Satisfiability Modulo Theories
Lezione 10 - Summary of the course and exercises

(slides revision: Saturday 14^{th} March, 2015, 11:46)

Roberto Bruttomesso

Seminario di Logica Matematica
(Corso Prof. Silvio Ghilardi)

19 Ottobre 2012

Outline

Solving SMT formulæ by reduction to SAT

Approaches to solve SMT formulæ are based on the observation that SMT can be **reduced** to SAT, i.e., the purely Boolean Satisfiability Problem
Consider for instance the $\mathcal{LIA}$ formula

$$\varphi \equiv (x - y \leq 0) \wedge (y - z \leq 0) \wedge ((z - x \leq -1) \vee (z - x \leq -2))$$

We may use a Boolean variable a to mean "$x - y \leq 0$" evaluates to $\top$ in the model.
Similarly we could use b, c, d for the other $\mathcal{T}$-atoms, obtaining

$$\psi \equiv a \wedge b \wedge (c \vee d)$$

However we are not done with the encoding ! In fact altough $\mu^{\mathbb{B}} \equiv \{a, b, c\}$ is a (Boolean) model for ψ, the correspondent set of $\mathcal{T}$-atoms $\{x - y \leq 0, y - z \leq 0, z - x \leq -1\}$ is not consistent in $\mathcal{LIA}$: we cannot extend $\mu^{\mathbb{B}}$ to a model μ that satisfies φ.
This information can be added to the encoding in the following form

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

Similarly we may derive **all** the remaining incompatibilities

$$\neg(a \wedge b \wedge d) \qquad \neg(\neg a \wedge \neg b \wedge \neg c) \qquad \neg(\neg a \wedge \neg b \wedge \neg d)$$

Solving SMT formulæ by reduction to SAT

Initial $\mathcal{LIA}$ formula

$$\varphi \equiv (x - y \leq 0) \wedge (y - z \leq 0) \wedge ((z - x \leq -1) \vee (z - x \leq -2))$$

Putting all the conditions together we get the Boolean formula

$$\psi \equiv \; a \wedge b \wedge (c \vee d) \;\wedge\; \neg(a \wedge b \wedge c) \;\wedge\; \neg(a \wedge b \wedge d) \;\wedge\; \neg(\neg a \wedge \neg b \wedge \neg c) \;\wedge\; \neg(\neg a \wedge \neg b \wedge \neg d)$$

Starting from φ we have

- (i) encoded the structure of φ
- (ii) **exhaustively** encoded all incompatible relations between $\mathcal{T}$-atoms

Theorem (Exercise 4 - correctness of the encoding)

φ is $\mathcal{T}$-satisfiable $\Leftrightarrow$ ψ is satisfiable, where ψ is obtained from φ with the steps (i)-(ii)

Exercise 4 - Proof

(a_i is the Boolean variable corresponding to a $\mathcal{T}$-atom P_i)

($\Rightarrow$)

If φ is $\mathcal{T}$-satisfiable, then it means that a model μ exists. A model $\mu^{\mathbb{B}}$ for ψ can be defined with $\mu^{\mathbb{B}}(a_i) = \mu(P_i)$.

($\Leftarrow$)

Suppose that ψ is satisfiable but φ is not. If so, then there is a model $\mu^{\mathbb{B}}$ (e.g., $\{\neg a_1, a_3\}$) such that its encoding (e.g., $(\neg P_1 \wedge P_3)$) represents an incompatible relation of $\mathcal{T}$-atoms. But if it is an incompatible relation, than its negation was added to ψ (e.g, $\neg(a_1 \wedge a_3)$), and it should not be satisfied by $\mu^{\mathbb{B}}$. Contradiction.

Exercize 2

Given the unsatisfiable formula

$$\psi \equiv \; a \wedge b \wedge (c \vee d) \;\wedge\; \neg(a \wedge b \wedge c) \;\wedge\; \neg(a \wedge b \wedge d) \;\wedge\; \neg(\neg a \wedge \neg b \wedge \neg c) \;\wedge\; \neg(\neg a \wedge \neg b \wedge \neg d)$$

show that $\neg(\neg a \wedge \neg b \wedge \neg c)$ and $\neg(\neg a \wedge \neg b \wedge \neg d)$ are redundant.

