Satisfiability Modulo Linear Arithmetic

Combinatorial Problem Solving (CPS)

Albert Oliveras    Enric Rodríguez-Carbonell

May 31, 2019
Linear Arithmetic Theories

- In linear arithmetic theories, atoms are of the form:

  \[ a_1x_1 + \ldots + a_nx_n \bowtie b \]

  where \( \bowtie \) is one of: \( =, \neq, <, >, \leq, \geq \)

  All symbols are interpreted with their usual meaning in arithmetic

- Example of atom:

  \[ x + y + 2z \geq 10 \]

- Example of formula:

  \[ x \geq 0 \land (x + y \leq 2 \lor x - y \geq 6) \land (x + y \geq 1 \lor x - y \geq 4) \]

- Variables can be of real sort (\( \mathbb{R} \)) or integer sort (\( \mathbb{Z} \))

- If all vars are \( \mathbb{R} \) we have a problem of Linear Real Arithmetic (LRA)

- If all vars are \( \mathbb{Z} \) we have a problem of Linear Integer Arithmetic (LIA)
Overview of the Lecture

- De Moura & Dutertre’s Algorithm for LRA
- LIA
De Moura & Dutertre’s Algorithm

- Problem: given an input formula $\phi$ of LRA, is $\phi$ SAT?
- Assume for the time being $\phi$ only contains linear constraints of the form $c^T x \leq d$
Problem: given an input formula $\phi$ of LRA, is $\phi$ SAT?

Assume for the time being $\phi$ only contains linear constraints of the form $c^T x \leq d$

Preprocessing: transform $\phi$ into $\hat{\phi} \land Ax = 0$, where:

1. $\hat{\phi}$ is obtained from $\phi$ by replacing each $c^T x \leq d$ by $s \leq d$, where $s$ is fresh variable
2. $Ax = 0$ consists of all definitions $s = c^T x$
De Moura & Dutertre’s Algorithm

- Problem: given an input formula $\phi$ of LRA, is $\phi$ SAT?
- Assume for the time being $\phi$ only contains linear constraints of the form $c^T x \leq d$
- **Preprocessing:** transform $\phi$ into $\hat{\phi} \land Ax = 0$, where:
  1. $\hat{\phi}$ is obtained from $\phi$ by replacing each $c^T x \leq d$ by $s \leq d$, where $s$ is fresh variable
  2. $Ax = 0$ consists of all definitions $s = c^T x$
- Example:

\[
x \geq 0 \land (x + y \leq 2 \lor x - y \geq 6) \land (x + y \geq 1 \lor x - y \geq 4)
\]

\[
x \geq 0 \land (s_1 \leq 2 \lor s_2 \geq 6) \land (s_1 \geq 1 \lor s_2 \geq 4) \land (s_1 = x + y \land s_2 = x - y)
\]
De Moura & Dutertre’s Algorithm

- Consistency checking is based on dual bounded simplex
- Theory solver handles feasibility problems of the form
  \[ Ax = 0 \land \ell \leq x \leq u \]
- Only bounds asserted during search:
  
  \[ Ax = 0 \text{ is asserted before any decision} \]

  There is no addition/deletion of rows!
De Moura & Dutertre’s Algorithm

- Consistency checking is based on dual bounded simplex
- Theory solver handles feasibility problems of the form

\[ Ax = 0 \land \ell \leq x \leq u \]

- Only bounds asserted during search:
  \( Ax = 0 \) is asserted before any decision
  There is no addition/deletion of rows!

- Free variables (those without any bound in the formula) can be eliminated before starting search by means of Gaussian elimination
- E.g.: if \( y \) is free then equation \( y = x - s_2 \) is not asserted

\[ x \geq 0 \land (s_1 \leq 2 \lor s_2 \geq 6) \land (s_1 \geq 1 \lor s_2 \geq 4) \land (s_1 = 2x - s_2 \land y = x - s_2) \]
Basic Solver

For solving $Ax = 0 \land \ell \leq x \leq u$, theory solver stores:

- A tableau: $x_i = \sum_{x_j \in \mathcal{R}} \alpha_{ij} x_j$, $x_i \in \mathcal{B}$
- For each variable $x_i$,
  - the strongest asserted lower bound $\ell_i$
  - the strongest asserted upper bound $u_i$
- An assignment $\beta$ such that
  - $A\beta = 0$
  - For each $x_j \in \mathcal{R}$: $\ell_j \leq \beta(x_j) \leq u_j$
For solving $Ax = 0 \land \ell \leq x \leq u$, theory solver stores:

- A tableau: $x_i = \sum_{j \in \mathcal{R}} \alpha_{ij} x_j$, $x_i \in \mathcal{B}$
- For each variable $x_i$,
  - the strongest asserted lower bound $\ell_i$
  - the strongest asserted upper bound $u_i$
- An assignment $\beta$ such that
  - $A\beta = 0$
  - For each $x_j \in \mathcal{R}$: $\ell_j \leq \beta(x_j) \leq u_j$

- Maybe for some $x_i \in \mathcal{B}$, $\ell_i > \beta(x_i)$ or $\beta(x_i) > u_j$
- Maybe for some $x_i \in \mathcal{R}$, $\ell_i < \beta(x_i) < u_j$

Supports two types of consistency checks: light-weight and heavy-weight
Light-Weight Consistency Check

- Ensures non basic vars satisfy bounds and $A\beta = 0$

  - If returns SAT: Then model is consistent
  - If returns UNSAT: Then model is inconsistent
  - If returns UNKNOWN: Don’t know

```
assert_lower(x_j ≥ c_j)
    if $c_j \leq \ell_j$ then return SAT
    if $c_j > u_j$ then return UNSAT
    $\ell_j := c_j$
    if $x_j \in R \land \beta(x_j) < \ell_j$ then update($x_j, \ell_j$)
    return UNKNOWN

update(x_j, v)
    for each $x_i \in B$, $\beta(x_i) := \beta(x_i) + \alpha_{ij}(v - \beta(x_j))$
    $\beta(x_j) := v$
```
Light-Weight Consistency Check

Ensures non basic vars satisfy bounds and \( A\beta = 0 \)

- If returns SAT : Then model is consistent
- If returns UNSAT: Then model is inconsistent
- If returns UNKNOWN: Don’t know

\[
\text{assert\_upper}(x_j \leq c_j) \\
\quad \text{if } c_j \geq u_j \text{ then return SAT} \\
\quad \text{if } c_j < \ell_j \text{ then return UNSAT} \\
\quad u_j := c_j; \\
\quad \text{if } x_j \in \mathcal{R} \land \beta(x_j) > u_j \text{ then update}(x_j, u_j) \\
\text{return UNKNOWN}
\]

\[
\text{update}(x_j, v) \\
\quad \text{for each } x_i \in \mathcal{B}, \quad \beta(x_i) := \beta(x_i) + \alpha_{ij}(v - \beta(x_j)) \\
\quad \beta(x_j) := v
\]
Heavy-Weight Consistency Check

- Light-weight consistency check is performed first (since it is cheaper)
- The only possible cases of unfeasibility that are left: bounds of basic vars
- **Dual Bounded Simplex** (with null objective function)
  is employed to get feasible basis
- Constraints are handled in blocks (as opposed to one at a time)
Heavy-Weight Consistency Check

check()
    loop
        select basic variable \( x_i \) such that \( \beta_i < \ell_i \) or \( \beta_i > u_i \)
        if there is no such \( x_i \) then return SAT
        if \( \beta_i < \ell_i \) then
            select non-basic variable \( x_j \) such that
            \[(\alpha_{ij} > 0 \land \beta(x_j) < u_j) \lor (\alpha_{ij} < 0 \land \beta(x_j) > \ell_j)\]
            if there is no such \( x_j \) then return UNSAT
            pivot_and_update\((x_i, x_j, \ell_i)\)
        if \( \beta_i > u_i \) then
            select non-basic variable \( x_j \) such that
            \[(\alpha_{ij} < 0 \land \beta(x_j) < u_j) \lor (\alpha_{ij} > 0 \land \beta(x_j) > \ell_j)\]
            if there is no such \( x_j \) then return UNSAT
            pivot_and_update\((x_i, x_j, u_i)\)

    /* pivot_and_update\((x_i, x_j, v)\):
    set basic \( x_i \) to \( v \), adjust non-basic \( x_j \) and other basic vars as needed,
    swap \( x_i \) and \( x_j \) in the basis */
pivot_and_update(x_i, x_j, v)
/* set basic x_i to v, adjust non-basic x_j and other basic vars as needed, swap x_i and x_j in the basis */

\[ \Theta := \frac{v - \beta(x_i)}{\alpha_{ij}} \]
\[ \beta(x_i) := v \]
\[ \beta(x_j) := \beta(x_j) + \Theta \]
for each \( x_k \in B \land x_k \neq x_i \), \[ \beta(x_k) := \beta(x_k) + \alpha_{kj} \Theta \]

