ECE/CS 584: Verification of Embedded and Cyberphysical Systems

Lecture 5: Satisfiability Modulo Theories (SMT) Concept and DPLL(T)

Prof. Huan Zhang

Office: CSL 262

huan@huan-zhang.com

Some slides adapted from Prof. Sayan Mitra's ECE584
Some of the slides for this lecture are adapted from slides by Prof. Clark Barrett

Assignment

- HW1 Due 2/17
- Keep thinking about class projects! Form teams (2 people)
- Next lecture: invited students who took this class in Spring 2024 will give presentations
  - Ask them about how they start their course project!

Previous lectures

- Boolean satisfiability problem

$$\alpha := (\neg x_1 \lor x_2) \land (\neg x_2 \lor x_4) \land (\neg x_2 \lor \neg x_3 \lor \neg x_4)$$

DPLL algorithm (backtracking + unit propagation + pure literal assignment)

```
function DPLL($\alpha$)
    unit-propagate
    pure-literal-assign        repeat
    check-stopping-conditions
    l ← choose-literal($\alpha$);
    return (DPLL($\alpha$ ∧ {l}) or
            DPLL($\alpha$ ∧ {¬l}));
```

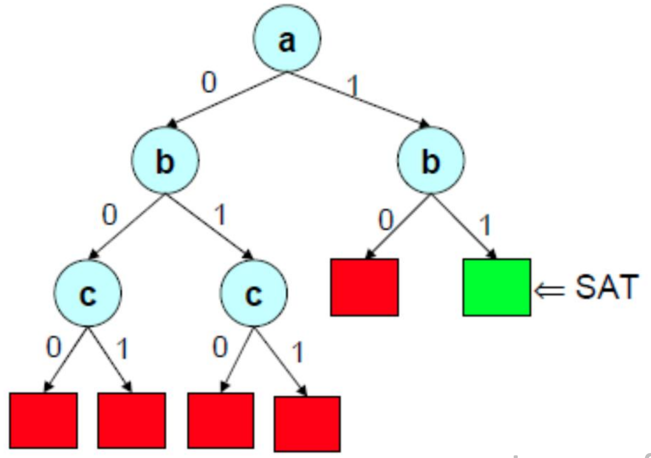


Image from Wikipedia

Today

- Satisfiability modulo theories (SMT): more general satisfiability problem
    - Theories, models, decision procedures
    - Uninterpreted Functions
    - Difference Logic

Satisfiability modulo theories

- SAT: Given a *well-formed formula* in propositional logic, determine whether there exists a satisfying solution

$$\alpha := (\neg x_1 \lor x_2) \land (\neg x_2 \lor x_4) \land (\neg x_2 \lor \neg x_3 \lor \neg x_4)$$

- 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**

$$\alpha := (sin(x_1) + x_2 \ \le 0) \land (x_2 + log(x_4) \ge 5.0) \land (x_2 x_3 \ge x_4)$$

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 some basic notions in mathematical logic
  - *theory* --- what does a well-formed formula look like ?
  - *models* --- what does it mean to satisfy a formula?

Building up a theory

First, we define the syntax for writing **formulas**

A *signature* $\Sigma = (\Sigma_F, \Sigma_P, V)$

- Σ_F : set of *function symbols*, $e.g.,\ \{+, -, f, g, sin, \ldots\}$
- Σ_P : set of *predicate symbols*, *e.g.*, {<, >, <=...}
  - *arity* of each function: $arity: \Sigma_F \to \mathbb{N}$
  - *0 arity* functions are constants
- V: set of *variables*

$Terms(\Sigma, V)$: combines variables and functions

- Elements of V are terms
- If $t_1, \ldots, t_k \in Terms(\Sigma, V)$ and $f \in \Sigma_F$ with arity k, then $f(t_1, \ldots, t_k) \in Terms(\Sigma, V)$
- *Ground terms* are terms without variables

