# Lecture 7: Satisfiability modulo theories Part 2: DPLL(T) and Simplex Algorithm

Huan Zhang

huan@huan-zhang.com

Slides adapted from Prof. Sayan Mitra's slides in Fall 2021

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

George B. Dantzig

# Today

- SMT (cont)
- Decision procedure for Linear Real Arithmetic
  Simplex Algorithm [Dantzig 1947]

- Next 2 - 3 weeks: Verification of Neural Networks and Machine Learning

# Review: theories in SMT

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

# Review: 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

We discussed its **decision procedure** (based on putting equal variables/functions into classes)

# Review: Difference Logic

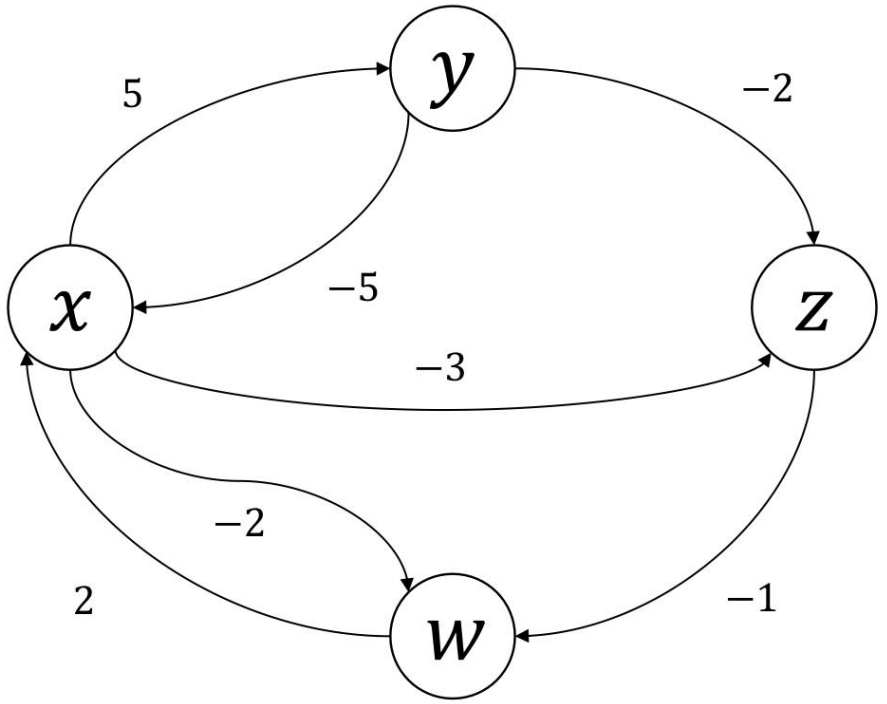
$\Sigma_F: = \{0,1,2,..,-\ \}$
$\Sigma_P: = \{<, \leq, \ = , \neq, >, \geq\}$

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

$x_1, x_2$ are Integers

We discussed the **decision procedure** (based checking negative loops on graph)