Recall Bland’s anticycling rule in dual pricing and dual ratio test:
- Set an order between variables
- Always take the least possible variable

THEOREM. This strategy guarantees termination
Conflict Explanations

- check() detects an inconsistency when:
  - If $\beta_i < \ell_i$ and for all non-basic $x_j$
    
    $\alpha_{ij} > 0 \rightarrow \beta(x_j) \geq u_j \land \alpha_{ij} < 0 \rightarrow \beta(x_j) \leq \ell_j$
  
  - If $\beta_i > u_i$ and for all non-basic $x_j$
    
    $\alpha_{ij} < 0 \rightarrow \beta(x_j) \geq u_j \land \alpha_{ij} > 0 \rightarrow \beta(x_j) \leq \ell_j$

- Let $R^+ = \{x_j \in R \mid \alpha_{ij} > 0\}$ and $R^- = \{x_j \in R \mid \alpha_{ij} < 0\}$

- Since $\beta$ satisfies all bounds on non-basic vars:
  - If $\beta(x_i) < \ell_i$
    
    - for all $x_j \in R^+, \beta(x_j) = u_j$
    
    - for all $x_j \in R^-, \beta(x_j) = \ell_j$
  
  - If $\beta(x_i) > u_i$
    
    - for all $x_j \in R^+, \beta(x_j) = \ell_j$
    
    - for all $x_j \in R^-, \beta(x_j) = u_j$
Conflict Explanations

■ Assume $\beta(x_i) < l_i$.

■ So for all $x_j \in R^+$, $\beta(x_j) = u_j$ and for all $x_j \in R^-$, $\beta(x_j) = l_j$

■ Hence $\beta(x_i) = \sum_{x_j \in R^+} \alpha_{ij} u_j + \sum_{x_j \in R^-} \alpha_{ij} l_j$

■ So for any $x$ such that $Ax = b$

$$\beta(x_i) - x_i = \sum_{x_j \in R^+} \alpha_{ij} (u_j - x_j) + \sum_{x_j \in R^-} \alpha_{ij} (l_j - x_j)$$

■ From this we can derive the implication

$$\wedge_{x_j \in R^+} x_j \leq u_j \land \wedge_{x_j \in R^-} x_j \geq l_j \Rightarrow x_i \leq \beta(x_i)$$

■ Since $\beta(x_i) < l_i$ this is inconsistent with $x_i \geq l_i$

■ The explanation of the conflict is

$$\{x_j \leq u_j \mid x_j \in R^+\} \cup \{x_j \geq l_j \mid x_j \in R^-\} \cup \{x_i \geq l_i\}$$

which is minimal (with respect to set inclusion)
Conflict Explanations

- Assume $\beta(x_i) > u_i$.
- So for all $x_j \in \mathcal{R}^+$, $\beta(x_j) = \ell_j$ and for all $x_j \in \mathcal{R}^-$, $\beta(x_j) = u_j$.
- Hence $\beta(x_i) = \sum_{x_j \in \mathcal{R}^+} \alpha_{ij} \ell_j + \sum_{x_j \in \mathcal{R}^-} \alpha_{ij} u_j$.
- So for any $x$ such that $Ax = b$
  \[
  \beta(x_i) - x_i = \sum_{x_j \in \mathcal{R}^+} \alpha_{ij} (\ell_j - x_j) + \sum_{x_j \in \mathcal{R}^-} \alpha_{ij} (u_j - x_j)
  \]
- From this we can derive the implication
  \[
  \bigwedge_{x_j \in \mathcal{R}^+} x_j \geq \ell_j \land \bigwedge_{x_j \in \mathcal{R}^-} x_j \leq u_j \Rightarrow x_i \geq \beta(x_i)
  \]
- Since $\beta(x_i) > u_i$ this is inconsistent with $x_i \leq u_i$.
- The explanation of the conflict is
  \[
  \{x_j \geq \ell_j \mid x_j \in \mathcal{R}^+\} \cup \{x_j \leq u_j \mid x_j \in \mathcal{R}^-\} \cup \{x_i \leq u_i\}
  \]
  which is minimal (with respect to set inclusion)
Backtracking