- $\Sigma_F = \{0, +\}, \Sigma_P = \{<\}$
- $arity(0) = 0$
- $arity(+) = 2$
- $arity(<) = 2$
- $V = \{x, y, z\}$

- Terms defined by this signature are $x, y, z,\ +(x, y),\ +(+(x, y), 0), 0, \ldots$

Building up a theory: Terms to Formulas

- Atomic formulas AF: combines terms and predicates
  - True, False
  - If $t_1, \ldots, t_k \in Terms(\Sigma, V)$ and p $\in \Sigma_P$ with arity k, then $\mathrm{p}(t_1, \ldots, t_k) \in AF(\Sigma, V)$
  - 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
- $\mathrm{Theory}(\Sigma, V)$ set of all sentences over (Σ, V)

AF examples:

- $x < y$
- $+(x, y) =+(y, x)$

QFF examples:

- $+(x, y) = 0 \wedge x > y$

First order formulas:

- $\forall x,\ \exists y: +(x, y) = 0$
- $\forall x,\ \exists y: x < y$
- $\forall x,\ \exists y: +(x, y) = x$

Sentence:

- $\exists x: +(x, 1) = x$

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 $\mathrm{T} = \mathrm{Theory}(\Sigma, V)$ has to define
  - A domain $|\mathrm{M}|$
  - interpretations of all **functions** and **predicate** symbols
  - $M(f)\colon |M|^n \to |M|$ if $\mathrm{arity}(f) = n$
  - $M(p) \subseteq |M|^n$ if $arity(p) = n$
  - Assignment $M(x) \in |M|$ for every variable $x \in V$
- A formula ϕ is true in M if it evaluates to true under the given interpretations over domain M

Example

A *model* gives meanings or *interpretations* to formulas in theory T

Example model for $\Sigma = \{\{0,+\}, < , \{x, y\}\}$
$|M| = \{a, b, c\}$
$M(0) = a$
$M(<) = \{\langle a, b\rangle c, \langle a, c\rangle, \langle b, c\rangle\}$

$M(+)$	a	b	c
a	a	b	c
b	b	c	a
c	c	a	b

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

then $M(+(x, y))$ is $M(+)\big(M(x), M(y)\big) =$
$M(+)(a, b) = b$
$M(+(+\,(x, y), y)) = c$

Define formula $\phi{:} = \forall x\, \exists y + (x, y) = x$
$M \vDash \forall x\, \exists y + (x, y) = x$

We say that the model M *T-satisfies* the formula ϕ

Example theories

- (Real) Linear arithmetic
  - $4x - 3y + 6z \leq 10, x + y - z \leq 1;$
- Real nonlinear arithmetic
  - $4x^2 + 6y - 9z^3 \leq 5$
- Bit vectors
- Arrays
  - $x'[i] = x[i] + 1$
- Uninterpreted functions (UF) $\Sigma_F := \{f, g, \ldots\}$, $\Sigma_P := \{=, \neq\}$, $V := \{x_i\}$
  - $x_1 = x_2 \;\wedge\; x_3 \neq x_2 \;\wedge\; f(x_3) \neq f(x_2)$
- Difference logic $\Sigma_F := \{0,1,2,3, \ldots, -\}, \Sigma_P := \{<, \leq, =, >, \geq\}$, $V := \{x_i\}$
  - $x_1 - x_2 \gtreqless k$, where $\gtreqless \in \{<, \leq, =, >, \geq\}$

Decision procedures over a set of literals

Given a theory T a theory solver or a decision procedure for T takes as input a set of literals ϕ and determines whether ϕ is T-satisfiable, that is,

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

Note that ϕ is not the general form; it is restricted to a set of literals

$x_1 = x_2 \qquad x_3 \neq x_2 \qquad f(x_3) \neq f(x_2)$ are literals

$(x_1 = x_2 \ \wedge \ x_3 \neq x_2), \quad (f(x_3) \neq f(x_2) \wedge g(x_3) \neq g(x_2))$ are not literals

