

# Satisfiability Modulo Theories

Verifying cyberphysical systems

Sayan Mitra

[mitras@illinois.edu](mailto:mitras@illinois.edu)

Some of the slides for this lecture are adapted from slides by Clark Barrett

# Review

- Boolean satisfiability problem

$$\alpha \equiv (\neg x_1 \wedge x_2) \wedge (\neg x_2 \vee x_4) \wedge (\neg x_2 \wedge \neg x_3 \vee x_4)$$

- DPLL algorithm (backtracking + unit propagation + pureliteral assignment)

function DPLL( $\alpha$ )

Repeat

unit-propagate

pure-literal-assign

check-stopping-conditions

$\ell \leftarrow$  choose-literal( $\alpha$ );

return (DPLL( $\alpha \wedge \{\ell\}$ ) or DPLL( $\alpha \wedge \{\neg \ell\}$ ))

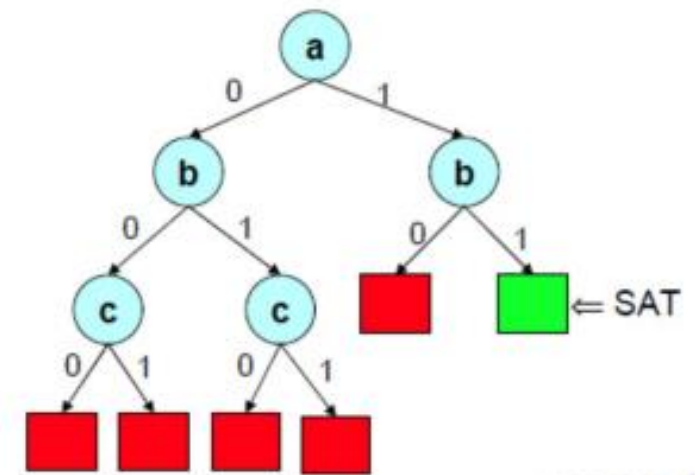


Image from Wikipedia

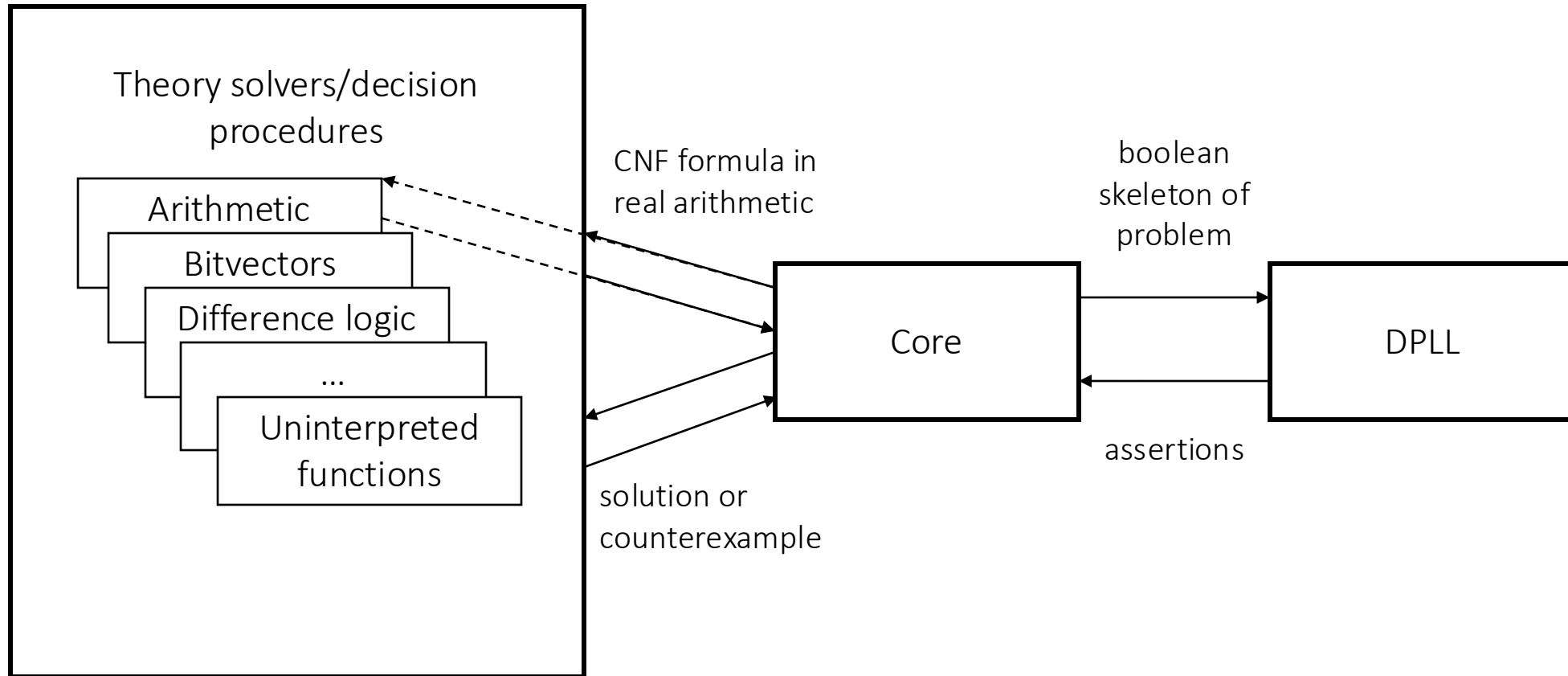
# Satisfiability modulo theories

- SAT: Given a *well-formed formula* in propositional logic, determine whether there exists a satisfying solution
- A *satisfiability modulo theory (SMT)* problem is a generalization of SAT in which some of the binary variables are replaced by predicates over a suitable set of non-binary variables
- $\phi_1(w, x, y, z) := (x - y = 5) \wedge (z - y \geq 2) \wedge (z - x > 2) \wedge (w - x = 2)$
- $\phi_2(x, y, z) := (3x^2 - 4y + 5z \leq 5) \wedge (-2x + 5z^3 \leq 7)$
- $\phi_1$  is a predicate in *difference logic* in which the variables are real-valued, and the clauses are constructed with standard comparison operations  $>$ ,  $\geq$ ,  $=$  and  $-$  (minus)
- $\phi_2$  is a predicate in real arithmetic

# What is a theory in mathematical logic?

- When we talk about **well-formed formulas with non-binary variables**, we have to say exactly what type of formulas are allowed
- and, what it means for assignments to *satisfy such formulas*
- This brings us to the notions **theory** and **models** in mathematical logic

# Architecture of an SMT solver



# Building up a theory

- First we define the syntax for writing formulas
