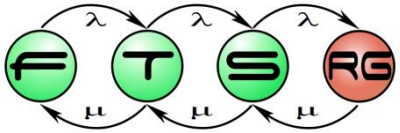
Satisfiability Modulo Theories

Tamás Tóth
totht@mit.bme.hu



First Order Logic

First Order Logic - Syntax

- Signature $\Sigma = \Sigma^F \cup \Sigma^P$
  - Σ^F: function symbols
  - Σ^P: predicate symbols
  - Together with their arities
  - Example: $\langle 0,1,+,\cdot,\leq,\doteq \rangle$ for Presburger arithmetic
- Syntax
  - Var is a countable set of variables
  - $\varphi ::= P^n(t_1, t_2, \ldots, t_n) \mid \top \mid \neg\varphi_0 \mid \varphi_1 \wedge \varphi_2 \mid \forall x.\varphi_0$
  - $t ::= x \mid F^n(t_1, t_2, \ldots, t_n)$

First Order Logic - Semantics

- Interpretetion I
  - D is a set (the domain)
  - $[\![P^n]\!]_I \subseteq D^n$
  - $[\![F^n]\!]_I : D^n \to D$
  - $[\![x]\!]_I \in D$
- Semantics
  - $[\![F(t_1, t_2, \ldots, t_n)]\!] = [\![F]\!]([\![t_1]\!], [\![t_2]\!], \ldots, [\![t_n]\!])$
  - $[\![P(t_1, t_2, \ldots, t_n)]\!]$ iff $([\![t_1]\!], [\![t_2]\!], \ldots, [\![t_n]\!]) \in [\![P]\!]$
  - (Connectives as in porpositional logic)
  - $[\![\forall x.\, \varphi]\!]_I$ iff $[\![\varphi]\!]_{I[x \leftarrow d]}$ for all $d \in D$
- $I \vDash \varphi$ iff $[\![\varphi]\!]_I$

Interpretations and formulas

- Models of a formula
  - $Mod(\varphi) = \{I \mid I \vDash \varphi\}$
  - $Mod(T) = \bigcap_{\varphi \in T} Mod(\varphi)$
- Satisfiability
  - $Mod(\varphi) \neq \emptyset$
- Validity: $\vDash \varphi$
  - $I \in Mod(\varphi)$ for all I
- Entailment: $T \vDash \varphi$
  - $Mod(T) \subseteq Mod(\varphi)$
- Equivalence: $\varphi \equiv \psi$
  - $Mod(\varphi) = Mod(\psi)$

Reduction to Satisfiability

- Validity
  - $\models \varphi$ iff $\neg\varphi$ is unsatisfiable
- Entailment
  - $\varphi \models \psi$ iff $\models \varphi \rightarrow \psi$ (Deduction Th.)
  - $\varphi \models \psi$ iff $\varphi \wedge \neg\psi$ is unsatisfiable
- Equivalence
  - $\varphi \equiv \psi$ iff $\varphi \models \psi$ and $\psi \models \varphi$
  - $\varphi \equiv \psi$ iff $\models \varphi \rightarrow \psi$ and $\models \psi \rightarrow \varphi$
  - $\varphi \equiv \psi$ iff $\varphi \wedge \neg\psi$ and $\psi \wedge \neg\varphi$ are unsatisfiable

The Satisfiability Problem

- Satisfiability problem
  - Instance: a first-order formula φ
  - Question: is φ satisfiable?
- The complement: validity
  - Hilbert „Entscheidungsproblem” (1928)
  - Validity is undecidable (Church-Turing Th., 1936)
  - Validity is semidecidable: e.g. resolution
- Thus satisfiability in FOL is not even semidecidable

Satisfiability Modulo Theories

First Order Theories

- Theory T over a signature Σ:
  - a set of closed Σ-formulas
- Satisfiability in T
  - $Mod(\varphi) \cap Mod(T) \neq \emptyset$
- Validity in T: $\models_T \varphi$
  - $I \in Mod(\varphi)$ for all $I \in Mod(T)$
- Entailment in T: $\varphi \models_T \psi$
  - $Mod(\varphi) \cap Mod(T) \subseteq Mod(\psi) \cap Mod(T)$
- Theory of an interpretation:
  - $Th(I) = \{\varphi \mid I \models \varphi\}$
  - $Th(\mathfrak{I}) = \bigcap_{I \in \mathfrak{I}} Th(I)$

Motivation

- First order logic is undecidable
- However, it can be restricted
  - Syntactically
  - Semantically

  to enable efficient decision procedures

→ First order theories

Equality with uninterpreted symbols

- $\Sigma = \{\doteq, f_1, f_2, \ldots, p_1, p_2, \ldots\}$
  - $\doteq$ is an interpreted symbol
  - $f_1, f_2, \ldots, p_1, p_2, \ldots$ are uninterpreted symbols
- Axioms of $\mathcal{EUF}$:
  1. $\forall x. x \doteq x$
  2. $\forall x, y. (x \doteq y \rightarrow y \doteq x)$
  3. $\forall x, y, z. (x \doteq y \wedge y \doteq z \rightarrow x \doteq z)$
  4. $\forall x_1, y_1, \ldots x_n, y_n.$
     $$\left(\bigwedge_{i=1}^{n} x_i \doteq y_i \rightarrow f(x_1, \ldots, x_n) \doteq f(y_1, \ldots, y_n)\right)$$
  1. $\forall x_1, y_1, \ldots x_n, y_n.$
     $$\left(\bigwedge_{i=1}^{n} x_i \doteq y_i \rightarrow (p(x_1, \ldots, x_n) \leftrightarrow p(y_1, \ldots, y_n))\right)$$
- Undecidable in general
- NP-complete for quantifier-free fragment