Example: Uninterpreted functions

Useful for abstractly reasoning about programs

- $\Sigma_F := \{f, g, \ldots\}$, $\Sigma_P := \{=\}$, $V := \{x_i\}$

Literals are of the form $x_1 = x_2 \;\; \wedge \;\; x_3 \neq x_2 \;\; \wedge \;\; f(x_3) \neq f(x_2)$

We know nothing about *f, g*, ... except for its name and arity

Decision procedure for 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 ϕ and they belong to the same class then return unsat else return sat

Decision procedure for Uninterpreted functions (UF)

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

Initial classes: all the variables, functions

Decision procedure for Uninterpreted functions (UF)

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

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

Decision procedure for Uninterpreted functions (UF)

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

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

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

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

$Unsat$

Difference Logic (conjunctive fragment)

A useful fragment of linear arithmetic
$\Sigma_F := \{1,2,..,-\ \}$
$\Sigma_P := \{< , \leq , \ = , \neq , > , \geq\}$

Literals are of the form $x_1 - x_2 \gtreqless k$, where $\gtreqless \in \{< , \leq , \ = , > , \geq\}$

x_1, x_2 are Integers or rational variables

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

Decision procedure: checking whether this formula is consistent

An Application: Job shop scheduling problem

Given a finite set of n jobs. Each job i of which consists of a chain of operations (m_1^i, d_1^i), (m_2^i, d_2^i),... There is a finite set of m machines $M = \{m_1, m_2, \ldots, m_m\}$, each of which can handle at most one operation at a time.

The problem of finding a shortest schedule---allocation of machine time to jobs---can be formulated in DL.

Decision procedure for 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:

Decision procedure for 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:

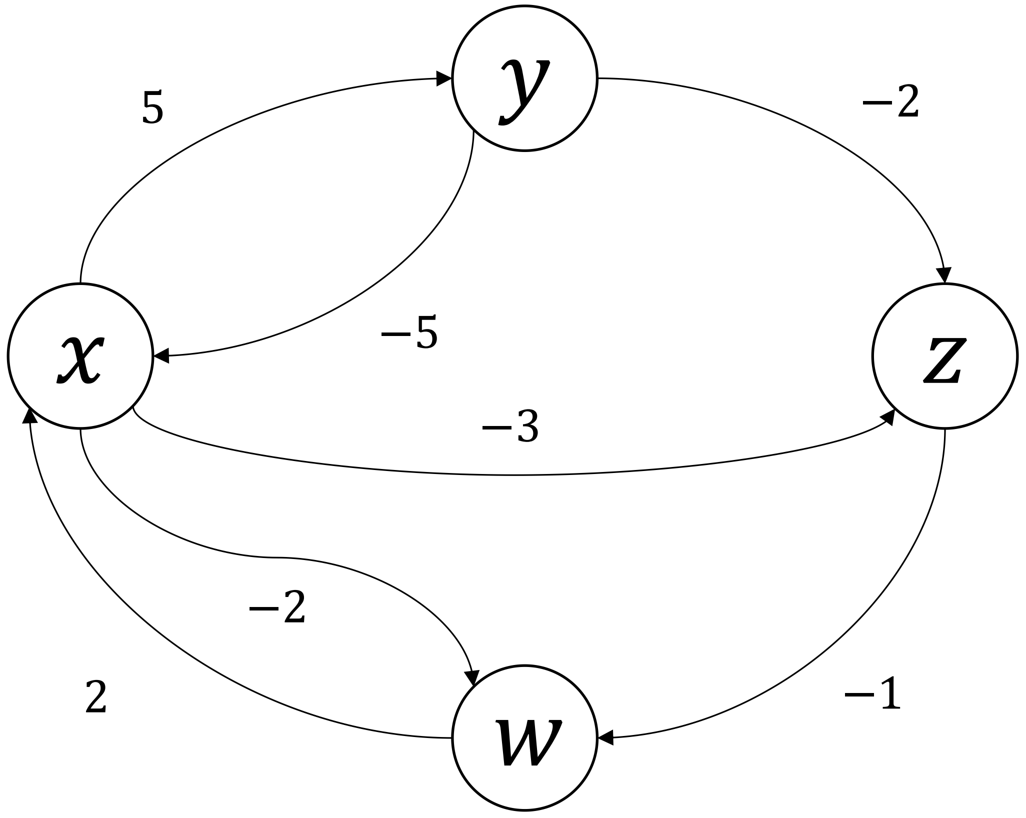
$$\begin{gathered}\phi' = (x - y \leq 5) \wedge (y - x \leq -5) \\ \wedge (y - z \leq -2) \wedge \\ (x - z \leq -3) \wedge \\ (w - x \leq 2) \wedge (x - w \leq -2) \\ (z - w \leq -1)\end{gathered}$$