The last two clauses are redundant if

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

is already unsatisfiable on its own. In every model a and b must be assigned to $\top$. This simplifies the formula to

$$(c \vee d) \;\wedge\; \neg(c) \;\wedge\; \neg(d)$$

Now in every model c and d must be assigned to $\bot$. This simplifies the formula to

$$\bot$$

SAT-Solving

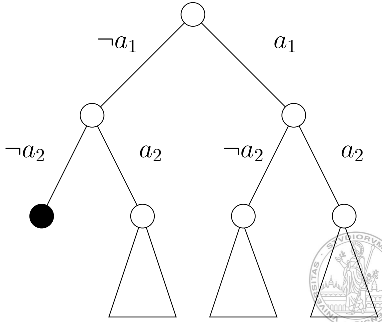
SAT-Solving is (the art of) finding the assignment satisfying all clauses

The assignment evolves **incrementally** by taking **Decisions**, starting from empty { }, but it can be **backtracked** if found wrong

The evolution of the assignment can be represented as a **tree**

$(\neg a_1 \vee a_3 \vee a_9)$
$(\neg a_2 \vee \neg a_3 \vee a_4)$
$(a_1 \vee a_2)$
$(\neg a_4 \vee a_5 \vee a_{10})$

$\{\ldots\}$

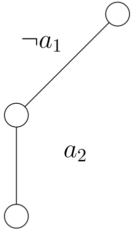


SAT-Solving

There are assignments which can be trivially driven towards the right direction. In the example below, given the current assignment, the third clause can be satisfied only by setting $a_2 \mapsto \top$

$$
\begin{array}{l}
(\neg a_1 \vee a_3 \vee a_9) \\
(\neg a_2 \vee \neg a_3 \vee a_4) \\
(a_1 \vee a_2) \\
(\neg a_4 \vee a_5 \vee a_{10})
\end{array}
$$

$\{\neg a_1, a_2\}$



This step is called **Boolean Constraint Propagation** (BCP). It represents a **forced** deduction. It triggers whenever all literals but one are assigned $\bot$

$$(\neg a_1 \vee a_2 \vee \neg a_3 \vee a_4 \vee \neg a_5)$$

SAT-Solving

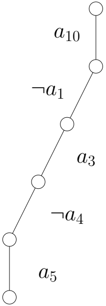
Decision-level: in an assignment, it is the number of decisions taken

Clearly, BCPs do not contribute to increase the decision level

When necessary, we will indicate decision level on top of literals $\overset{5}{a}$ in the assignment

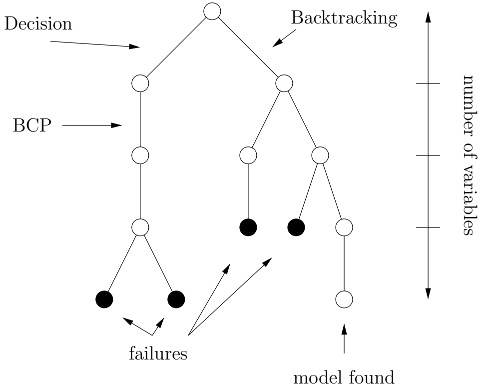
Example:

$$\{\overset{0}{a_{10}}, \overset{1}{\neg a_1}, \overset{2}{a_3}, \overset{3}{\neg a_4}, \overset{3}{a_5}\}$$



SAT-Solving

The process of finding the satisfying assignment is called **search**, and in its basic version it evolves with **Decisions**, **BCPs**, and **backtracking**



Consider the following scenario before and after BCP

$$\begin{array}{l} \ldots \\ (\neg a_{10} \vee \neg a_1 \vee a_4) \\ (a_3 \vee \neg a_1 \vee a_5) \\ (\neg a_4 \vee a_6) \\ (\neg a_5 \vee \neg a_6) \\ \ldots \end{array} \qquad \begin{array}{l} \ldots \\ (\neg a_{10} \vee \neg a_1 \vee a_4) \\ (a_3 \vee \neg a_1 \vee a_5) \\ (\neg a_4 \vee a_6) \\ (\neg a_5 \vee \neg a_6) \\ \ldots \end{array}$$

$$\{\overset{0}{a_{10}}, \overset{1}{\neg a_3}, \overset{2}{a_7}, \overset{3}{\neg a_2}, \overset{4}{a_1}\} \qquad \{\overset{0}{a_{10}}, \overset{1}{\neg a_3}, \overset{2}{a_7}, \overset{3}{\neg a_2}, \overset{4}{a_1}, \overset{4}{a_4}, \overset{4}{a_5}, \overset{4}{a_6}\}$$

BCP leads to a conflict, but, what is the **reason** for it ? Do a_7 and a_2 play any role ?