Linear Arithmetic

- Linear arithmetic $\mathcal{LA}$

$$\sum c_i x_i \leq b$$

  - $\mathcal{LA}(\mathbb{Q})$: P
  - $\mathcal{LA}(\mathbb{Z})$: NP-complete
- Unit-two-variable-per-inequality $\mathcal{UTVPI}$

$$x_i \pm x_j \leq b$$

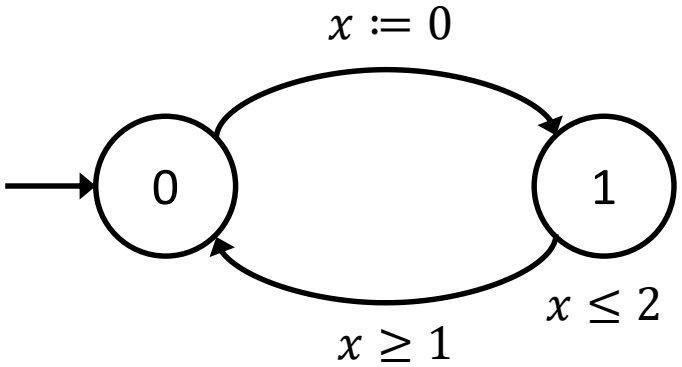
  - $\mathcal{UTVPI}(\mathbb{Z})$: P
- Difference logic $\mathcal{DL}$

$$x_i - x_j \leq b$$

Example

- $\mathcal{LA}(\mathbb{Q})$:

$$\ell_0 = 0 \wedge x_0 = 0$$
$$\ell_1 = 1 \wedge x_1 = 0 \wedge d_1 \geq 0$$
$$\ell_2 = 0 \wedge x_2 = x_1 + d_1 \wedge x_2 \leq 2 \wedge x_2 \geq 1$$



Arrays

- $\Sigma = \{rd, wr, \doteq\}$
- Axioms:
  1. Axioms of equality for rd, wr
  2. $\forall a, i, v. rd(wr(a, i, v), i) \doteq v$
  3. $\forall a, i, j, v. \left(i \neq j \rightarrow rd(wr(a, i, v), j) \doteq rd(a, j)\right)$
  4. $\forall a, b. (\forall i. rd(a, i) \doteq rd(b, i) \rightarrow a \doteq b)$
- Quantifier-free fragment is NP-complete

LISP-like Lists

- $\Sigma = \{cons, car, cdr, atom, \doteq\}$
- Axioms
  1. Axioms for equality for $cons, car, cdr, atom$
  2. $\forall x, y.\, car\big(cons(x, y)\big) \doteq x$
  3. $\forall x, y.\, cdr\big(cons(x, y)\big) \doteq y$
  4. $\forall x.\, (\neg atom(x) \rightarrow cons\big(car(x), cdr(x)\big) \doteq x)$
  5. $\forall x, y.\, (\neg atom(cons(x, y)))$

Combination of Theories

- Nelson-Oppen Combination Method
  - $\Sigma_1 \cap \Sigma_2 = \{\doteq\}$
  - T_1 and T_2 are stably infinite: satisfiable quantifier-free formulas have an infinite model

  Then the quantifier-free fragment of the combined theory $T_1 \oplus T_2$ is decidable