For integer domain $(x_1 - x_2 < k)$ is replaced by $(x_1 - x_2 \leq k - 1)$

How to check satisfiability or consistency of formula ϕ'?

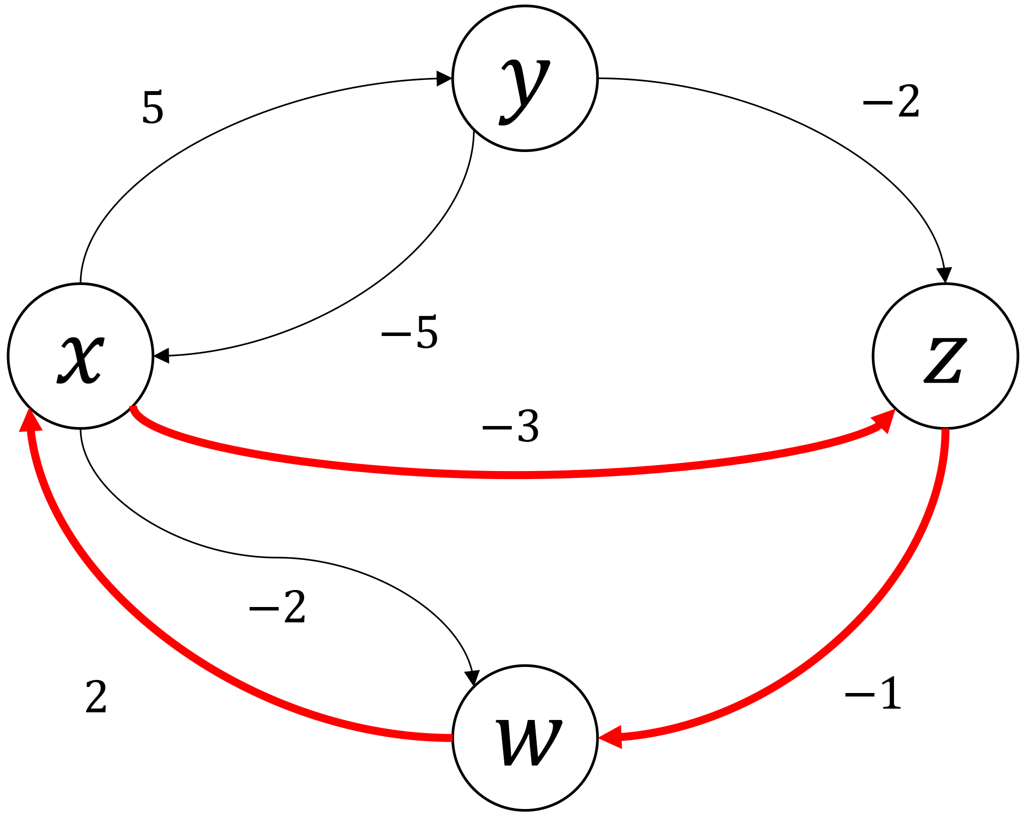
$$\phi' = (x - y \leq 5) \wedge (y - x \leq -5)$$
$$\wedge (y - z \leq -2) \wedge$$
$$(x - z \leq -3) \wedge$$
$$(w - x \leq 2) \wedge (x - w \leq -2)$$
$$(z - w \leq -1)$$