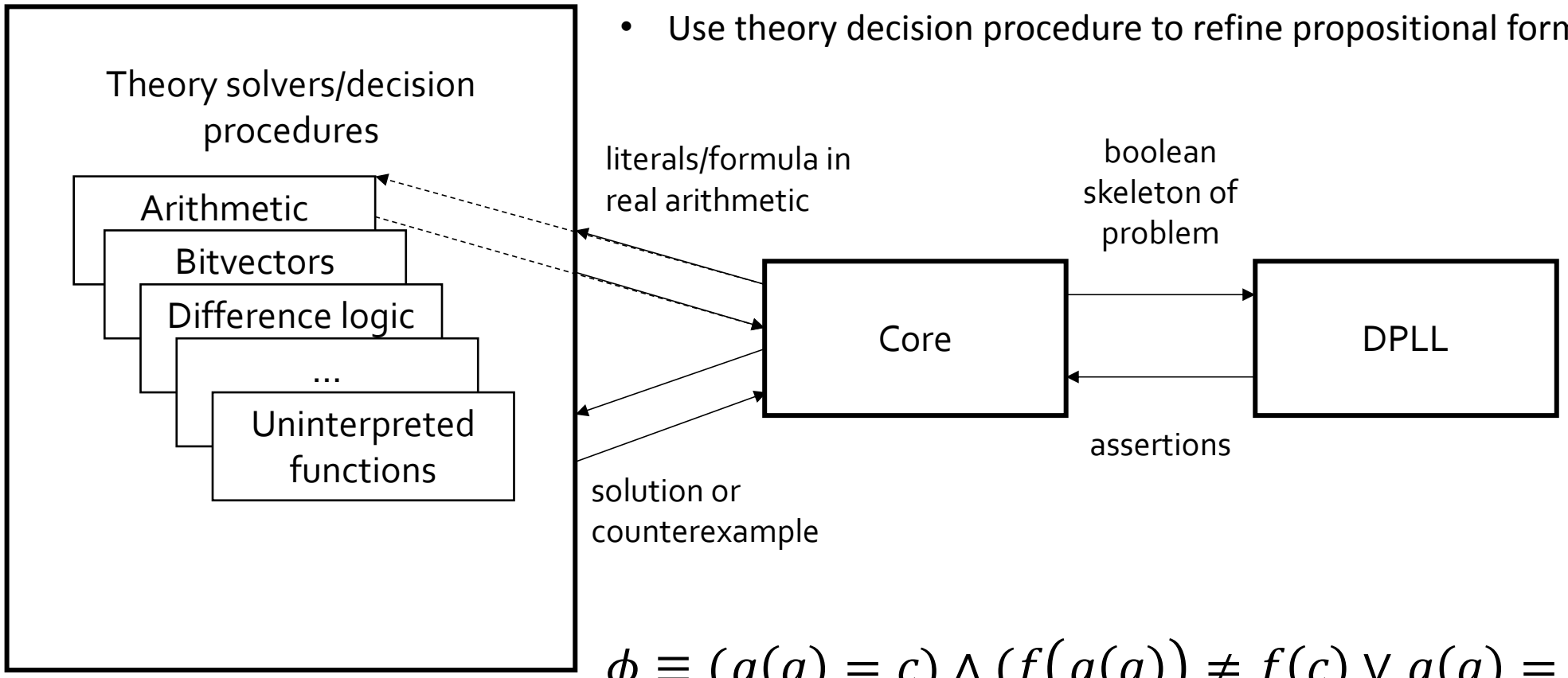
Example: $\phi = \ (x - y = 5) \wedge (z\ - y \geq 2) \wedge (z\ - x > 2) \wedge (w - x = 2) \wedge \ (z\ - w < 0)$



# How to solve SMT

Several approaches, lazy approach:

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




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

$$\text{abstract } \phi \equiv x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$$

# 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
until we can conclusively say SAT or UNSAT

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

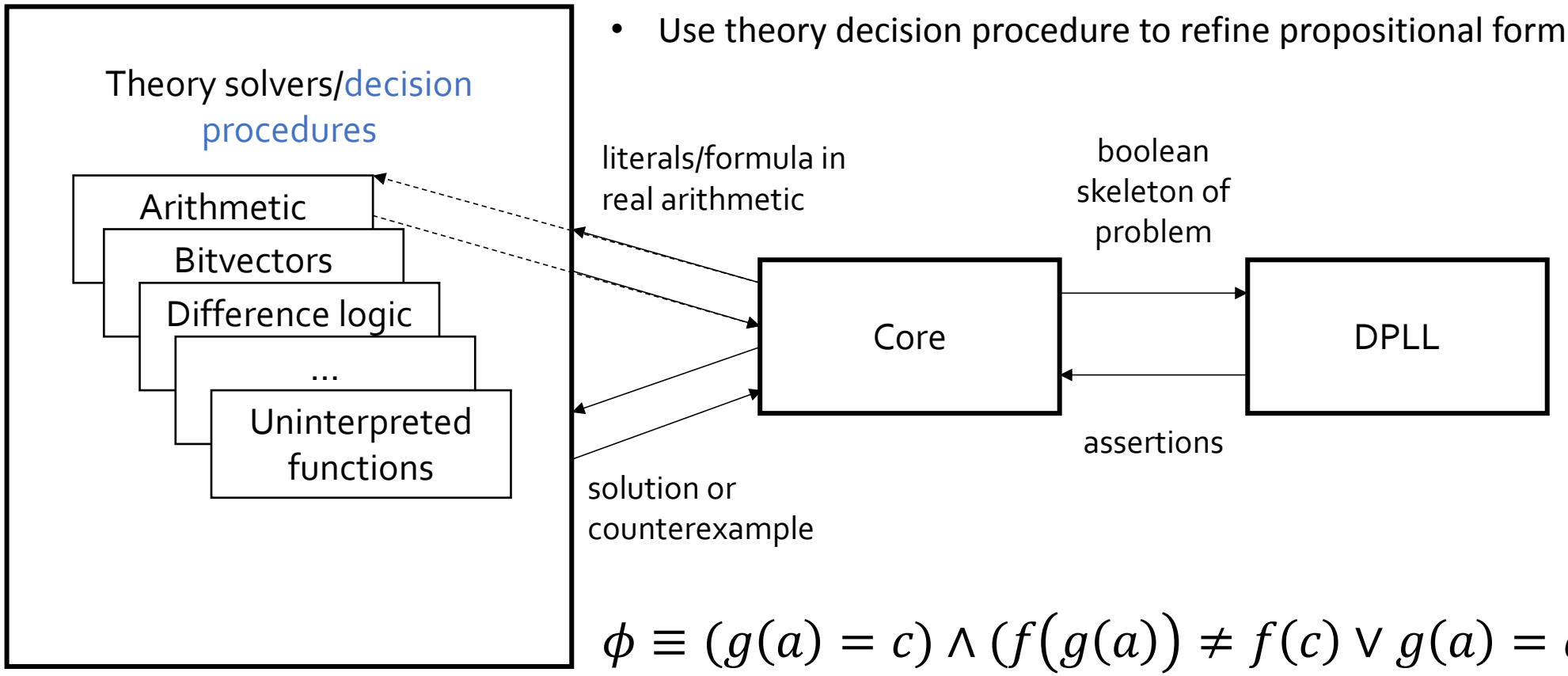
- Abstract $\phi$ to propositional form
- Feed to DPLL
- Use theory decision procedure to refine propositional formula to guide DPLL