Basic idea

- Purification

$$\varphi: 1 \leq x, x \leq 2, f(x) \neq f(1), f(x) \neq f(2)$$

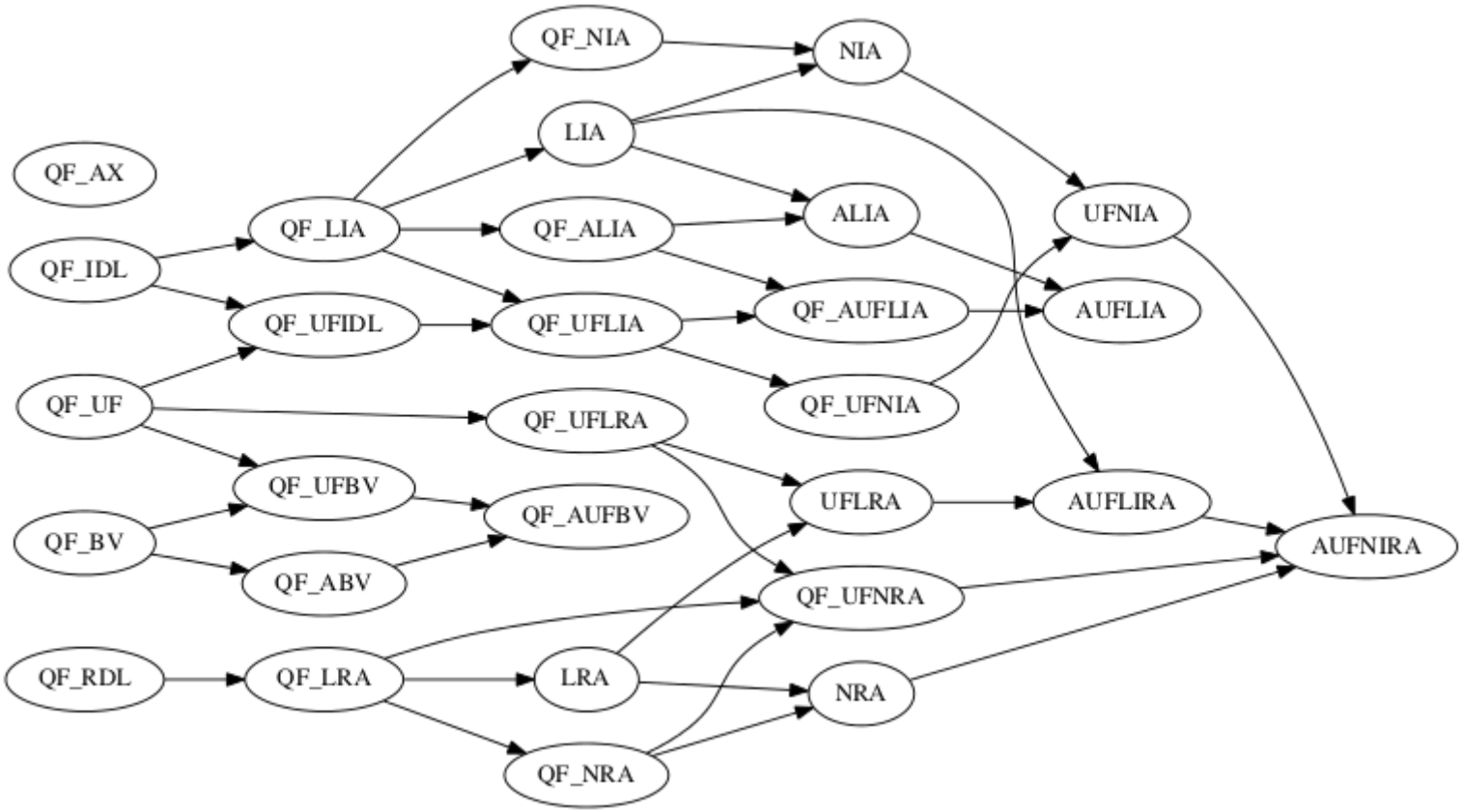
$$\rightarrow$$

$$\varphi_1: 1 \leq x, x \leq 2, y \doteq 1, z \doteq 2$$

$$\varphi_2: f(x) \neq f(y), f(x) \neq f(z)$$

- Partitioning
  - $\{\{x, y, z\}\}$ – inconsistent with φ_2
  - $\{\{x, y\}, \{z\}\}$ – inconsistent with φ_2
  - $\{\{x, z\}, \{y\}\}$ – inconsistent with φ_2
  - $\{\{x\}, \{y, z\}\}$ – inconsistent with φ_1
  - $\{\{x\}, \{y\}, \{z\}\}$
    - In $\mathcal{LA}(\mathbb{Z})$: inconsistent with φ_1 → UNSAT
    - In $\mathcal{LA}(\mathbb{Q})$: consistent with φ_1 (and φ_2) → SAT

Some Combined Theories



Examples

- $\mathcal{LA}(\mathbb{Z})$ + Arrays

$$\forall i. rd(a, i) \leq rd(a, i + 1)$$

- $\mathcal{EUF}$ + $\mathcal{LA}(\mathbb{Z})$ + Lists

$$atom(x) \rightarrow l(x) \doteq 1$$
$$\neg atom(x) \rightarrow l(x) \doteq l\big(car(x)\big) + l(cdr(x))$$

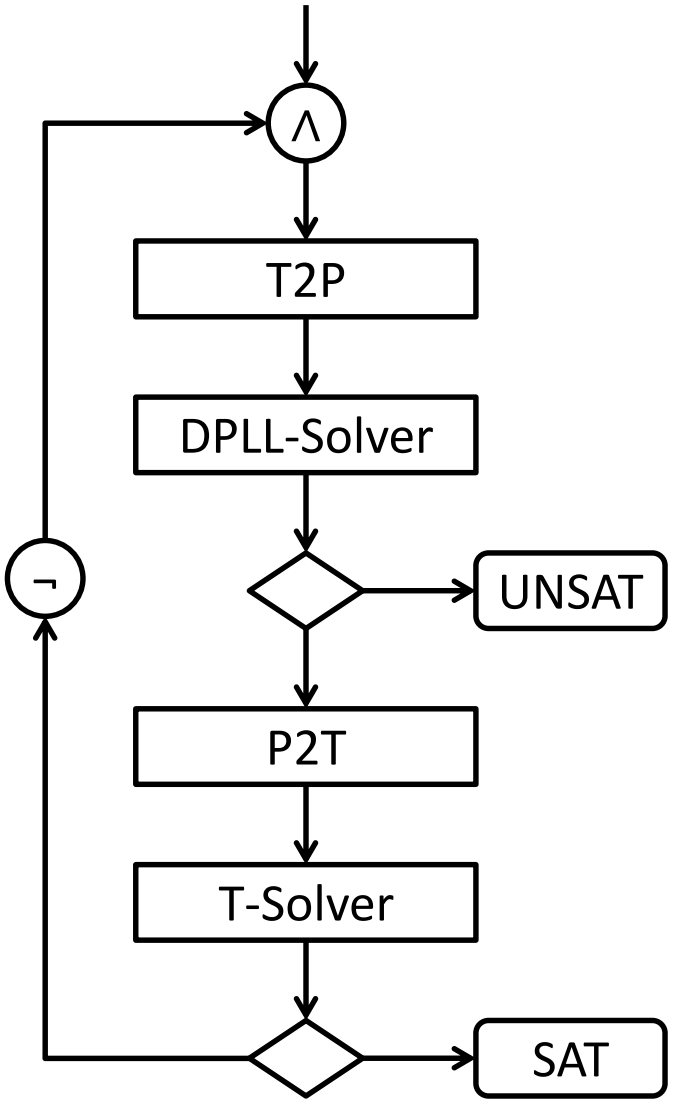
SMT Solving

- Eager SMT Solving:
  - Transform the formula to an equisatisfiable propositional formula