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


$$\begin{tikzcd}
 & y \arrow[ld, bend left, "-5"] \arrow[rd, bend left, "-2"] & \\
x \arrow[ru, bend left, "5"] \arrow[rr, bend right, "-3"] \arrow[rd, bend right, "-2"] & & z \arrow[ld, bend left, "-1"] \\
 & w \arrow[lu, bend left, "2"] &
\end{tikzcd}$$

$$\phi' = (x - y \leq 5) \wedge (y - x \leq -5)$$
$$\wedge\, (y - z \leq -2) \wedge$$
$$(x - z \leq -3) \wedge$$
$$(w - x \leq 2) \wedge (x - w \leq -2)$$
$$(z - w \leq -1)$$

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

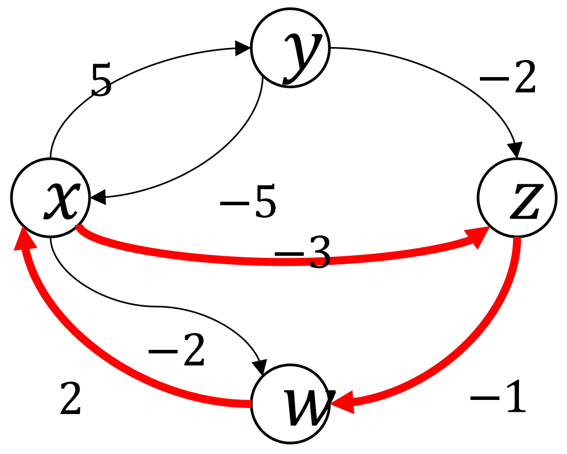


Proposition. ϕ is satisfiable iff $G_{\phi'}$ is negative cycle free.

Example: if there is a negative cycle then

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

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



Proposition. ϕ is satisfiable iff G_{ϕ}, is negative cycle free.

Proof. (<=) If there is a negative cycle then

$(x - z \leq -3)$; $(z - w \leq -1)$; $(w - x \leq 2)$ adding all up: $(0 \leq -2)$ which is inconsistent.

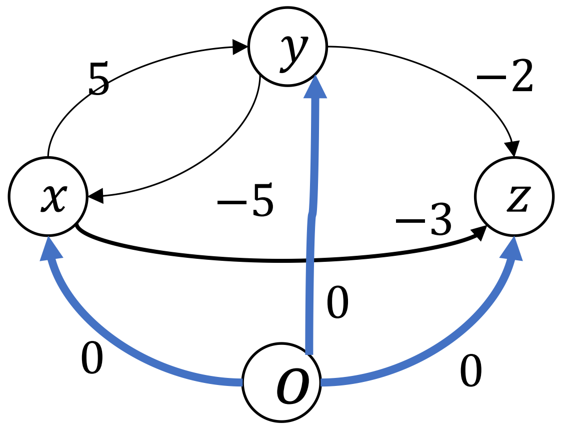
(=>) Let us assume that there is no negative cycle. We will construct a satisfying solution $\sigma: V \to \mathbb{Z}$

Consider additional vertex o with $o \to^{0} v$ 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, σ does not satisfy a literal $x - y \leq k$ then
$-dist(o,x) + dist(o,y) > k$
$dist(o,y) > k + dist(o,x)$
$dist(o,y) > dist(x,y) + dist(o,x)$

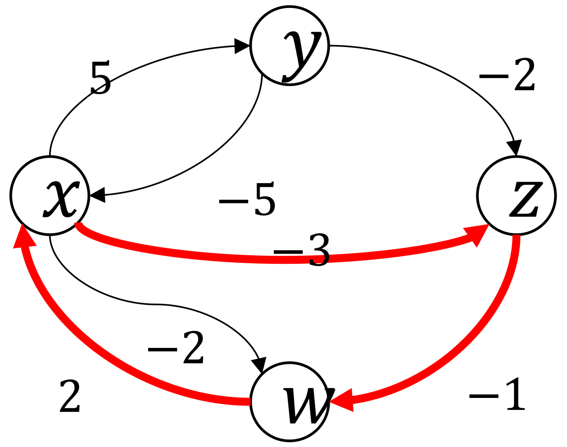
violates definition of $dist(o,y)$!