Does it make sense to consider the assignments (which backtracking would produce) ?
$\{\overset{0}{a_{10}}, \overset{1}{\neg a_3}, \overset{2}{\neg a_7}, \overset{3}{\neg a_2}, \overset{4}{a_1}\}$ — $\{\overset{0}{a_{10}}, \overset{1}{\neg a_3}, \overset{2}{a_7}, \overset{3}{\neg a_2}, \overset{4}{a_1}\}$ — $\{\overset{0}{a_{10}}, \overset{1}{\neg a_3}, \overset{2}{a_7}, \overset{3}{a_2}, \overset{4}{a_1}\}$

No, because *whenever* a_{10}*,* $\neg a_3$ *is assigned, then* a_1 *must not be set to* $\top$. This translates to an additional clause, which can be *learnt*, i.e., it can be added to the formula

$$(\neg a_{10} \vee a_3 \vee \neg a_1)$$

SAT-Solving

In practice, we use **resolution** steps to compute the learnt clause:

- we start from the clause with all literals to $\bot$ (conflicting clause)
- iteratively, we take the clause that propagated the last literal on the trail and we apply resolution
- we stop when only one literal from the current decision level is left in the clause

$$\dfrac{\dfrac{\dfrac{(\neg a_5 \lor \neg a_6) \quad (\neg a_4 \lor a_6)}{(\neg a_5 \lor \neg a_4)} \quad (a_3 \lor \neg a_1 \lor a_5)}{(\neg a_4 \lor a_3 \lor \neg a_1)} \quad (\neg a_{10} \lor \neg a_1 \lor a_4)}{(\neg a_{10} \lor a_3 \lor \neg a_1)}$$

Trail	dl	Reason
a_{10}	0	(a_{10})
$\neg a_3$	1	Decision
a_7	2	Decision
$\neg a_2$	3	Decision
a_1	4	Decision
a_4	4	$(\neg a_{10} \lor \neg a_1 \lor a_4)$
a_5	4	$(a_3 \lor \neg a_1 \lor a_5)$
a_6	4	$(\neg a_4 \lor a_6)$

$$\{\overset{0}{a_{10}}, \overset{1}{\neg a_3}, \overset{2}{a_7}, \overset{3}{\neg a_2}, \overset{4}{a_1}, \overset{4}{a_4}, \overset{4}{a_5}, \overset{4}{a_6}\}$$

We say that $(\neg a_{10} \lor a_3 \lor \neg a_1)$ is the **conflict clause** and it is the one we learn

Consider the following clause set and assignment

$$
\begin{array}{l}
(\neg a_1 \vee a_2) \\
(\neg a_1 \vee a_3 \vee a_9) \\
(\neg a_2 \vee \neg a_3 \vee a_4) \\
(\neg a_4 \vee a_5 \vee a_{10}) \\
(\neg a_4 \vee a_6 \vee a_{11}) \\
(\neg a_5 \vee \neg a_6) \\
(a_1 \vee a_7 \vee \neg a_{12}) \\
(a_1 \vee a_8) \\
(\neg a_7 \vee \neg a_8 \vee \neg a_{13}) \\
\ldots
\end{array}
$$

$$\{\overset{1}{\neg a_9}, \overset{2}{a_{12}}, \overset{2}{a_{13}}, \overset{3}{\neg a_{10}}, \overset{3}{\neg a_{11}}, \ldots, \overset{6}{a_1}\}$$

- Find the correct conflict clause
- Find the correct decision level to backtrack