  - A *signature*  $\Sigma = (\Sigma_F, \Sigma_P, V)$ 
    - set of *function symbols*  $\Sigma_F$ , e. g.,  $\{+, -, f, g, \sin, \dots\}$
    - set of *predicate symbols*  $\Sigma_P$
    - *arity* of each function:  $\text{arity}: \Sigma_F \rightarrow \mathbb{N}$
    - *0 arity* functions are constants
    - $V$ : set of *variables*
  - *Terms*  $(\Sigma, V)$ 
    - Elements of  $V$  are terms
    - If  $t_1, \dots, t_k \in \text{Terms}(\Sigma, V)$  and  $f \in \Sigma_F$  with arity  $k$ , then  $f(t_1, \dots, t_k) \in \text{Terms}(\Sigma, V)$
    - *Ground terms* are terms without variables
- $\Sigma_F = \{0, +\}, \Sigma_P = \{<\}$
  - $\text{arity}(0) = 0$
  - $\text{arity}(+) = 2$
  - $\text{arity}(<) = 2$
  - $V = \{x, y, z\}$
  - Terms defined by this signature are  $x, y, z, +(x, y), ++(x, y), 0, \dots$

# Terms to Formulas

- **Atomic formulas**  $AF$ 
    - True, False
    - If  $t_1, \dots, t_k \in Terms(\Sigma, V)$  and  $p \in \Sigma_p$  with arity  $k$ , then  $p(t_1, \dots, t_k) \in AF(\Sigma, V)$
    - A *literal* is an AF or its negation
    - Set of all atomic formulas  $AF(\Sigma, V)$
  - **Quantifier free formulas**  $QFF(\Sigma, V)$ 
    - $AF$
    - if  $\phi_1, \phi_2 \in QFF$  then
      - $\neg\phi_1 \in QFF, \phi_1 \wedge \phi_2 \in QFF, \phi_1 \vee \phi_2 \in QFF, \phi_1 \rightarrow \phi_2 \in QFF$
    - Set of all quantifier free formulas  $QFF(\Sigma, V)$
  - **First order formulas** is the set of quantifier free formulas under universal and existential quantifiers
    - **Bound variables** are those that are attached to quantifiers
    - **Free variables**: variables not bound
  - **Sentence**: First order formula with no free variables
  - **Theory** $(\Sigma, V)$  set of all sentences over  $(\Sigma, V)$
- $x < y$
  - $+(x, y) = +(y, x)$
  - $+(x, y) = 0 \wedge x > y$
  - $\forall x, \exists y: +(x, y) = 0$
  - $\forall x, \exists y: x < y$
  - $\forall x, \exists y: +(x, y) = x$
  - $\exists x: +(x, c) = x$