$\sigma(x) = -\,dist(o,x) = 5$
$\sigma(y) = -\,dist(o,y) = 0$
$\sigma(z) = -\,dist(o,z) = 8$

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 checks



DPLL$^{T:}$ 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^T: DPLL modulo theories

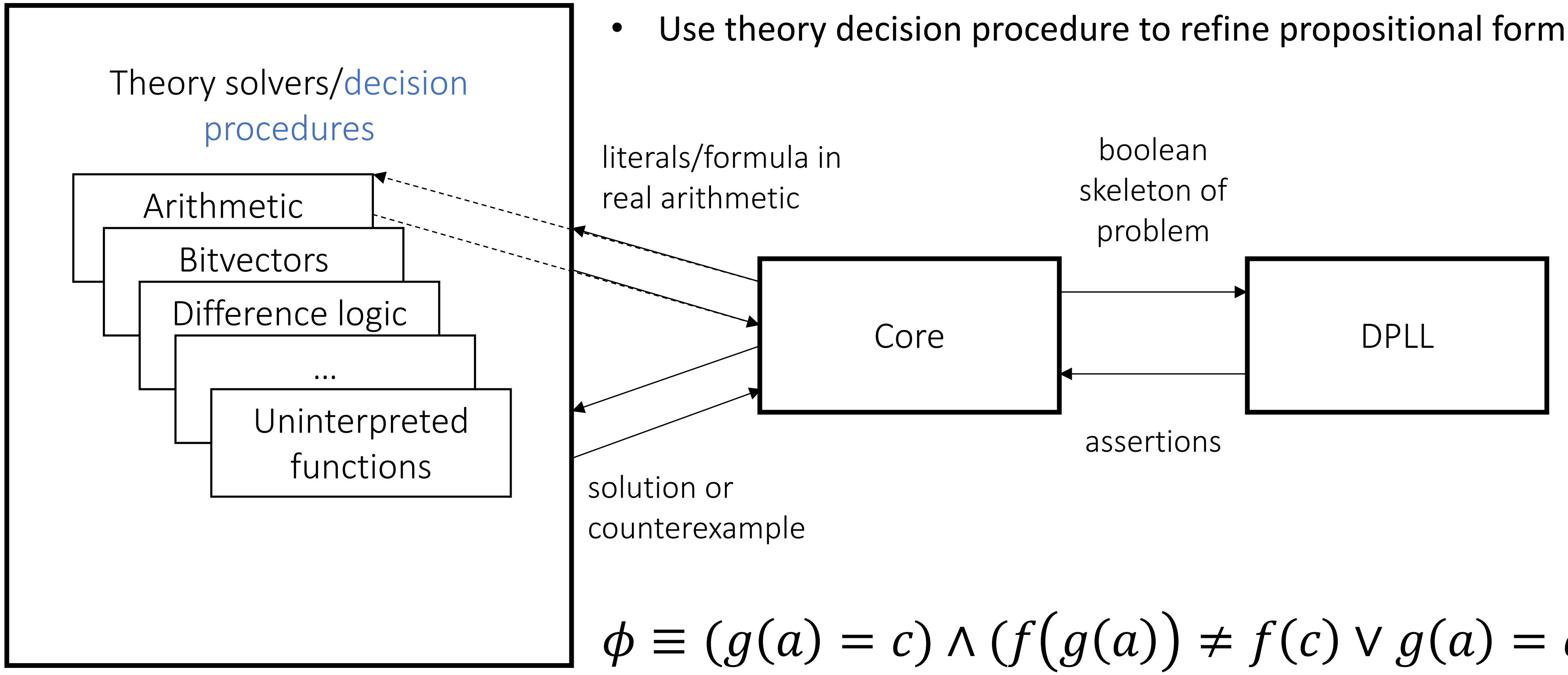
- Abstract ϕ to propositional form
- Feed to DPLL
- Use theory decision procedure to refine propositional formula to guide DPLL