Lecture 3 - Exercize 1

Trail	dl	Reason
$\neg a_9$	1	Decision
a_{12}	2	Decision
a_{13}	2	(some clause)
$\neg a_{10}$	3	Decision
$\neg a_{11}$	3	(some clause)
$\dots$		
a_1	6	Decision
a_2	6	$(\neg a_1 \vee a_2)$
a_3	6	$(\neg a_1 \vee a_3 \vee a_9)$
a_4	6	$(\neg a_2 \vee \neg a_3 \vee a_4)$
a_5	6	$(\neg a_4 \vee a_5 \vee a_{10})$
a_6	6	$(\neg a_4 \vee a_6 \vee a_{11})$

$(\neg a_1 \vee a_2)$
$(\neg a_1 \vee a_3 \vee a_9)$
$(\neg a_2 \vee \neg a_3 \vee a_4)$
$(\neg a_4 \vee a_5 \vee a_{10})$
$(\neg a_4 \vee a_6 \vee a_{11})$
$(\neg a_5 \vee \neg a_6)$
$(a_1 \vee a_7 \vee \neg a_{12})$
$(a_1 \vee a_8)$
$(\neg a_7 \vee \neg a_8 \vee \neg a_{13})$

$$\{\overset{1}{\neg a_9}, \overset{2}{a_{12}}, \overset{2}{a_{13}}, \overset{3}{\neg a_{10}}, \overset{3}{\neg a_{11}}, \dots, \overset{6}{a_1}\}$$

$$\frac{(\neg a_5 \vee \neg a_6) \qquad (\neg a_4 \vee a_6 \vee a_{11})}{(\neg a_5 \vee \neg a_4 \vee a_{11})}$$

$$\frac{(\neg a_5 \vee \neg a_4 \vee a_{11}) \qquad (\neg a_4 \vee a_5 \vee a_{10})}{(\neg a_4 \vee a_{10} \vee a_{11})}$$

$$\frac{(\neg a_4 \vee a_{10} \vee a_{11}) \qquad (\neg a_2 \vee a_3 \vee a_4)}{(\neg a_2 \vee \neg a_3 \vee a_{10} \vee a_{11})}$$

$$\frac{(\neg a_2 \vee \neg a_3 \vee a_{10} \vee a_{11}) \qquad (\neg a_2 \vee a_3 \vee a_9)}{(\neg a_2 \vee a_9 \vee a_{10} \vee a_{11})}$$

$$\frac{(\neg a_2 \vee a_9 \vee a_{10} \vee a_{11}) \qquad (\neg a_1 \vee a_2)}{(\neg a_1 \vee a_9 \vee a_{10} \vee a_{11})}$$

Conflict clause: $(\neg a_1 \vee a_9 \vee a_{10} \vee a_{11})$
Backtracking level: 3

The Lazy Approach

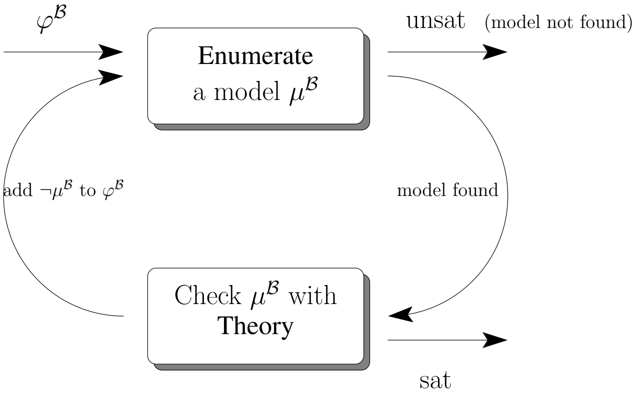
Notice that

- Assignments μ of φ are many (potentially ∞), infeasible to check if any of them is a model **systematically**
- Models $\mu^{\mathcal{B}}$ of $\varphi^{\mathcal{B}}$ are finite in number, and easy to enumerate with a SAT-solver
- A model $\mu^{\mathcal{B}}$ is nothing but a **conjunction of $\mathcal{T}$-atoms**, can be checked efficiently with a $\mathcal{T}$-solver

These observations suggest us a methodology to tackle the SMT($\mathcal{T}$) problem