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

$$\text{abstract } \phi \equiv x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$$

# How to solve SMT

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

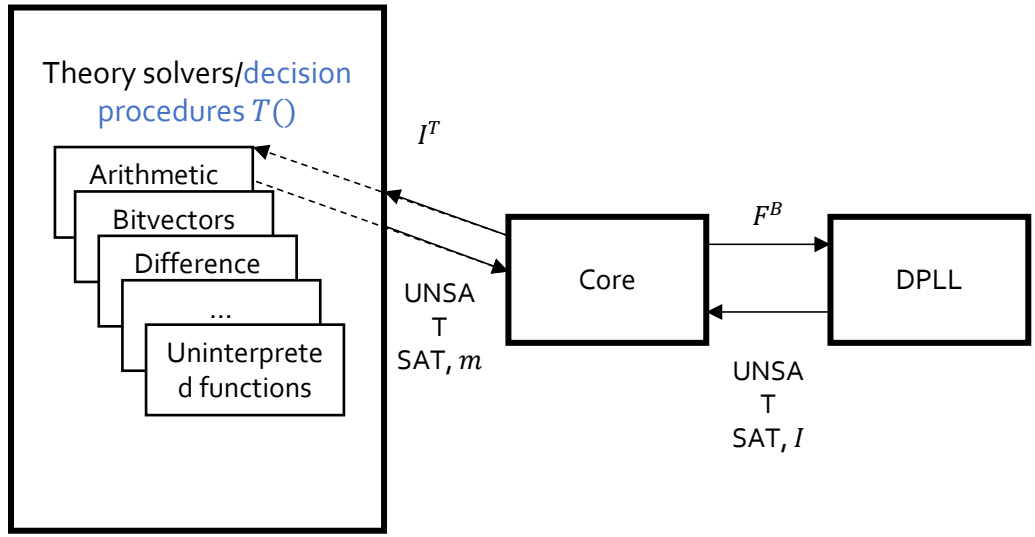



$$\phi \equiv (g(a) = c) \wedge (f(g(a)) \neq f(c) \vee g(a) = d) \wedge c \neq d$$

$$\text{abstract } \phi \equiv x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$$

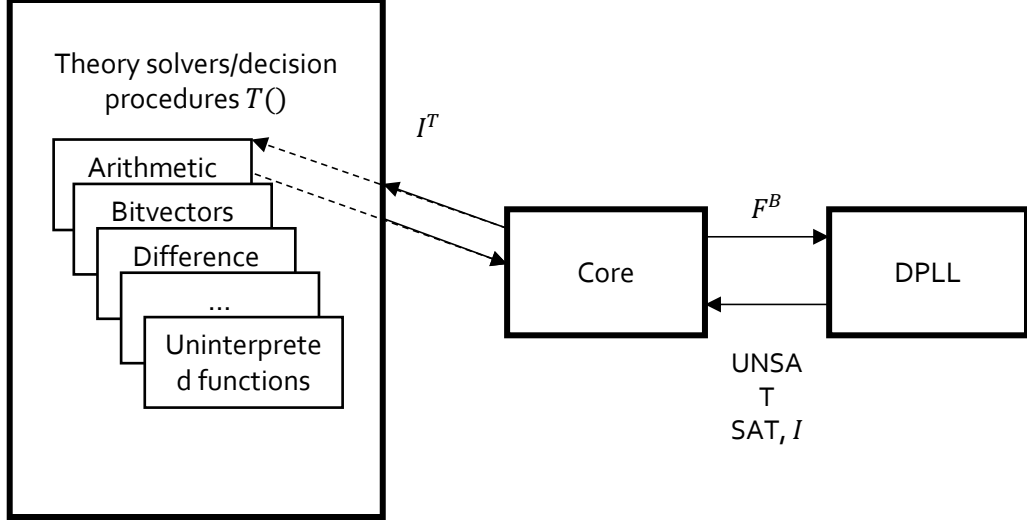
# DPLL$^T$ Algorithm using a Decision Procedure $T()$