$$\phi \equiv (g(a) = c) \land (f(g(a)) \neq f(c) \lor g(a) = d) \land c \neq d$$

$$\text{abstract } \phi \equiv x_1 \land (\neg x_2 \lor x_3) \land \neg x_4$$

How to solve SMT

- Abstract ϕ to propositional form
- Feed to DPLL
- Use theory decision procedure to refine propositional formula a guide SAT

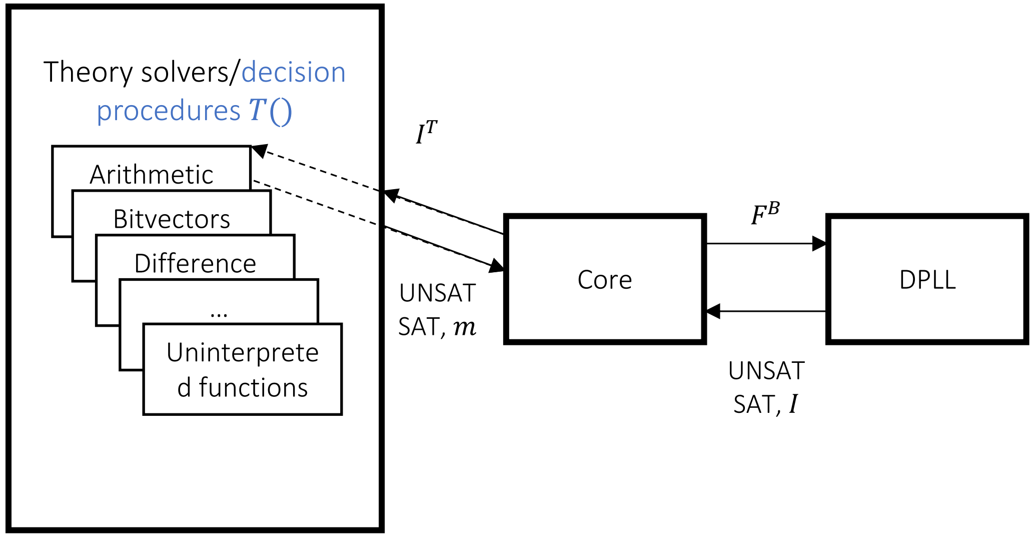

Theory solvers/decision procedures
Arithmetic
Bitvectors
Difference logic
...
Uninterpreted functions
literals/formula in real arithmetic
solution or counterexample
Core
boolean skeleton of problem
assertions
DPLL

$$\phi \equiv (g(a) = c) \land (f(g(a)) \neq f(c) \lor g(a) = d) \land c \neq d$$

$$\text{abstract } \phi \equiv x_1 \land (\neg x_2 \lor x_3) \land \neg x_4$$

Lazy DPLL^T Algorithm using a Decision Procedure $T()$

```
Input: A formula $F$ in CNF form over theory T
Output: $I \vDash F$ or UNSAT
Let $F^B$ be the abstraction of $F$
while true do
  if $\mathrm{DPLL}(F^B)$ 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 $F^B := F^B \wedge \neg I$
```



Example: DPLL$^{\text{LRA}}$ (Linear real arithmetic)

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

Boolean abstraction: replace every unique linear inequality with a Boolean variable

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

where p *abstracts* $x \leq 0$ and q *abstracts* $x \leq 10$

Example: DPLL$^{\text{LRA}}$

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

Boolean abstraction: replace every unique linear inequality with a Boolean variable