# Models for theories

This notion of model from mathematical logic is not to be confused with the notion of a model for a computational or physical process

- A *model* gives meanings or *interpretations* to formulas in theory  $T$
- A model  $M$  for  $T = \text{Theory}(\Sigma, V)$  has to define
  - A *domain*  $|M|$
  - interpretations of all functions and predicate symbols
  - $M(f): |M|^n \rightarrow |M|$  if  $\text{arity}(f) = n$
  - $M(p) \subseteq |M|^n$  if  $\text{arity}(p) = n$
  - Assignment  $M(x) \in |M|$  for every variable  $x \in V$
- A *formula*  $\phi$  is *true in*  $M$  if it evaluates to true under the given interpretations over domain  $M$

Example model for  $\Sigma = \{0, +, <\}$

$$|M| = \{a, b, c\}$$

$$M(0) = a$$

$$M(<) = \{\langle a, b \rangle, \langle a, c \rangle, \langle b, c \rangle\}$$

$M(+)$	$a$	$b$	<del><math>b</math></del>	□
$a$	$a$	$b$	$c$	
$b$	$b$	$c$	$a$	
$c$	$c$	$a$	$b$	

$$\text{if } M(x) = a, M(y) = b$$

$$\text{then } M(+ (x, y)) \text{ is } M(+)(M(x), M(y)) = M(+)(a, b) = b$$

$$M(+ (+ (x, y), y)) = c$$

$$M \models \forall x \exists y + (x, y) = x$$

We say that the model  $M$  *T-satisfies* the formula  $\phi$

# Example theories

- Uninterpreted functions (UF)  $\Sigma_F := \{f, g, \dots\}$ ,  $\Sigma_P := \{=\}$ ,  $V := \{x_i\}$ 
  - $x_1 = x_2 \wedge x_3 \neq x_2 \wedge f(x_3) \neq f(x_2)$
- Difference logic  $\Sigma_F := \{1, 2, \dots, -\}$ ,  $\Sigma_P := \{<, \leq, =, >, \geq\}$ 
  - $x_1 - x_2 \geq k$ , where  $\geq \in \{<, \leq, =, >, \geq\}$
- Linear arithmetic
  - $4x - 3y + 6z \leq 10$
- Real arithmetic (nonlinear)
  - $4x^2 + 6y - 9z^3 \leq 5$
- Bit vectors
- Arrays
  - $x'[i] = x[i] + 1$

# Decision procedures

Given a theory  $T$  a **theory solver or a decision procedure** for  $T$  takes as input a set of **literals**  $\phi$  (atomic propositions) and determines whether  $\phi$  is  **$T$ -satisfiable**, that is,

$\exists$  a model  $M$  such that  $M \models \phi$ ?

Note that  $\phi$  is not a general formula; it is restricted to be a set of **literals**

Literals in the theory of uninterpreted functions:  $x_1 = x_2$     $x_3 \neq x_2$     $f(x_3) \neq f(x_2)$

Not literals:  $(x_1 = x_2 \wedge x_3 \neq x_2)$     $(f(x_3) \neq f(x_2) \wedge g(x_3) \neq g(x_2))$

# Example decision procedure 2: Uninterpreted functions (UF)

$$\phi = x_1 = x_2 \wedge (x_2 = x_3) \wedge (x_4 = x_5) \wedge (x_5 \neq x_1) \wedge (F(x_1) \neq F(x_3))$$

## Decision procedure

1. Put all variables and function instances in their own classes
2. If  $t_1 = t_2$  is a literal then merge the classes containing them; do this repeatedly
3. If  $t_1$  and  $t_2$  are terms in the same class then merge classes containing  $F(t_1)$  and  $F(t_2)$ ; repeat
4. If  $t_1 \neq t_2$  is a literal in  $\phi$  and they belong to the same class then return unsat else return sat  $t_1$  and  $t_2$