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



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

  $1 \quad \bar{2} \quad 3 \quad \bar{4}$



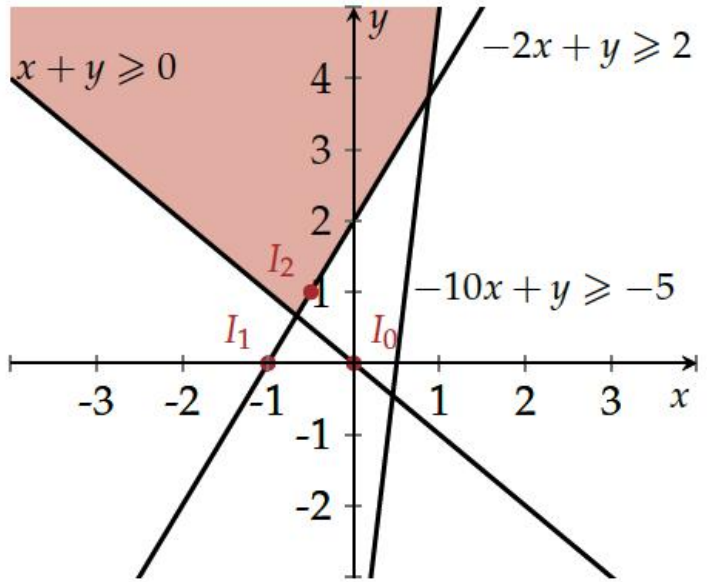

- abstract $\phi \equiv x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$
- send $\phi^B \equiv \{1, \bar{2} \vee 3, \bar{4}\}$ to DPLL
- DPLL returns SAT with model $I$:$\{1, \bar{2}, \bar{4}\}$
- UF solver concretizes $I^{UF} \equiv g(a) = c\,, f(g(a)) \neq f(c), c \neq d$
- UF checks $I^{UF}$ as UNSAT
- send $\phi^B \wedge \neg I$: $\{1, \bar{2} \vee 3, \bar{4}\,, \bar{1} \vee 2 \vee 4\}$ 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 $\phi^B \wedge \neg I \wedge \neg I'$: $\{1, \bar{2} \vee 3, \bar{4}\,, \bar{1} \vee 2 \vee 4, \bar{1} \vee \bar{2} \vee \bar{3} \vee 4\}$ to DPLL; another fact
- returns UNSAT

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

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

- Difference Logic (DL)
- Uninterpreted functions (UFs)
- **Today: Linear Real Arithmetic (LRA)**

$(x + y \geq 0) \wedge (-2x + y \geq 2) \wedge (-10x + y \geq -5)$



# Decision Procedure for Linear Real Arithmetic

Input: $F \equiv \wedge_{i=1}^{n} \Sigma_{j=1}^{m} c_{ij} x_j \leq b_i$ where $c_{ij}, b_i \in \mathbb{R}$

Output: $\exists$ a model $\boldsymbol{x} \in \mathbb{R}^m$ such that $\boldsymbol{x} \vDash F$?

Solution based on Simplex Algorithm [Dantzig 1947]

Simplex solves

$\max \Sigma_{j=1}^{m} a_j x_j$ subject to

$\wedge_{i=1}^{n} \Sigma_{j=1}^{m} c_{ij} x_j \leq b_i$

Our focus will be on finding any solution $\boldsymbol{x} \in \mathbb{R}^m$ that satisfies $F$

# Decision Procedure for Linear Real Arithmetic

Input: $F \equiv \wedge_{i=1}^{n} \Sigma_{j=1}^{m} c_{ij} x_j \leq b_i$ where $c_{ij}, b_i \in \mathbb{R}$

Output: $\exists$ a model $\boldsymbol{x} \in \mathbb{R}^m$ such that $\boldsymbol{x} \vDash F$?