- Number of backtrackings is often very large: needs to be efficiently implemented
- The algorithm only requires, for each variable $x_i$,
  - one stack for lower bounds $\ell_i$
  - one stack for upper bounds $u_i$
- No need to save successive $\beta$ on a stack!
  Only one assignment $\beta$ is kept
- Recall: for each $x_j \in \mathcal{R}$, then $\ell_j \leq \beta(x_j) \leq u_j$
  Maybe for some $x_i \in \mathcal{R}, \ell_i < \beta(x_i) < u_j$
- Does not require pivoting: very cheap
Theory Propagation

■ Simple propagation: $x \geq c$ implies $x \geq c'$ for all $c' \leq c$

■ Bound refinement:
  
given an equation $x_i = \sum \alpha_j x_j$ that holds for any $x$ solution to $Ax = 0$,
  then we can deduce bounds:

$$x_i \geq \sum_{\alpha_j > 0} \alpha_j \ell_j + \sum_{\alpha_j < 0} \alpha_j u_j$$
$$x_i \leq \sum_{\alpha_j > 0} \alpha_j u_j + \sum_{\alpha_j < 0} \alpha_j \ell_j$$

■ Might not be better bounds than those already asserted

■ Used with tableau rows
  
  (but can be used with rows of original problem
  or any linear combination of them)
Example

Assert literals $x \leq -4$, $x \geq -8$, $-x + y \leq 1$, $x + y \geq -3$

- **TABLEAU**

\[
\begin{cases}
  s_1 = -x + y \\
  s_2 = x + y
\end{cases}
\]

- **ASSIGNMENT**

\[
\begin{align*}
  x & \rightarrow 0 \\
  y & \rightarrow 0 \\
  s_1 & \rightarrow 0 \\
  s_2 & \rightarrow 0
\end{align*}
\]

- **BOUNDS**
Example

Assert literals $x \leq -4$, $x \geq -8$, $-x + y \leq 1$, $x + y \geq -3$

- **TABLEAU**
  \[
  \begin{cases}
  s_1 &= -x + y \\
  s_2 &= x + y
  \end{cases}
  \]

- **ASSIGNMENT**
  \[
  \begin{align*}
  x &\rightarrow -4 \\
  y &\rightarrow 0 \\
  s_1 &\rightarrow 4 \\
  s_2 &\rightarrow -4
  \end{align*}
  \]

- **BOUNDS**
  \[
  x \leq -4
  \]
Example

Assert literals $x \leq -4$, $x \geq -8$, $-x + y \leq 1$, $x + y \geq -3$

- **TABLEAU**
  \[
  \begin{align*}
  s_1 &= -x + y \\
  s_2 &= x + y
  \end{align*}
  \]

- **ASSIGNMENT**
  
  \[
  \begin{align*}
  x &\rightarrow -4 \\
  y &\rightarrow 0 \\
  s_1 &\rightarrow 4 \\
  s_2 &\rightarrow -4
  \end{align*}
  \]

- **BOUNDS**
  
  $-8 \leq x \leq -4$
Example

Assert literals $x \leq -4$, $x \geq -8$, $-x + y \leq 1$, $x + y \geq -3$

- **TABLEAU**
  \[
  \begin{align*}
  y &= x + s_1 \\
  s_2 &= 2x + s_1
  \end{align*}
  \]

- **ASSIGNMENT**
  \[
  \begin{align*}
  x &\rightarrow -4 \\
  y &\rightarrow -3 \\
  s_1 &\rightarrow 1 \\
  s_2 &\rightarrow -7
  \end{align*}
  \]

- **BOUNDS**
  \[
  \begin{align*}
  -8 &\leq x \leq -4 \\
  s_1 &\leq 1
  \end{align*}
  \]
Example

Assert literals $x \leq -4$, $x \geq -8$, $-x + y \leq 1$, $x + y \geq -3$

- **TABLEAU**

\[
\begin{align*}
  y &= x + s_1 \\
  s_2 &= 2x + s_1
\end{align*}
\]

- **ASSIGNMENT**

\[
\begin{align*}
  x &\rightarrow -4 \\
  y &\rightarrow -3 \\
  s_1 &\rightarrow 1 \\
  s_2 &\rightarrow -7
\end{align*}
\]

- **BOUNDS**

\[
\begin{align*}
  -8 &\leq x \leq -4 \\
  s_1 &\leq 1 \\
  -3 &\leq s_2
\end{align*}
\]
Example