- Enumerate a Boolean model $\mu^{\mathcal{B}}$ of $\varphi^{\mathcal{B}}$ (abstraction). If no model exist we are done (φ is unsatisfiable)
- Check if $\mu^{\mathcal{B}}$ is satisfiable using the $\mathcal{T}$-solver. If so $\mu^{\mathcal{B}}$ can be extended to a model μ of φ, and so we are done ! (φ is satisfiable)
- It not, we tell the SAT-solver not to enumerate $\mu^{\mathcal{B}}$ again, thus **cutting away systematically an infinite number** of assignments for φ (refinement)
- It can be blocked by adding a clause $\neg\mu^{\mathcal{B}}$. Go up
- It terminates because there are finite Boolean models

The lazy approach falls into the so-called **abstraction-refinement** paradigm



The Lazy Approach

The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

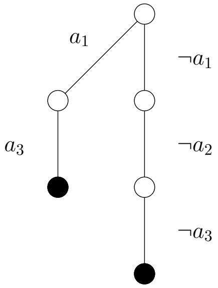
$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\varphi^{\mathcal{B}} \equiv \begin{array}{l} (a_1 \vee \neg a_2) \\ (a_1 \vee \neg a_3) \\ (a_3 \vee \neg a_2) \\ (a_3 \vee \neg a_1) \\ (\neg a_1 \vee \neg a_3) \\ (\neg a_1) \\ (a_1 \vee a_2 \vee a_3) \\ (\ ) \end{array}$$

$$\begin{array}{l} a_1 \equiv x = 3 \\ a_2 \equiv x < 3 \\ a_3 \equiv x > 3 \end{array}$$

$\mu^{\mathcal{B}}$: $\{\ \ \}$

SAT-solver:	UNS
$\mathcal{T}$-solver:	Idle



$(a_1 \vee \neg a_2)$
$(a_1 \vee \neg a_3)$
$(a_3 \vee \neg a_2)$
$(a_3 \vee \neg a_1)$
$(\neg a_1 \vee \neg a_3)$

Trail	dl	Reason
a_1	1	Decision
a_3	1	$(a_3 \vee \neg a_1)$

$\{\overset{1}{a_1}, \overset{1}{a_3}\}$

$$\frac{(\neg a_1 \vee \neg a_3) \qquad (a_3 \vee \neg a_1)}{(\neg a_1)}$$

Conflict clause: $(\neg a_1)$
Backtracking level: 0

$(a_1 \vee \neg a_2)$
$(a_1 \vee \neg a_3)$
$(a_3 \vee \neg a_2)$
$(a_3 \vee \neg a_1)$
$(\neg a_1 \vee \neg a_3)$
$(\neg a_1)$
$(a_1 \vee a_2 \vee a_3)$

Trail	dl	Reason
$\neg a_1$	0	$(\neg a_1)$
$\neg a_2$	0	$(a_1 \vee \neg a_2)$
$\neg a_3$	0	$(a_1 \vee \neg a_3)$

$\{\neg \overset{0}{a_1}, \neg \overset{0}{a_2}, \neg \overset{0}{a_3}\}$

$$\dfrac{\dfrac{\dfrac{(a_1 \vee a_2 \vee a_3) \qquad (a_3 \vee \neg a_1)}{(a_1 \vee a_2)} \qquad (a_1 \vee \neg a_2)}{(a_1)} \qquad (\neg a_1)}{\bot}$$

Conflict clause: $\bot$
Backtracking level: 0

A $\mathcal{T}$-solver for $\mathcal{IDL}$

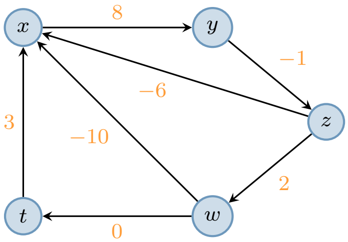
The constraint $x - y \leq c$ says that

"the distance between x and y is at most c"

This can be encoded as $(x, y; c)$

So, a set $\mu^{\mathcal{B}}$ can be encoded as a graph. Concrete example:

$$\begin{array}{rcr} x - y & \leq & 8 \\ y - z & \leq & -1 \\ z - x & \leq & -6 \\ z - w & \leq & 2 \\ w - x & \leq & -10 \\ w - t & \leq & 0 \\ t - x & \leq & 3 \end{array}$$



$G(V, E)$: $V = \{x, y, z, w, t\}$, $E = \{(x, y; 8), (y, z; -1), (z, x; -6), (z, w; 2), (w, x; -10), \ldots\}$