Simplex expects $F$ to be expressed in the Simplex form, which is a **conjunction** of

- Linear equalities: $\Sigma_{i=1}^{m} c_i x_i = 0$
- Bounds: $\mathrm{l_i} \leq x_i \leq u_i$

# Transforming to Simplex Form

Consider the $i^{th}$ inequality in $F$: $\Sigma_{j=1}^{m} c_{ij} x_j \leq b_i$

Rewrite this as:
$s_i = \Sigma_{j=1}^{m} c_{ij} x_j \wedge s_i \leq b_i$

$s_i$ is called a *slack variable*

Putting together all the rewritten conjuncts we get $F_S$

**Proposition.**

1. Any model of $F_S$ is a model of $F$, disregarding the assignments to the slack variables.

2. If $F_S$ is UNSAT then $F$ is UNSAT.

# The decison procedure: simplex algorithm

Idea: Simultaneously try to find a model or a proof of UNSAT

Start with some *model (or valuation)* that satisfies all linear equalities (say, $x_i = 0,\ \forall i$), but may violate bounds

In each iteration, pick a bound that is not satisfied and modify the model to satisfy the bound
OR
discover that the formula is UNSAT

# Variable naming and ordering for Simplex

The input formula $F_S$ (after rewriting) has two types of variables

- **Basic variables** appear on the **LHS** of **one** equality; initially these are the *slack variables*
- **Non-basic variables** all others

$$
\begin{aligned}
s_1 &= x + y \\
s_2 &= -2x + y \\
s_3 &= -10x + y \\
s_1 &\geq 0 \\
s_2 &\geq 2 \\
s_3 &\geq -5 \\
-\infty &\leq x, y \leq \infty
\end{aligned}
$$

$$
\begin{aligned}
x &= -0.5s_2 + 0.5y \\
s_1 &= -0.5s_2 + 1.5y \\
s_3 &= 5s_2 - 4y \\
s_1 &\geq 0 \\
s_2 &\geq 2 \\
s_3 &\geq -5 \\
-\infty &\leq x, y \leq \infty
\end{aligned}
$$

# Variable naming and ordering for Simplex

The input formula $F_S$ (after rewriting) has two types of variables

- **Basic variables** appear on the **LHS** of **one** equality; initially these are the *slack variables*
- **Non-basic variables** all others

We fix an *arbitrary total ordering* on variables $x_1, \ldots, x_n$

For a basic variable $x_i$ and non-basic variable $x_j$ we denote by $c_{ij}$ the coefficient of $x_j$ in the definition of $x_i$, i.e.,
$x_i = \ \ldots + c_{ij}\, x_j + \ \ldots$

The upper and lower bounds of $x_i$ are called $u_i$ and $l_i$ (possibly $\infty$, $-\infty$)

# Pivoting: switch basic and non-basic variables

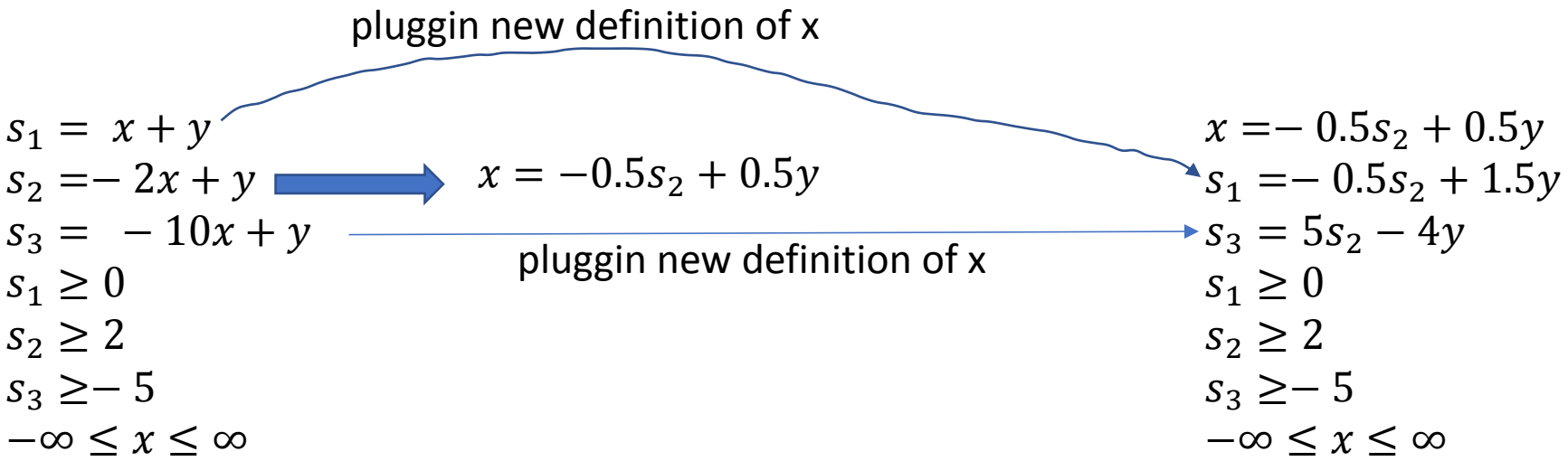
The pivoting operation change one non-basic variable to a basic variable (we say this variable is “entering”), while one other basic variable is changed to non-basic (we say this variable is “leaving”)