- Boolean reasoning + Theory reasoning
  - Naive approach
    - transform to DNF
    - reason only about conjuncts of theory atoms
  - More efficient approach:
    - Enumerate conjuncts lazily with a SAT solver

    → Lazy SMT Solving

Lazy SMT Solving



Lazy SMT Solving

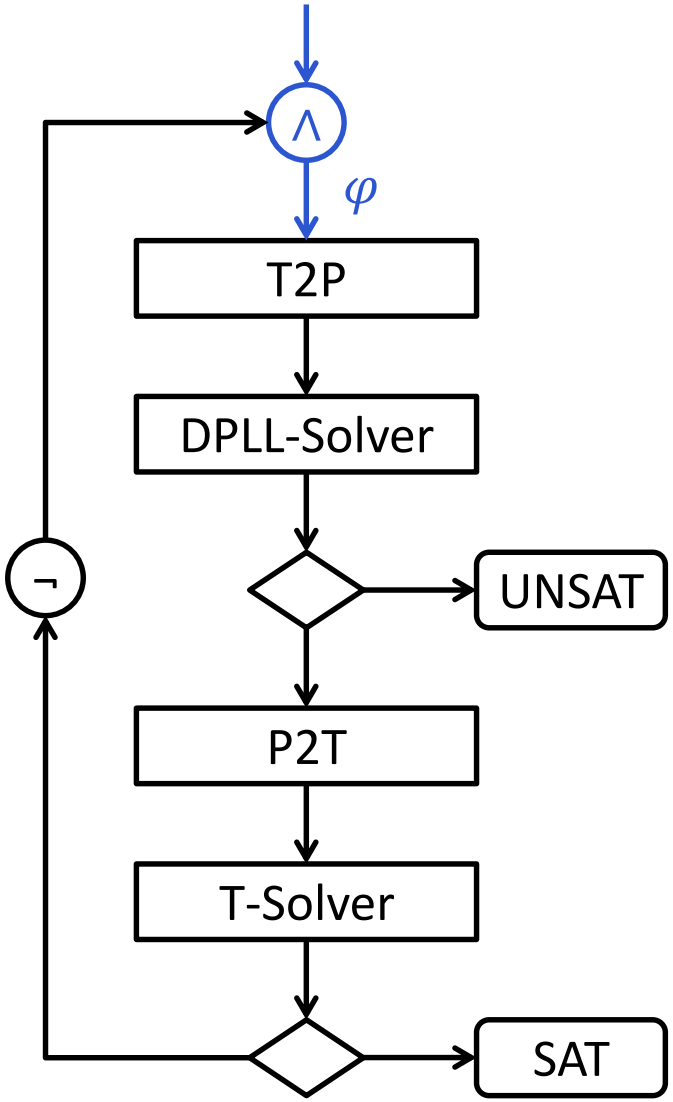
$(0 \leq x_1 + 2x_2) \vee p$

$\neg(0 \leq -3x_1 - x_2 + 1) \vee \neg q$