A $\mathcal{T}$-solver for $\mathcal{IDL}$

Theorem (Translation)

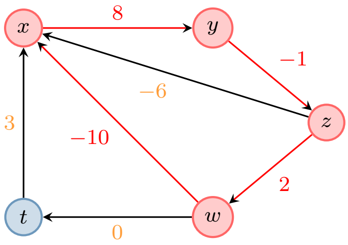
$\mu^{\mathcal{B}}$ is $\mathcal{IDL}$-unsatisfiable

iff

there is a **negative cycle** *in the corresponging graph $G(V, E)$*

E.g.:

$$
\begin{array}{rcr}
x - y & \leq & 8 \\
y - z & \leq & -1 \\
z - x & \leq & -6 \\
z - w & \leq & 2 \\
w - x & \leq & -10 \\
w - t & \leq & 0 \\
t - x & \leq & 3
\end{array}
$$



Lecture 5 - Exercize 4

Let's first recall the notion of **minimality**

A conflict $\nu^{\mathbb{B}}$ is **minimal** if it does not contain redundant $\mathcal{T}$-atoms

A $\mathcal{T}$-atom P in a conflict $\nu^{\mathbb{B}}$ is redundant if $\nu^{\mathbb{B}} \setminus \{P\}$ is still a conflict

So, how do we check, in general, that a conflict $\nu^{\mathbb{B}} = \{P_1, \ldots, P_n\}$ is minimal ? Iteratively for $i = 1, \ldots, n$, we see if $\nu^{\mathbb{B}} \setminus \{P_i\}$ is still a conflict.

In the case of difference logic every conflict is minimal **by construction**. In fact $\nu^{\mathbb{B}}$ is a conflict if and only if it is a **cycle** with negative sum.

Doing $\nu^{\mathbb{B}} \setminus \{P_i\}$ is equivalent to breaking the cycle, no matter what P_i. Therefore all $\mathcal{T}$-atoms are not redundant, and so conflicts are minimal.

Prove

Observation 2

A set of constraints
$\{ x_1 - x_2 \leq c_1,\ x_2 - x_3 \leq c_2,\ \ldots,\ x_{n-1} - x_n \leq c_{n-1} \}$
implies $\quad x_1 - x_n \leq c_n \quad$ iff $\quad c_1 + c_2 + \ldots c_{n-1} \leq c_n$

using

Lemma (Farka's Lemma for $\mathcal{IDL}$)

$\mu^{\mathcal{B}}$ is unsatisfiable iff there exists a subset
$\nu^{\mathcal{B}} = \{ x_1 - x_2 \leq c_1,\ x_2 - x_3 \leq c_2,\ \ldots,\ x_n - x_1 \leq c_n \}$ of $\mu^{\mathcal{B}}$ such that $c_1 + \ldots + c_n < 0$

Lecture 5 - Exercize 5

Observation 2

A set of constraints
$\{\ x_1 - x_2 \leq c_1,\ x_2 - x_3 \leq c_2,\ \dots,\ x_{n-1} - x_n \leq c_{n-1}\ \}$
implies $\quad x_1 - x_n \leq c_n \quad$ iff $\quad c_1 + c_2 + \dots c_{n-1} \leq c_n$

is better formalized as

$$(x_1 - x_2 \leq c_1 \ \wedge\ x_2 - x_3 \leq c_2 \ \wedge\ \dots \ \wedge\ x_{n-1} - x_n \leq c_{n-1}) \rightarrow (x_1 - x_n \leq d_n)$$

is valid iff

$$c_1 + c_2 + \dots c_{n-1} \leq d_n$$

Lemma (Farka's Lemma for $\mathcal{IDL}$)