$$
\begin{aligned}
s_1 &= x + y \\
s_2 &= -2x + y \\
s_3 &= -10x + y \\
s_1 &\geq 0 \\
s_2 &\geq 2 \\
s_3 &\geq -5 \\
-\infty &\leq x, y \leq \infty
\end{aligned}
$$

$$
\begin{aligned}
x &= -0.5s_2 + 0.5y \\
s_1 &= -0.5s_2 + 1.5y \\
s_3 &= 5s_2 - 4y \\
s_1 &\geq 0 \\
s_2 &\geq 2 \\
s_3 &\geq -5 \\
-\infty &\leq x, y \leq \infty
\end{aligned}
$$

# Pivoting: switch basic and non-basic variables

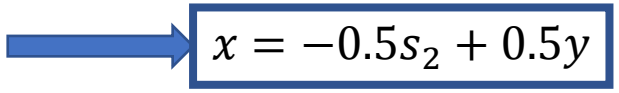
The pivoting operation change one non-basic variable to a basic variable (we say this variable is “entering”), while one other basic variable is changed to non-basic (we say this variable is “leaving”)



# Pivoting: switch basic and non-basic variables

The pivoting operation change one non-basic variable to a basic variable (we say this variable is “entering”), while one other basic variable is changed to non-basic (we say this variable is “leaving”)

$s_1 = x + y$
$s_2 = -2x + y$
$s_3 = -10x + y$
$s_1 \geq 0$
$s_2 \geq 2$
$s_3 \geq -5$
$-\infty \leq x \leq \infty$



$x = -0.5s_2 + 0.5y$

$x = -0.5s_2 + 0.5y$
$s_1 = -0.5s_2 + 1.5y$
$s_3 = 5s_2 - 4y$
$s_1 \geq 0$
$s_2 \geq 2$
$s_3 \geq -5$
$-\infty \leq x \leq \infty$

$x_i = \Sigma_{k \in N}^{m} c_{ik} x_k, j \in N$

Pivoting $x_i$ and $x_j$ rewrites $x_j$ as basic variable

$x_i = c_{ij} x_j + \Sigma_{k \in N \setminus \{j\}}^{m} c_{ik} x_k$

$$x_j = \frac{x_i}{c_{ij}} - \Sigma_{k \in N \setminus \{j\}}^{m} \frac{c_{ik}}{c_{ij}} x_k$$

# Simplex (Formal)

The algorithm maintains two invariants

(1). The model $\boldsymbol{x}$ always satisfies the **equalities**; bounds may be violated.

Why is this invariant satisfied by our initialization of all 0s?

(2). The **bounds** of all **non-basic** variables are all satisfied.

Why is this invariant satisfied by our initialization?

$F \equiv \Lambda_{i=1}^{n} \Sigma_{j=1}^{m} c_{ij} x_j \leq b_i$ where $c_{ij}, b_i \in \mathbb{R}$

Linear equalities: $\Sigma_{i=1}^{m} c_i x_i = 0$

Bounds: $\mathrm{l}_\mathrm{i} \leq x_i \leq u_i$ (only for slack variables)

# Transforming to Simplex Form

The algorithm maintains two invariants (1) and (2). Given the intialization:

$\Sigma_{j=1}^{m} c_{ij} x_j \leq b_i$

Rewrite this as:

$s_i = \Sigma_{j=1}^{m} c_{ij} x_j \wedge s_i \leq b_i$

$s_i$ is called a *slack variable*

(1) Setting all $x_j$ and $s_i$ to 0 satisfies all equality constraints (but may violating bounds on $s_i$)

(2) Initially, all slack variables are basic variables. Non-basic variables have no bounds.