$0 \leq -3x_1 - x_2 + 1$

$(0 \leq -3x_2 - 2) \vee q$

$\neg p \vee q$



Lazy SMT Solving

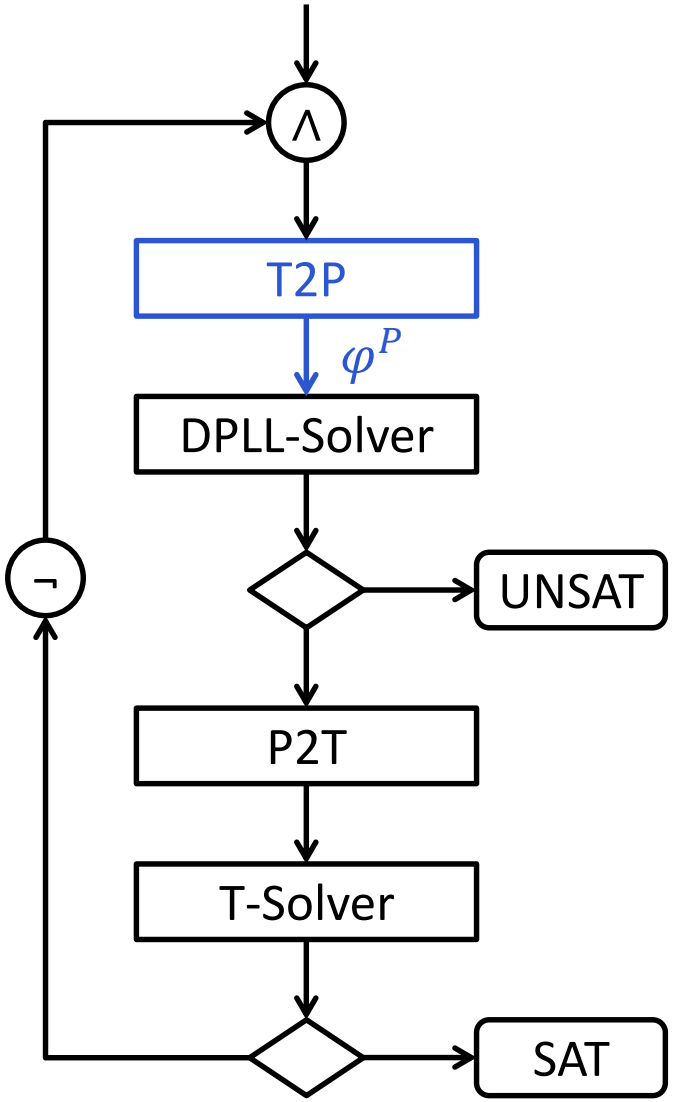
$a \vee p$

$\neg b \vee \neg q$

b

$c \vee q$

$\neg p \vee q$



Lazy SMT Solving

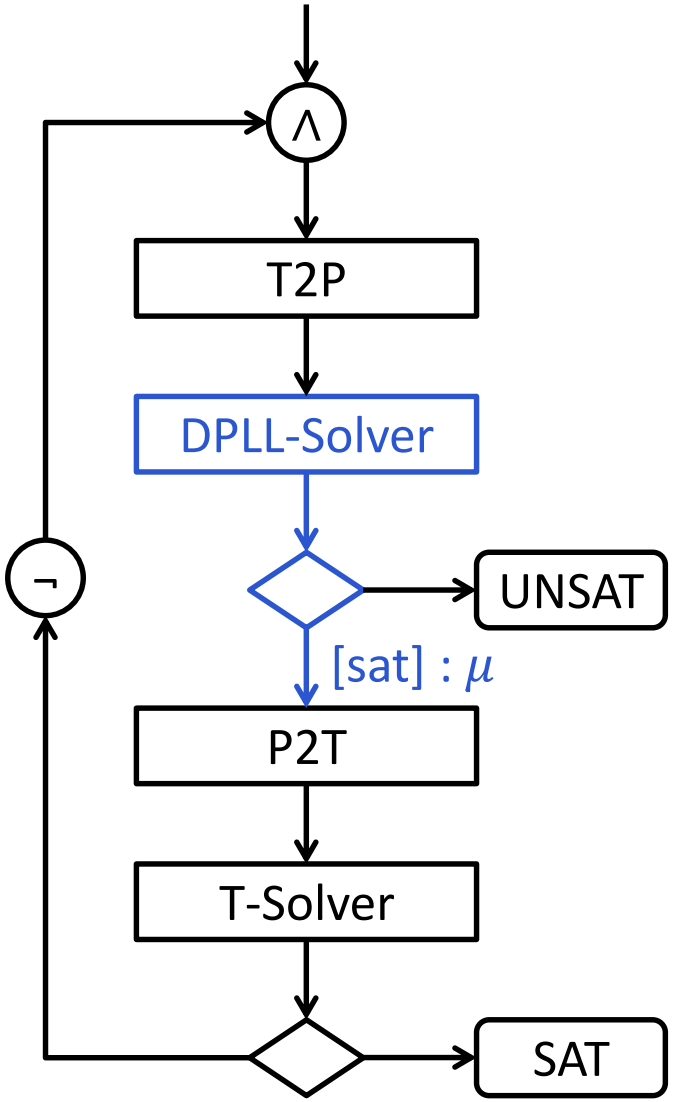
$a \mapsto 1$

$b \mapsto 1$

$c \mapsto 1$

$p \mapsto 0$

$q \mapsto 0$

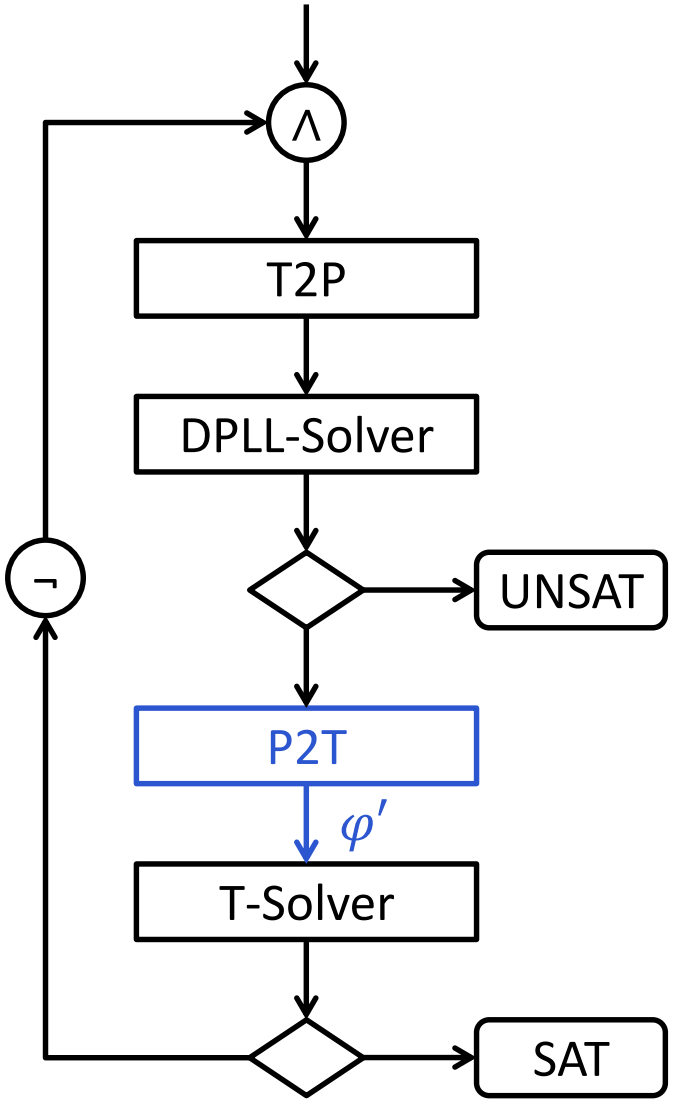


