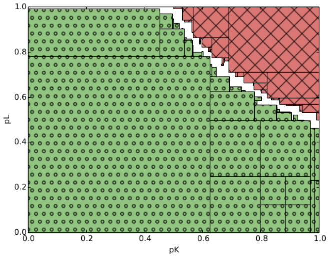
4. Relevant domains for testing (Siemens)




System model
Simplified over-approximative system model
Target scenario
Simplified over-approximative target scenario
Logical encoding φ
Test domain
φ sat
$\neg\varphi$ unsat
Test domain
φ unsat
$\neg\varphi$ sat
Test domain
?
?
$\rightsquigarrow$
Test domain







Source: C. Dehnert, S. Junges, N. Jansen, F. Corzilius, M. Volk, H. Bruintjes, J.-P. Katoen, E. Ábrahám.

PROPhESY: A probabilistic parameter synthesis tool.

In Proc. of CAV'15.

Contents

Usage of SMT solvers

- Standard input language, benchmarks
- Online usage, command-line, programming interfaces
- Black-box usage possible, but specific knowledge is advantageous
  - for efficient usage and
  - selection of the best fitting tool (e.g. fast vs complete).

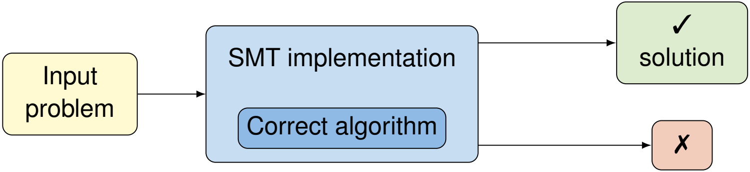
Proof generation

- Theoretical basics: algorithms with correctness proofs.

Correct algorithm

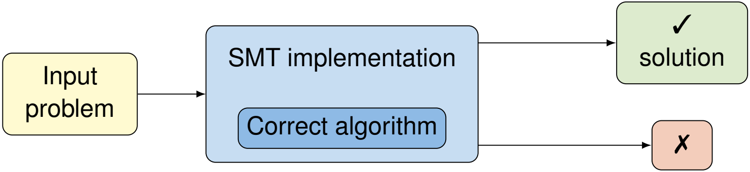
Proof generation

- Theoretical basics: algorithms with correctness proofs.
- Reliable tools: in QF_NRA for SMT-COMP'21, no bugs discovered on large benchmark sets.



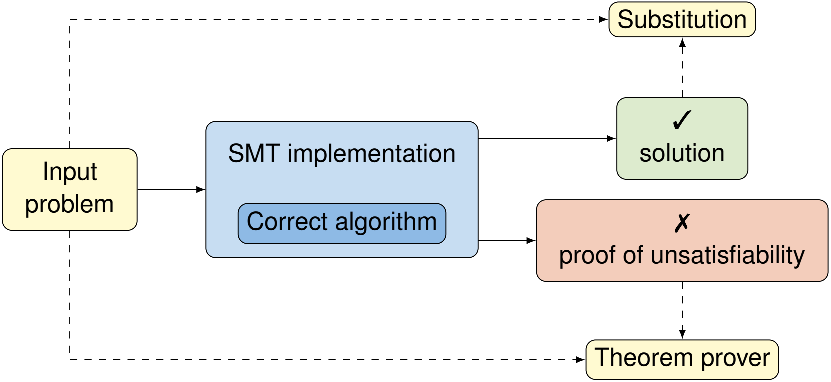
Proof generation

- Theoretical basics: algorithms with correctness proofs.
- Reliable tools: in QF_NRA for SMT-COMP'21, no bugs discovered on large benchmark sets.
- But still: bugs can remain undetected for a long time.



Proof generation

- Theoretical basics: algorithms with correctness proofs.
- Reliable tools: in QF_NRA for SMT-COMP'21, no bugs discovered on large benchmark sets.
- But still: bugs can remain undetected for a long time.
- Solution: automatically checkable proof certificates.



Further functionalities

- Model generation
- Explanations of unsatisfiability (unsat cores, interpolants)
- Optimization

- Satisfiability for quantified formulas
- Quantifier elimination (get all solutions symbolically)

- Scalability
  - Preprocessing
  - Heuristics, especially variable ordering
  - Machine learning
  - Closer integration of decision procedures
  - Parallelization

Contents

SMT-LIB theories

Syntax of core theory

```
      :sorts ((Bool 0))
      :funs (
(true Bool)
(false Bool)
(not Bool Bool)
(and Bool Bool Bool :left-assoc)
...
(par (A) (= A A Bool :chainable))
(par (A) (ite Bool A A A))
...
```

SMT-LIB theories

Syntax of real theory

```
       :sorts ((Real 0))
       :funs (
...
(+ Real Real Real :left-assoc)
(* Real Real Real :left-assoc)
...
(<  Real Real Bool :chainable)
...
       )
```

SMT-LIB commands

- Lisp-like script language
- Supported by essentially all SMT solvers
- Easy to parse and extend

Boolean example

```
(set-logic QF_UF)
(declare-const p Bool)
(assert (and p (not p)))
(check-sat)
```

SMT-LIB commands

- Lisp-like script language
- Supported by essentially all SMT solvers
- Easy to parse and extend

Linear integer example

```
(set-logic QF_LIA)
(declare-const x Int)
(declare-const y Int)
(assert (= (- x y) (+ x (- y) 1)))
(check-sat)
```

SMT-LIB commands

- Lisp-like script language
- Supported by essentially all SMT solvers
- Easy to parse and extend

Unsatisfiable cores

```
(set-logic QF_UF)
(set-option :produce-unsat-cores true)
(declare-const p Bool)
(declare-const q Bool)
(declare-const r Bool)
(assert (! (=> p q) :named a))
(assert (! (=> q r) :named b))
(assert (! (not (=> p r)) :named c))
(assert ...)
(check-sat)
(get-unsat-core)
```

SMT-LIB commands

- Lisp-like script language
- Supported by essentially all SMT solvers
- Easy to parse and extend

Optimization

```
(set-logic QF_LIA)
(declare-const x Int)
(declare-const y Int)
(assert (and (< y 5) (< x 2)))
(assert (< (- y x) 1))
(maximize (+ x y))
(check-sat)
(get-objectives)
```

Solving theory formulas with SMT solvers

- https://cvc5.github.io/tutorials/beginners
- SMT-LIB input:
  https://microsoft.github.io/z3guide/docs/logic/intro/
  https://smt-lib.org/examples.shtml
- Z3/cvc5 Python interface:
  https://ericpony.github.io/z3py-tutorial/guide-examples.htm