# Simplex Algorithm: DP for LRA

**Input**: A formula $F_S$ in Simplex form
**Output**: $\boldsymbol{x} \vDash F_S$ or UNSAT
$\boldsymbol{x} \coloneqq \langle x_i \mapsto 0 \rangle$
**while** true **do**
**if** $\boldsymbol{x} \vDash F_S$ **then return** $\boldsymbol{x}$
Let $x_i$ be the first basic variable s.t. $x_i < \mathrm{l_i}$ or $x_i > \mathrm{u_i}$
**if** $x_i < \mathrm{l_i}$ **then**
Let $x_j$ be the first non-basic variable s.t.
$(x_j < u_j \wedge c_{ij} > 0) \vee (x_j > l_j \wedge c_{ij} < 0)$
**If** no such $x_j$ exists **then return** UNSAT
$x_j \coloneqq x_j + \frac{l_i - x_i}{c_{ij}}$ ; $x_i \coloneqq \mathrm{l}_i$
**else** Let $x_j$ be the first non-basic variable s.t.
$(x_j > l_j \wedge c_{ij} > 0) \vee (x_j < u_j \wedge c_{ij} < 0)$
**If** no such $x_j$ exists **then return** UNSAT
$x_j \coloneqq x_j + \frac{u_i - x_i}{c_{ij}}$ ; $x_i \coloneqq \mathrm{u}_i$
Pivot $x_i$ and $x_j$; update $x_i$, $x_j$, and all basic variables

$x_i$ currently violates its bounds

Finding a suitable variable we can "adjust" in order to satisfy the bound of $x_i$

"blue" conditions

Pivoting procedure: given
$x_i = \Sigma_{k \in N}^{m} c_{ik} x_k, j \in N$
Pivoting $x_i$ and $x_j$ rewrites $x_j$ as basic variable
$x_i = c_{ij} x_j + \Sigma_{k \in N \setminus \{j\}}^{m} c_{ik} x_k$
$x_j = \frac{x_i}{c_{ij}} - \Sigma_{k \in N \setminus \{j\}}^{m} \frac{c_{ik}}{c_{ij}} x_k$

# Example

$x + y \geq 0$
$-2x + y \geq 2$
$-10x + y \geq -5$

Rewritten in Simplex form
$s_1 = x + y$
$s_2 = -2x + y$
$s_3 = -10x + y$
$s_1 \geq 0$
$s_2 \geq 2$
$s_3 \geq -5$

# Example continued

$$s_1 = x + y$$
$$s_2 = -2x + y$$
$$s_3 = -10x + y$$
$$s_1 \geq 0$$
$$s_2 \geq 2$$
$$s_3 \geq -5$$
$$-\infty \leq x, y \leq \infty$$

Variable ordering
$x, y, s_1, s_2, s_3$

Initialization $\boldsymbol{x_0} = \langle x \mapsto 0, y \mapsto 0, s_1 \mapsto 0, s_2 \mapsto 0, s_3 \mapsto 0 \rangle$

$\boldsymbol{x_0}$ satisfies equalities, bounds of $s_1$ $s_3$ are satisfied

Pick the first variable $x$ to fix the bound of $s_2$

Since upper and lower bounds of $x$ are $\infty$ and $-\infty$ it easily satisfies the "blue" conditions

To increase $s_2$ to 2 and satisfy its lowerbound we decrease $x$ to -1

$$\boldsymbol{x_1} = \langle x \mapsto 0 + \frac{2 - 0}{-2} = -1, y \mapsto 0, s_1 \mapsto -1, s_2 \mapsto 2, s_3 \mapsto 10 \rangle$$

Pivot $s_2$ with $x$

Now $x$ becomes a basic variable, $s_2$ non-basic

$$x = -0.5s_2 + 0.5y$$
$$s_1 = -0.5s_2 + 1.5y$$
$$s_3 = 5s_2 - 4y$$
$$s_1 \geq 0$$
$$s_2 \geq 2$$
$$s_3 \geq -5$$
$$-\infty \leq x, y \leq \infty$$

# Example continued

$$x = -0.5s_2 + 0.5y$$
$$s_1 = -0.5s_2 + 1.5y$$
$$s_3 = 5s_2 - 4y$$
$$s_1 \geq 0$$
$$s_2 \geq 2$$
$$s_3 \geq -5$$
$$-\infty \leq x, y \leq \infty$$

$\boldsymbol{x_1} = \langle x \mapsto -1, y \mapsto 0, s_1 \mapsto -1, s_2 \mapsto 2, s_3 \mapsto 10 \rangle$

All equalities are still satisfied (invariant)

The only basic variable not satisfying its bounds is now $s_1$

The first non-basic variable we can tweak is $y$

Setting $s_1 \mapsto 0$ to satisfy the lowerbound of s1 we get

$$\boldsymbol{x_2} = \langle x \mapsto -2/3, y \mapsto 0 + \frac{0 - (-1)}{1.5} = 2/3, s_1 \mapsto 0, s_2 \mapsto 2, s_3 \mapsto 22/3 \rangle$$