# Example decision procedure 2: Uninterpreted functions (UF)

Initial classes  $\phi = x_1 = x_2 \wedge (x_2 = x_3) \wedge (x_4 = x_5) \wedge (x_5 \neq x_1) \wedge (F(x_1) \neq F(x_3))$

Classes  $\{x_1\} \{x_2\} \{x_3\} \{x_4\} \{x_5\} \{F(x_1)\} \{F(x_3)\}$

$\{x_1, x_2, x_3\} \{x_4, x_5\} \{F(x_1)\} \{F(x_3)\}$

$\{x_1, x_2, x_3\} \{x_4, x_5\} \{F(x_1), F(x_3)\}$

*Unsat*

# Example decision procedure 1: Difference logic

$$\phi = (x - y = 5) \wedge (z - y \geq 2) \wedge (z - x > 2) \wedge (w - x = 2) \wedge (z - w < 0)$$

Decision procedure:

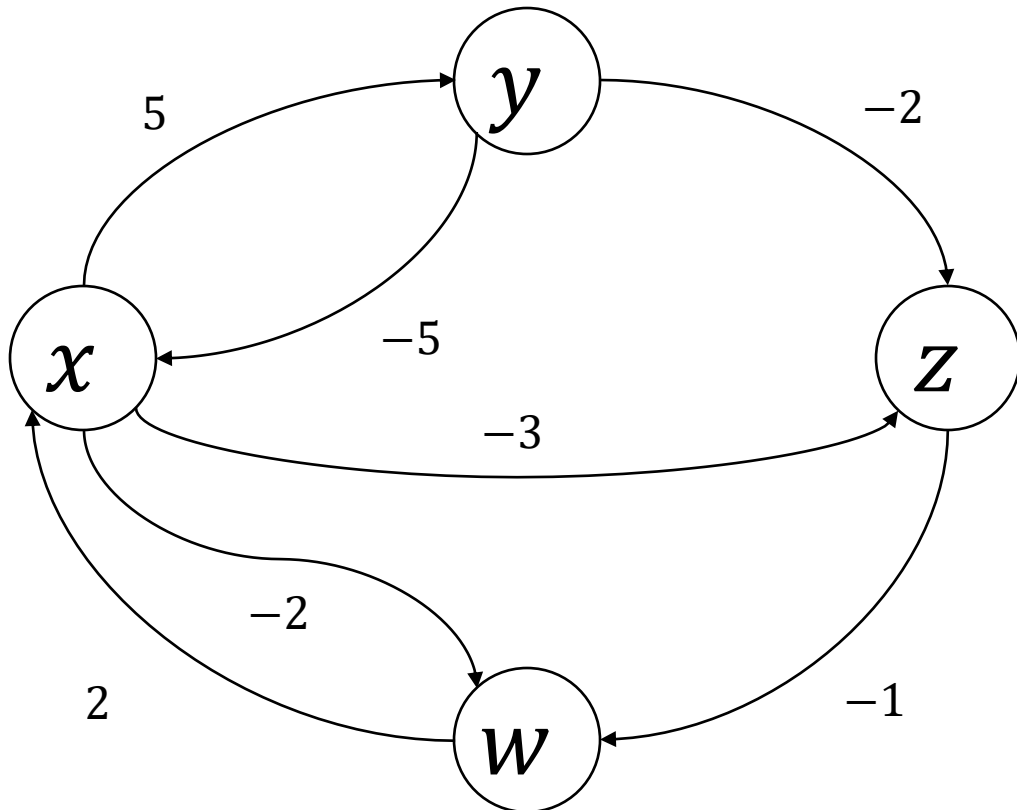
Convert each literal (AF) to  $x_1 - x_2 \leq c$  form:

$$\begin{aligned} \phi' = & (x - y \leq 5) \wedge (y - x \leq -5) \\ & \wedge (y - z \leq -2) \wedge \\ & (x - z \leq -3) \wedge \\ & (w - x \leq 2) \wedge (x - w \leq -2) \\ & (z - w \leq -1) \end{aligned}$$