Lazy SMT Solving

$$0 \leq x_1 + 2x_2$$

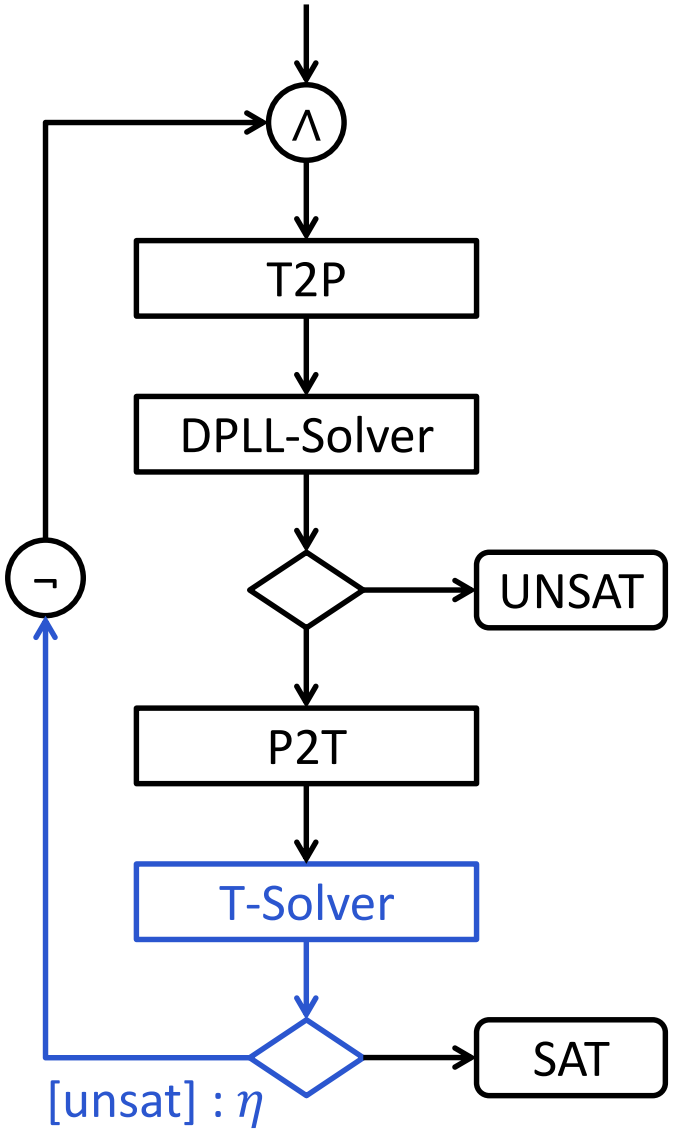
$$0 \leq -3x_1 - x_2 + 1$$

$$0 \leq -3x_2 - 2$$



Lazy SMT Solving

$$\{$$
$$0 \leq x_1 + 2x_2,$$
$$0 \leq -3x_1 - x_2 + 1,$$
$$0 \leq -3x_2 - 2$$
$$\}$$



Lazy SMT Solving

$(0 \leq x_1 + 2x_2) \vee p$

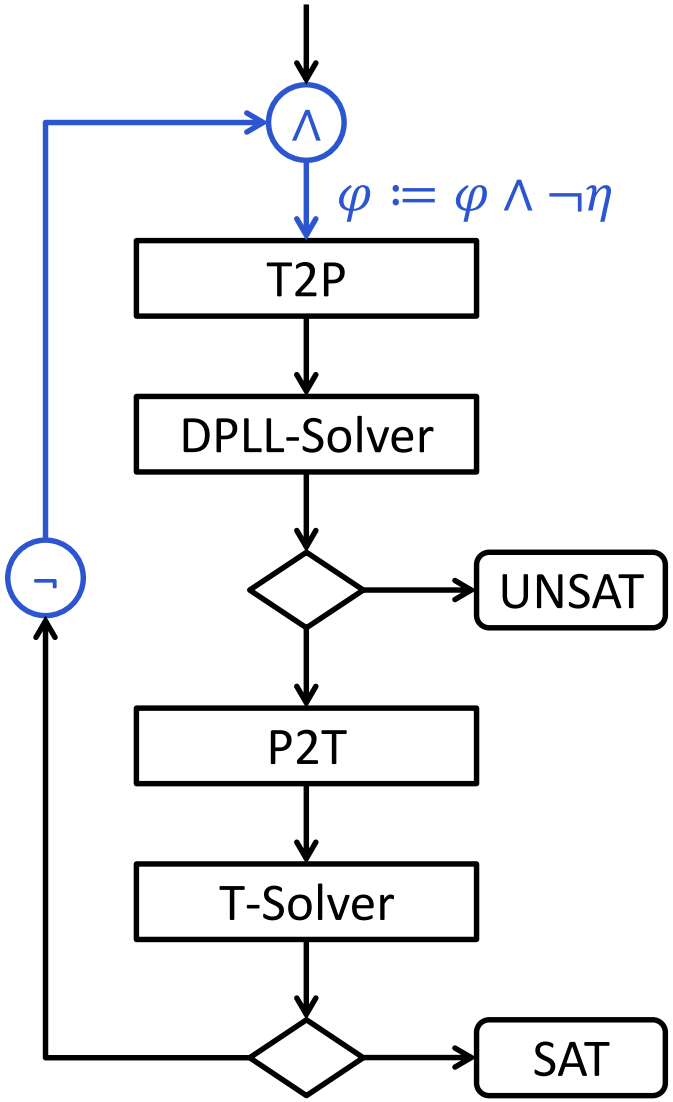
$\neg(0 \leq -3x_1 - x_2 + 1) \vee \neg q$