Pivot $s_1$ with $y$

$$s_1 = -0.5s_2 + 1.5y \longrightarrow y = \frac{2}{3}s_1 + \frac{1}{3}s_2$$

$\boldsymbol{x_2} \vDash F_S$

$$y = \frac{2}{3}s_1 + \frac{1}{3}s_2$$
$$x = \frac{1}{3}s_1 - \frac{1}{3}s_2$$
$$s_3 = -\frac{8}{3}s_1 + \frac{11}{3}s_2$$
$$s_2 \geq 2$$
$$s_1 \geq 0$$
$$s_3 \geq -5$$
$$-\infty \leq x, y \leq \infty$$

# Why is simplex correct?

- Why does it terminate?

  Because we always looks for the first variable violating the bounds. There is a property (Bland's rule) that ensures that we never revisit the same set of basic and non-basic variables.

- Why does it give the right answer (sound)?
  - If it returns $\boldsymbol{x}$ does it satisfy $\boldsymbol{x} \vDash F$?

    This follows from the condition before **return** $\boldsymbol{x}$
  - If it returns UNSAT is $F$ really unsatisfiable?

For proofs, check Dutertre, B., de Moura, L.: Integrating Simplex with DPLL(T). Technical report, CSL-06-01, SRI International (2006)

# Unsatisfiable example

$s_1 = x + y$

$s_2 = -x - 2y$

$s_3 = -x + y$

$s_1 \geq 0$

$s_2 \geq 2$

$s_3 \geq 1$

Consider a Simplex execution in which there are two pivots:

Pivot 1: $s_1$ with $x$ , $s_1$set to 0.

$x = s_1 - y$

$s_2 = -s_1 - y$

$s_3 = -s_1 + 2y$

Pivot 2: $s_2$ with $y$ , $s_2$set to 2

$x = 2s_1 + s_2$

$y = -s_1 - s_2$

$s_3 = -3s_1 - 2s_2$

Non-basic variables satisfy their bounds (invariant), and so

$s_1 \geq 0, s_2 \geq 2$

If $s_3$ violates the bound then

$s_3 = -3s_1 - 2s_2 < 1$

We can make $s_3$ bigger by decreasing $s_1$ and $s_2$ but even at the extreme (smallest possible) $s_1$ and $s_2$

$s_3 = -3 \times 0 - 2 \times 2 = -4$

which is still less than 1 and Simplex concludes that the formula is UNSAT.

The “blue” conditions for choosing $x_j$ encodes this condition.

# Simplex Algorithm: DP for LRA

**Input**: A formula $F_S$ in Simplex form
**Output**: $\boldsymbol{x} \vDash F_S$ or UNSAT
$\boldsymbol{x} \coloneqq \langle x_i \mapsto 0 \rangle$
**while** true **do**
**if** $\boldsymbol{x} \vDash F_S$ **then return** $\boldsymbol{x}$
Let $x_i$ be the first basic variable s.t. $x_i < \mathrm{l_i}$ or $x_i > \mathrm{u_i}$
**if** $x_i < \mathrm{l_i}$ **then**
    Let $x_j$ be the first non-basic variable s.t.
    $(x_j < u_j \wedge c_{ij} > 0) \vee (x_j > l_j \wedge c_{ij} < 0)$
    **If** no such $x_j$ exists **then return** UNSAT
    $x_j \coloneqq x_j + \frac{l_i - x_i}{c_{ij}}$ ; $x_i \coloneqq \mathrm{l}_i$
**else** Let $x_j$ be the first non-basic variable s.t.
    $(x_j > l_j \wedge c_{ij} > 0) \vee (x_j < u_j \wedge c_{ij} < 0)$
    **If** no such $x_j$ exists **then return** UNSAT
    $x_j \coloneqq x_j + \frac{u_i - x_i}{c_{ij}}$ ; $x_i \coloneqq \mathrm{u}_i$
Pivot $x_i$ and $x_j$; update $x_i$, $x_j$, and all basic variables

i.e., find a non-basic variable that is **not** set to their extreme value $u_j$ or $l_j$ according to the sign of $c_{ij}$

If the non-basic variable $x_j$ has been set to its extreme, we cannot further change it to fix $x_i$

# Summary and Takeaways

- Satisfiability modulo theory solvers use theory solvers and DPLL to check satisfiability of formulas in other theories
  - DPLL takes care of disjunctions
  - Theory solvers take care of conjunctions

- Simplex or more generally Linear programming (LP) solvers is a theory solver for linear real arithmetic
  - Simplex algorithm solves LP by incrementally fixing the bounds of basic variables