For integer domain  $(x_1 - x_2 < k)$  is replaced by  $(x_1 - x_2 \leq k)$

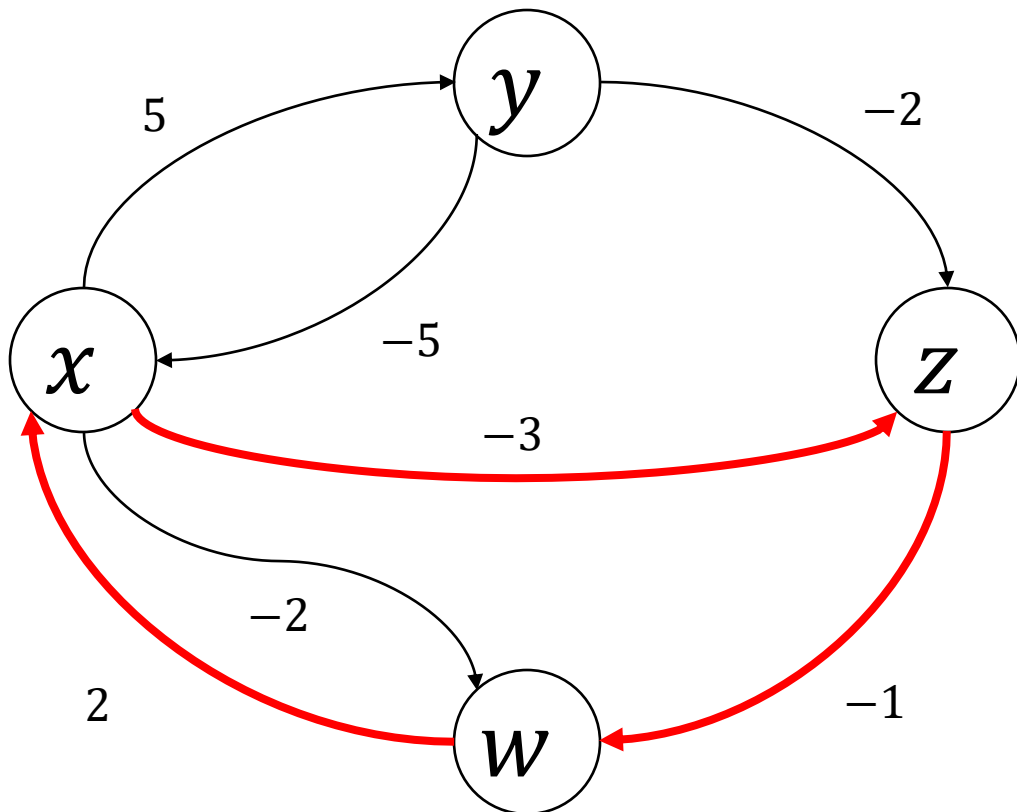
$$\begin{aligned} \phi' = & (x - y \leq 5) \wedge (y - x \leq -5) \\ & \wedge (y - z \leq -2) \wedge \\ & (x - z \leq -3) \wedge \\ & (w - x \leq 2) \wedge (x - w \leq -2) \\ & (z - w \leq -1) \end{aligned}$$

Construct a graph with edge from  $x \rightarrow^c y$  for each literal  $\phi'$

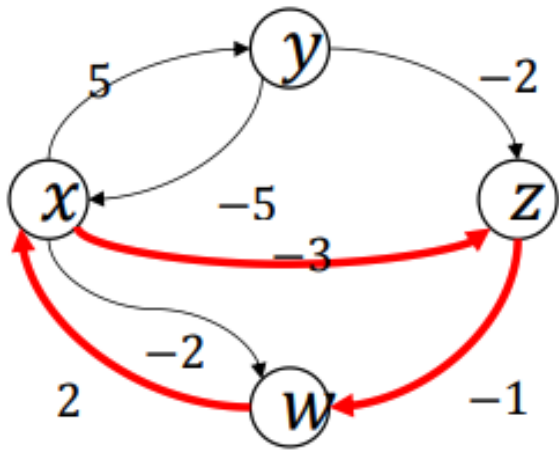


$$\begin{aligned} \phi' = & (x - y \leq 5) \wedge (y - x \leq -5) \\ & \wedge (y - z \leq -2) \wedge \\ & (x - z \leq -3) \wedge \\ & (w - x \leq 2) \wedge (x - w \leq -2) \\ & (z - w \leq -1) \end{aligned}$$

Construct a graph  $G_{\phi'}$ , with edge from  $x \rightarrow^c y$  for each  $x - y \leq c$  literal  $\phi'$