$0 \leq -3x_1 - x_2 + 1$

$(0 \leq -3x_2 - 2) \vee q$

$\neg p \vee q$

$\neg(0 \leq x_1 + 2x_2) \vee$
$\neg(0 \leq -3x_1 - x_2 + 1) \vee$
$\neg(0 \leq -3x_2 - 2)$



Lazy SMT Solving

$a \vee p$

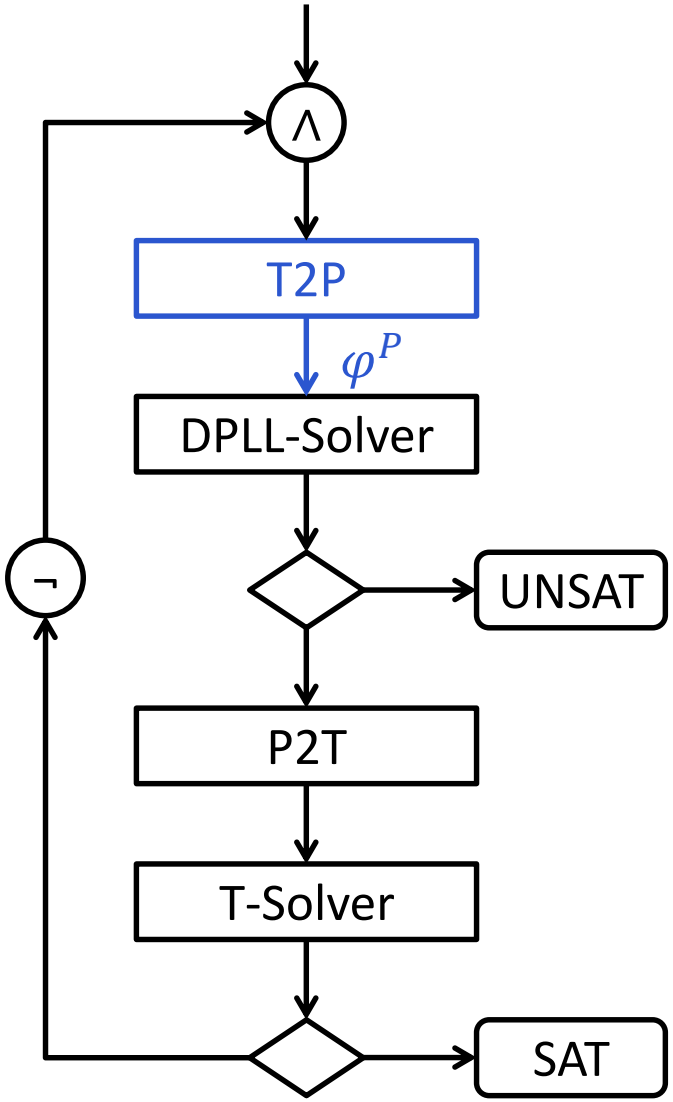
$\neg b \vee \neg q$

b

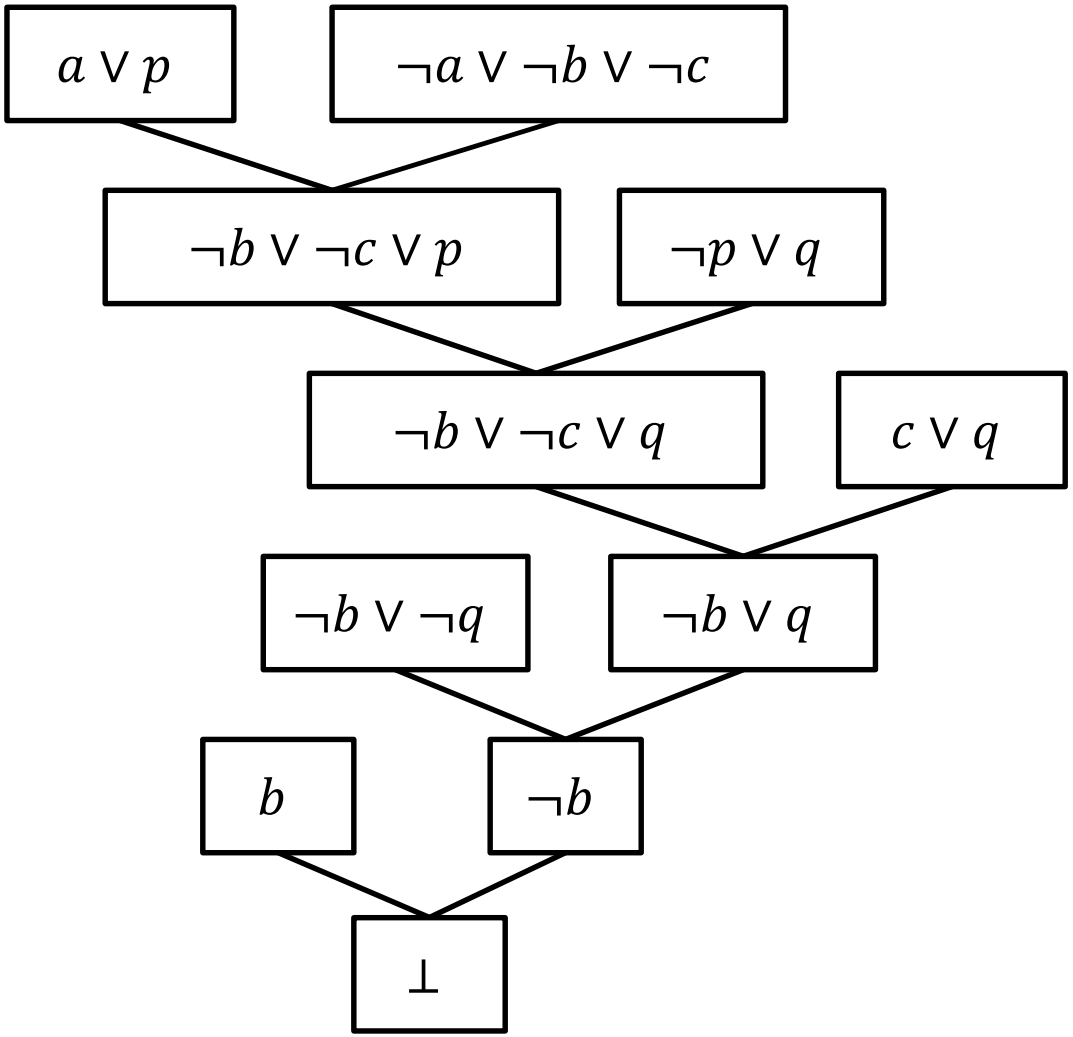
$c \vee q$

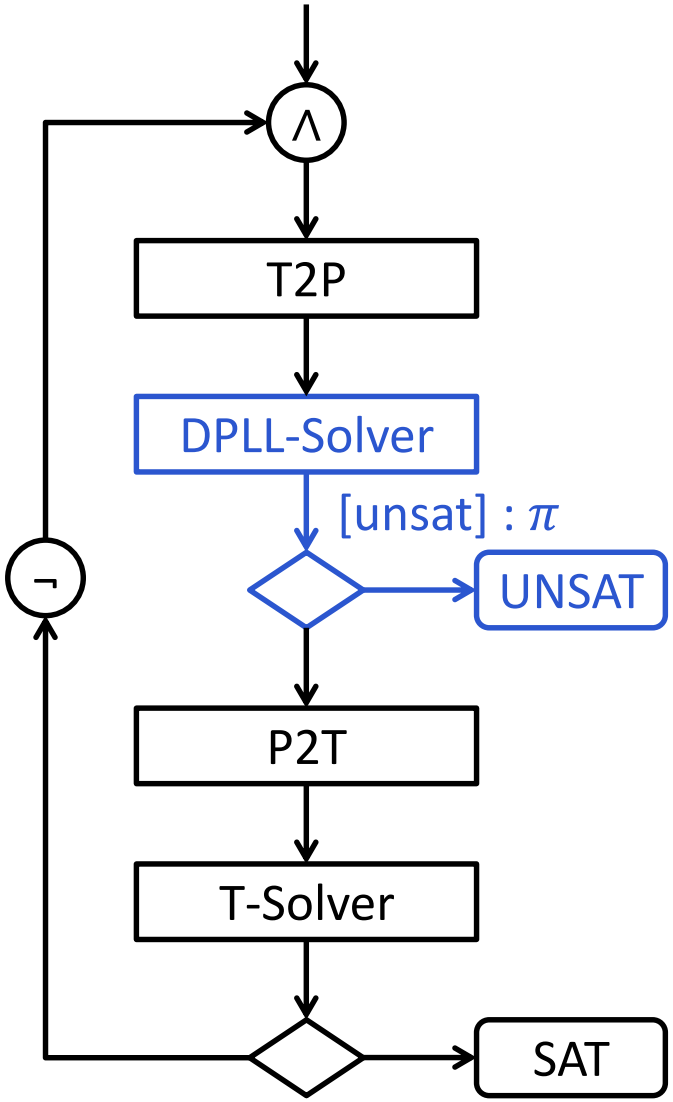
$\neg p \vee q$

$\neg a \vee \neg b \vee \neg c$



Lazy SMT Solving





SMT in practice

- http://rise4fun.com/z3/tutorial

z3 Microsoft Research

Is this formula satisfiable?

```
1  ; This example illustrates basic arithmetic and
2  ; uninterpreted functions
3
4  (declare-fun x () Int)
5  (declare-fun y () Int)
6  (declare-fun z () Int)
7  (assert (>= (* 2 x) (+ y z)))
8  (declare-fun f (Int) Int)
9  (declare-fun g (Int Int) Int)
10 (assert (< (f x) (g x x)))
11 (assert (> (f y) (g x x)))
12 (check-sat)
13 (get-model)
14 (push)
15 (assert (= x y))
16 (check-sat)
17 (pop)
18 (exit)
19
```