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

where p *abstracts* $x \leq 0$ and q *abstracts* $x \leq 10$

Abstraction because information is lost

The relationship $x > 10 \Rightarrow x > 0$, i.e., $\neg q \Rightarrow \neg p$ is lost in F_B

Example: DPLL$^{\text{LRA}}$

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

Boolean abstraction: replace every unique linear inequality with a Boolean variable
$F^B \equiv (p \vee q) \wedge (\neg p)$

where p *abstracts* $x \leq 0$ and q *abstracts* $x \leq 10$

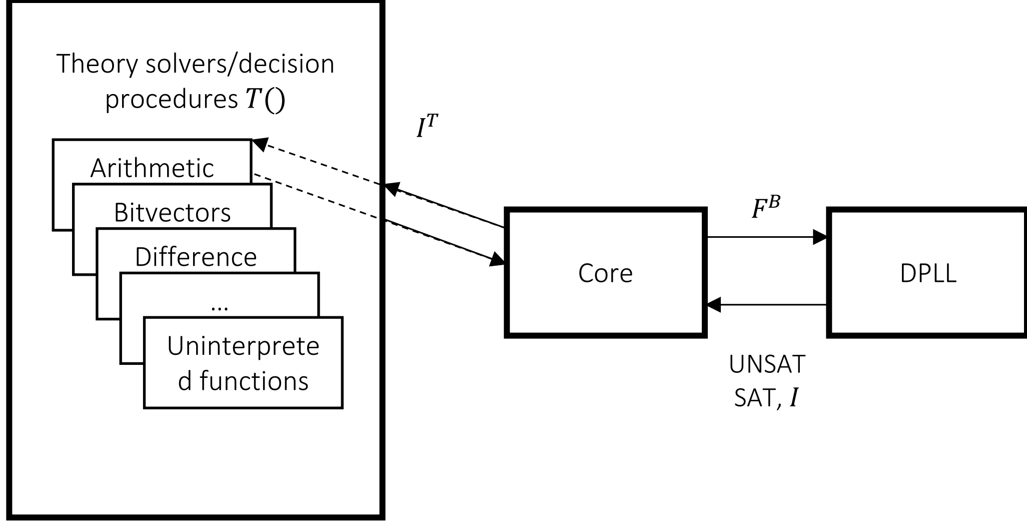
Abstraction because information is lost

The relationship $x > 10 \Rightarrow x > 0$, i.e., $\neg q \Rightarrow \neg p$ is lost 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, i.e., F^B is SAT does not mean that F is SAT.

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



- $\phi \equiv \underbrace{g(a) = c}_{1} \land (\underbrace{f(g(a)) \neq f(c)}_{\bar{2}} \lor \underbrace{g(a) = d}_{3}) \land \underbrace{c \neq d}_{\bar{4}}$
- abstract $\phi \equiv x_1 \land (\neg x_2 \lor x_3) \land \neg x_4$
- send $\phi^B \equiv \{1, \bar{2} \lor 3, \bar{4}\}$ to DPLL
- DPLL returns SAT with model I: $\{1, \bar{2}, \bar{4}\}$
- UF solver concretizes $I^{UF} \equiv g(a) = c, f(g(a)) \neq f(c), c \neq d$
- UF checks I^{UF} as UNSAT, we must exclude I in the abstraction
- send $\phi^B \land \neg I$: $\{1, \bar{2} \lor 3, \bar{4}, \bar{1} \lor 2 \lor 4\}$ to DPLL; this is a new fact learned by DPLL
- DPLL returns model I': $\{1, 2, 3, \bar{4}\}$
- UF solver concretizes I'^{UF} and finds this to be UNSAT
- send $\phi^B \land \neg I \land \neg I'$: $\{1, \bar{2} \lor 3, \bar{4}, \bar{1} \lor 2 \lor 4, \bar{1} \lor \bar{2} \lor \bar{3} \lor 4\}$ to DPLL; another fact
- returns UNSAT