**Proposition.**  $\phi$  is satisfiable iff  $G_{\phi'}$  is negative cycle free.



**Proposition.**  $\phi$  is satisfiable iff  $G_\phi$  is negative cycle free.

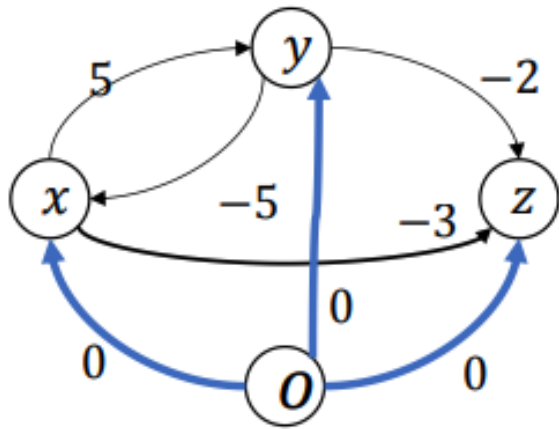
Proof. ( $\Leftarrow$ ) If there is a negative cycle then there exists

$$(x - z \leq -3); (z - w \leq -1); (w - x \leq 2)$$

adding all up:  $(0 \leq -2)$  which is inconsistent.

( $\Rightarrow$ ) Let us assume that there is no negative cycle. We will construct a satisfying solution  $\sigma: V \rightarrow \mathbb{Z}$

Consider additional vertex  $o$  with  $o \rightarrow^0 v$  edges for all  $v$



For each variable  $v$  define solution as the shortest distance from  $o$  to  $v$  (be aware of negative distances):  $\sigma(v) = -dist(o, v)$

Suppose FSOC,  $\sigma$  does not satisfy a literal  $x - y \leq k$  then

$$-dist(o, x) + dist(o, y) > k$$

$$dist(o, y) > k + dist(o, x)$$

$$dist(o, y) > dist(x, y) + dist(o, x) \text{ [because } x \rightarrow^k y \text{]}$$

violates triangle inequality definition of  $dist(o, y)$ !

$$\sigma(x) = -dist(o, x) = 5$$

$$\sigma(y) = -dist(o, y) = 0$$

$$\sigma(z) = -dist(o, z) = 8$$

# Decision Procedure for Difference Logic(DL)

Satisfiability check for difference procedure of DL can be performed using Bellman-Ford algorithm in time  $O(|V|.|E|)$

Inconsistency/unsatisfiability explanations are negative cycles

Amenable to incremental check

# DPLL<sup>T</sup>:DPLL modulo theories

How can we extend DPLL to handle formulas over other theories like

- Difference Logic (DL)
- Uninterpreted functions (UF)
- Linear Real Arithmetic (LRA)

Idea: Start with a Boolean abstraction (or skeleton) and incrementally add more theory information using decision procedure until we can conclusively say SAT or UNSAT

# DPLL<sup>T</sup>:DPLL modulo theories

- Abstract  $\phi$  to propositional form
- Feed to DPLL
- Use theory decision procedure to refine propositional formula to guide DPLL

UF formula  $\phi \equiv (a = b) \wedge (b = c) \wedge ((d = e) \vee (e \neq a)) \wedge (F(a) \neq F(c))$

abstract  $\phi \equiv x_1 \wedge x_2 \wedge (x_2 \vee \neg x_4) \wedge \neg x_5$

LRA formula  $F \equiv (x \leq 0 \vee x \leq 10) \wedge (\neg x \leq 0)$

abstract  $F^B \equiv (p \vee q) \wedge (\neg p)$

Abstraction because information is lost:  $x > 10 \Rightarrow x > 0$ , *i. e.*,  $\neg q \Rightarrow \neg p$  not in  $F^B$