$\mu^{\mathcal{B}}$ is unsatisfiable iff there exists a subset
$\nu^{\mathcal{B}} = \{\ x_1 - x_2 \leq c_1,\ x_2 - x_3 \leq c_2,\ \dots,\ x_n - x_1 \leq c_n\ \}$ of $\mu^{\mathcal{B}}$ such that $c_1 + \dots + c_n < 0$

is better formalized as

$$x_1 - x_2 \leq c_1 \ \wedge\ x_2 - x_3 \leq c_2 \ \wedge\ \dots \ \wedge\ x_{n-1} - x_n \leq c_{n-1} \ \wedge\ x_n - x_1 \leq c_n$$

is unsatisfiable iff

$$c_1 + c_2 + \dots c_{n-1} + c_n < 0$$

$$(x_1 - x_2 \leq c_1 \ \wedge \ x_2 - x_3 \leq c_2 \ \wedge \ \ldots \ \wedge \ x_{n-1} - x_n \leq c_{n-1}) \rightarrow (x_1 - x_n \leq d_n)$$
is valid iff
$$c_1 + c_2 + \ldots c_{n-1} \leq d_n$$

is equivalent to (using the well-known fact: $\varphi \rightarrow \psi$ is valid iff $\varphi \wedge \neg\psi$ is unsat)

$$x_1 - x_2 \leq c_1 \ \wedge \ x_2 - x_3 \leq c_2 \ \wedge \ \ldots \ \wedge \ x_{n-1} - x_n \leq c_{n-1} \ \wedge \ \neg(x_1 - x_n \leq d_n)$$
is unsatisfiable iff
$$c_1 + c_2 + \ldots c_{n-1} \leq d_n$$

which is equivalent to (using the fact $\neg(x - y \leq c) \iff y - x \leq -c - 1$)

$$x_1 - x_2 \leq c_1 \ \wedge \ x_2 - x_3 \leq c_2 \ \wedge \ \ldots \ \wedge \ x_{n-1} - x_n \leq c_{n-1} \ \wedge \ x_n - x_1 \leq -d_n - 1$$
is unsatisfiable iff
$$c_1 + c_2 + \ldots c_{n-1} \leq d_n$$

which is equivalent to (using the fact $c \leq d \iff c - d \leq 0 \iff c - d - 1 < 0$)

$$x_1 - x_2 \leq c_1 \ \wedge \ x_2 - x_3 \leq c_2 \ \wedge \ \ldots \ \wedge \ x_{n-1} - x_n \leq c_{n-1} \wedge x_n - x_1 \leq -d_n - 1$$
is unsatisfiable iff
$$c_1 + c_2 + \ldots c_{n-1} - d_n - 1 < 0$$

which is Farka's Lemma if we set $c_n \equiv -d_n - 1$

Tableau			*lb*		Bounds		*ub*		μ	
	...		-4	$\leq$	x_1	$\leq$	10	x_1	$\mapsto$	12
x_1	$=$	$3x_2 - 4x_3 + 2x_4 - x_5$	1	$\leq$	x_2	$\leq$	3	x_2	$\mapsto$	1
	...		-4	$\leq$	x_3	$\leq$	-1	x_3	$\mapsto$	-1
			1	$\leq$	x_4	$\leq$	2	x_4	$\mapsto$	2
			-1	$\leq$	x_5	$\leq$	10	x_5	$\mapsto$	-1

which among $\mathcal{N} = \{x_2, x_3, x_4\}$ do I choose for pivoting ? Clearly, the value of $\mu(x_1)$ is too high, I have to decrease it by playing with the values of $\mathcal{N}$:

- $3x_2$ cannot decrease, as $\mu(x_2) = lb(x_2)$ and cannot be moved down
- $-4x_3$ cannot decrease, as $\mu(x_3) = ub(x_3)$ and cannot be moved up
- $2x_4$ can decrease, as $\mu(x_4) = ub(x_4)$, and can be moved down
- $-x_5$ can decrease, as $\mu(x_5) = lb(x_5)$, and can be moved up

both x_4 and x_5 are therefore good candidates for pivoting. To avoid loops, choose variable with smallest subscript (Bland's Rule). This rule is not necessarily efficient, though

There might be cases in which **no suitable variable for pivoting can be found**. This indicates unsatisfiability. Consider the following where we have just asserted $x_1 \leq 9$