Notation:  $(F^B)^T$  maps  $F^B$  back to theory T, *i.e.*,  $(F^B)^T = F$

**Proposition.** If  $F^B$  is UNSAT then  $F$  is UNSAT, but the converse does not hold.

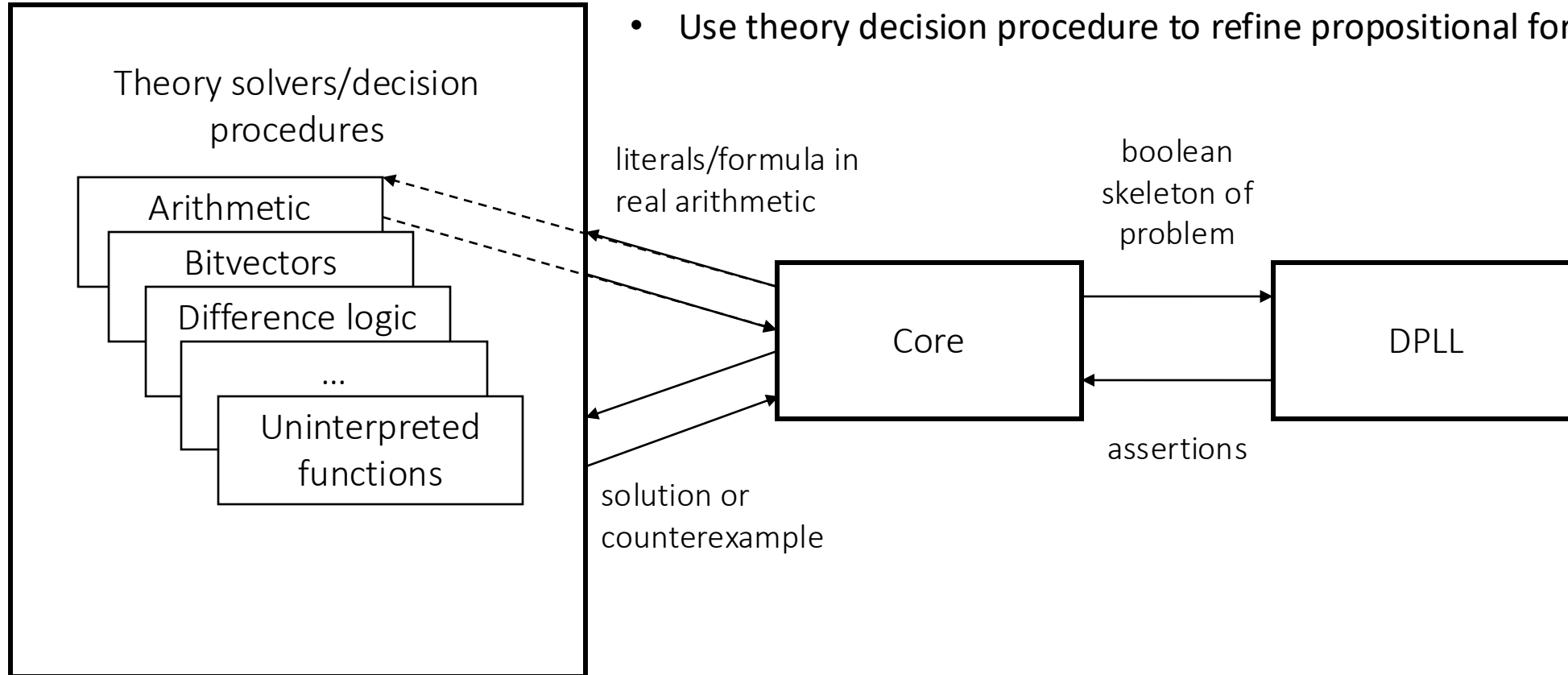
**Example.**  $F_1 \equiv (x \leq 0 \wedge x \geq 10)$  is UNSAT, however  $F^B \equiv p \wedge q$  is SAT

# Return to SMT

$$\phi \equiv g(a) = c \wedge f(g(a)) \neq f(c) \vee g(a) = d \wedge c \neq d$$

Several approaches, lazy approach:

- Abstract  $\phi$  to propositional form  $\phi \equiv x_1 \wedge (\neg x_2 \vee x_3) \wedge x_4$
- Feed to DPLL
- Use theory decision procedure to refine propositional formula a guide SAT



# DPLL for SMT

**Input:** A formula  $F$  in CNF form over theory  $T$

**Output:**  $I \models F$  or UNSAT

Let  $F^B$  be the abstraction of  $F$

**while true do**

**if** DPLL( $F^B$ ) is unsat **then return** UNSAT

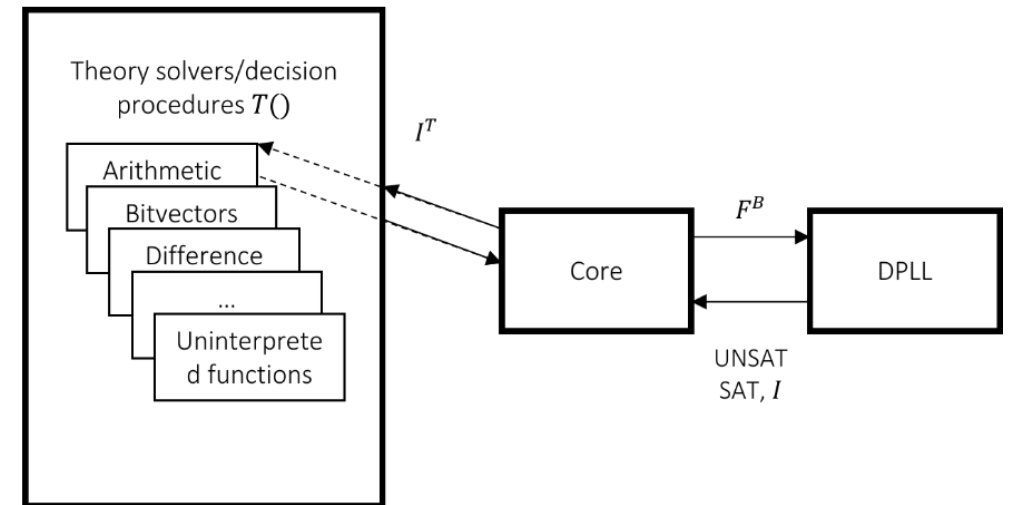
**else**

    Let  $I$  be the model returned by DPLL

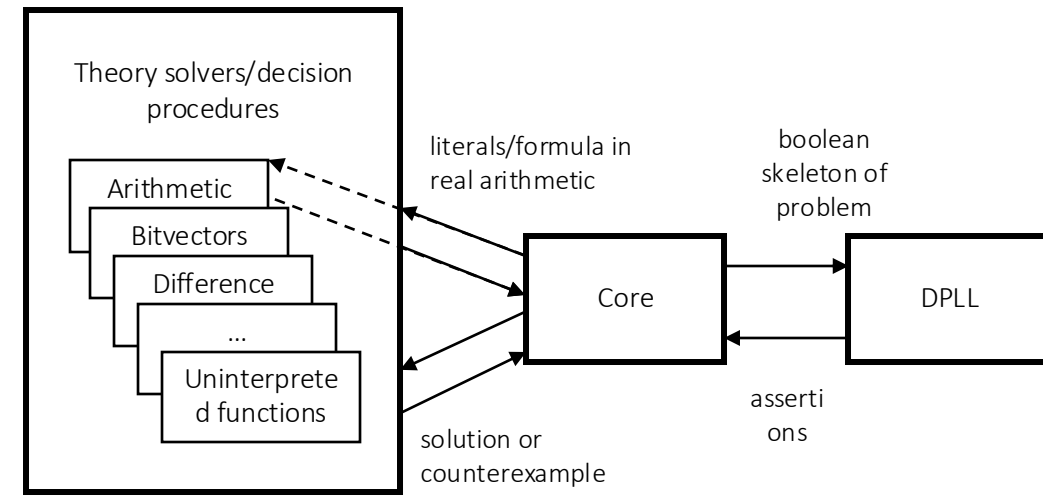
    Assume  $I$  is represented as a formula

**if**  $T(I^T)$  is sat **then return** SAT and the model returned by  $T()$

**else**  $F^B := F^B \wedge \neg I$



$$\phi \equiv \underbrace{g(a) = c}_1 \wedge \underbrace{f(g(a)) \neq f(c)}_{\bar{2}} \vee \underbrace{g(a) = d}_3 \wedge \underbrace{c \neq d}_{\bar{4}}$$



- send  $\{1, \bar{2} \vee 3, \bar{4}\}$  to DPLL
- returns model  $\{1, \bar{2}, \bar{4}\}$
- UF solver concretizes to  $g(a) = c, f(g(a)) \neq f(c), c \neq d$
- checks this as UNSAT
- send  $\{1, \bar{2} \vee 3, \bar{4}, \bar{1} \vee 2 \vee 4\}$  to DPLL
- returns model  $\{1, 2, 3, \bar{4}\}$
- UF solver concretizes and finds this to be UNSAT
- send  $\{1, \bar{2} \vee 3, \bar{4}, \bar{1} \vee 2 \vee 4, \bar{1} \vee \bar{2} \vee \bar{3} \vee 4\}$  to DPLL
- returns UNSAT