Tableau	*lb*	Bounds	*ub*	μ
$\begin{array}{rcl} & \dots & \\ x_1 & = & 3x_2 - 4x_3 + 2x_4 - x_5 \\ & \dots & \end{array}$	$\begin{array}{r} -4 \\ 1 \\ -4 \\ 2 \\ -1 \end{array}$	$\begin{array}{ccc} \leq & x_1 & \leq \\ \leq & x_2 & \leq \\ \leq & x_3 & \leq \\ \leq & x_4 & \leq \\ \leq & x_5 & \leq \end{array}$	$\begin{array}{r} 9 \\ 3 \\ -1 \\ 2 \\ -1 \end{array}$	$\begin{array}{ccr} x_1 & \mapsto & 12 \\ x_2 & \mapsto & 1 \\ x_3 & \mapsto & -1 \\ x_4 & \mapsto & 2 \\ x_5 & \mapsto & -1 \end{array}$

no variable among $\mathcal{N} = \{x_2, x_3, x_4\}$ can be chosen for pivoting. This is because (due to tableau)

$$x_2 \geq 1 \;\wedge\; x_3 \leq -1 \;\wedge\; x_4 \geq 4 \;\wedge\; x_5 \leq -1 \;\Rightarrow\; x_1 \geq 12 \;\Rightarrow\; \neg(x_1 \leq 9)$$

Therefore

$$\{x_2 \geq 1,\; x_3 \leq -1,\; x_4 \geq 4,\; x_5 \leq -1,\; \neg(x_1 \leq 9)\}$$

is a $\mathcal{T}$-conflict (modulo the row $x_1 = 3x_2 - 4x_3 + 2x_4 - x_5$)

A conflict returned by the Simplex involves a row

$$x_1 = a_2 x_2 + \ldots + a_n x_n$$

and exactly n bounds

$$\{x_1 \sim_1 b_1, x_2 \sim_2 b_2, \ldots, x_n \sim_n b_n\}$$

with $\sim_i \in \{\leq, \geq\}$.
This conflict is minimal: we show that we can find a model μ if we remove a bound. W.l.o.g., we remove $x_2 \sim_2 b_2$: then we can set $\mu(x_j) = b_j$ for $j \neq 2$. Then we can pivot the row

$$x_2 = c_1 x_1 + c_3 x_3 + \ldots + c_n x_n$$

and compute a suitable value for $\mu(x_2) = c_1\mu(x_1) + \ldots + c_n\mu(x_n)$.
(we can set the value we want to $\mu(x_2)$ because it is unbounded now!)

A $\mathcal{T}$-solver for $\mathcal{UF}$

The solving phase inside the $\mathcal{UF}$-solver happens in two steps

Given a conjunction of $\mathcal{T}$-literals φ to check for satisfiability, the $\mathcal{UF}$-solver

first it constructs **equivalence classes** using φ^+ (the set of positive $\mathcal{T}$-literals), using, for example, the Union-Find algorithm

and then checks one by one the negative $\mathcal{T}$-literals in φ^-

A $\mathcal{T}$-solver for $\mathcal{UF}$

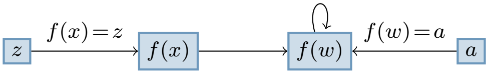
First of all notice that $\mathcal{T}$-conflicts are always of the form

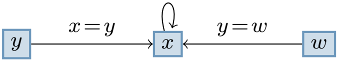
$$\{\ s \neq t, \text{“other equalities that cause } s \text{ and } t \text{ to join the same class”}\}$$

We reconstruct the conflict by storing the reason that caused two classes to collapse. When a $s \neq t$ causes *unsat*, we call a routine Explain(s,t) that collects all the reasons that made s equal to t

Example

$$\{\ x = y,\ y = w,\ f(x) = z,\ f(w) = a,\ a \neq z\ \}$$





On processing $a \neq z$, we call Explain(a, z)

Lecture 7 - Exercize 2

We can store a reason of a Union inside the struct `Node`

```
struct Node {
  ...
  Node *  root;     // Points to class' representant
  Enode * reason;   // T-atom that caused union
};
```

explanation = { } // Global variable

procedure Explain(s, t)

if ($s == t$) return // Nothing to explain ...

while($s.next \neq s$) // Collect reasons while moving to root
  if ($s.reason ==$ NULL) // No reason: union caused by congruence
    // Let $s \equiv f(s_1, \ldots, s_n)$, $s.next \equiv f(t_1, \ldots, t_n)$
    foreach $i \in \{1, \ldots, n\}$
      Explain(s_i, t_i)
  else // Reason exists, save it
    explanation = explanation $\cup$ $\{s.reason\}$
  $s = s.next$

// Same "while" loop for t ...