Assert literals $x \leq -4$, $x \geq -8$, $-x + y \leq 1$, $x + y \geq -3$

■ TABLEAU

\[
\begin{align*}
  y &= x + s_1 \\
  s_2 &= 2x + s_1
\end{align*}
\]

■ ASSIGNMENT

\[
\begin{align*}
  x &\rightarrow -4 \\
  y &\rightarrow -3 \\
  s_1 &\rightarrow 1 \\
  s_2 &\rightarrow -7
\end{align*}
\]

■ BOUNDS

\[
\begin{align*}
  -8 &\leq x \leq -4 \\
  s_1 &\leq 1 \\
  -3 &\leq s_2
\end{align*}
\]

Conflict between $x \leq -4$, $s_1 \leq 1$, $-3 \leq s_2$!
LEMMA. A set of linear arithmetic literals $\Gamma$ containing strict inequalities $S = \{p_1 > 0, \ldots, p_n > 0\}$ is satisfiable iff there exists a rational number $\delta > 0$ s.t. for all $\delta'$ s.t. $0 < \delta' \leq \delta$, $\Gamma_{\delta} = (\Gamma \cup S_{\delta}) - S$ is satisfiable, where $S_{\delta} = \{p_1 \geq \delta, \ldots, p_n \geq \delta\}$

Strict inequalities are transformed into non-strict ones using an infinitesimal positive symbolic value $\delta$:

- $x > a \quad \rightarrow \quad x \geq a + \delta$
- $x < a \quad \rightarrow \quad x \leq a - \delta$

Disequalities $c^T x \neq d$ have to be split into $c^T x < d \lor c^T x > d$ while parsing.

Equalities $c^T x = d$ have to be split into $c^T x \leq d \land c^T x \geq d$ while parsing.
Strict Inequalities and Disequalities

- $\delta$ is not given a concrete value
  Just treated symbolically!

- Values are pairs of rationals with ordering:
  - $a + b\delta \leq a' + b'\delta$ iff $a < a'$, or $a = a'$ and $b \leq b'$

- Arithmetic operations are defined pairwise:
  - $(a + b\delta) + (a' + b'\delta) = (a + a') + (b + b')\delta$
  - $c \cdot (a + b\delta) = (c \cdot a) + (c \cdot b)\delta$

- From now on let $\mathbb{Q}_\delta = \{a + b\delta \mid a, b \in \mathbb{Q}\}$
Strict Inequalities and Disequalities

**LEMMA.** Let \( v_i = c_i + k_i \delta \), \( w_i = d_i + h_i \delta \) \((i = 1 \ldots m)\) be such that \( v_i \leq w_i \) hold. Then there is \( \delta_0 \in \mathbb{Q} \) such that \( \delta_0 > 0 \) and

\[
\begin{align*}
    c_1 + k_1 \epsilon & \leq d_1 + h_1 \epsilon \\
    \vdots & \\
    c_m + k_m \epsilon & \leq d_m + h_m \epsilon
\end{align*}
\]

are satisfied for any \( \epsilon \) such that \( 0 < \epsilon \leq \delta_0 \).
Strict Inequalities and Disequalities

PROOF: By definition
\[ c_i + k_i \delta \leq d_i + h_i \delta \text{ iff } c_i < d_i, \text{ or } c_i = d_i \text{ and } k_i \leq h_i \]

We distinguish several cases:

- If \( c_i = d_i \) and \( k_i \leq h_i \) then \( c_i + k_i \epsilon \leq d_i + h_i \epsilon \) for any \( \epsilon > 0 \)
- If \( c_i < d_i \) and \( k_i \leq h_i \) then \( c_i + k_i \epsilon \leq d_i + h_i \epsilon \) for any \( \epsilon > 0 \)
- If \( c_i < d_i \) and \( k_i > h_i \) then \( c_i + k_i \epsilon \leq d_i + h_i \epsilon \) for any \( \epsilon \) such that \( 0 < \epsilon \leq \frac{d_i - c_i}{k_i - h_i} \)

So for example take \( \delta_0 \) such that

\[
0 < \delta_0 < \min \left\{ \frac{d_i - c_i}{k_i - h_i} \mid c_i < d_i \text{ and } k_i > h_i \right\}
\]
Let $S$ be a linear problem of the form

$$Ax = 0 \land \ell \preceq x \preceq u$$

where $\ell, u \in \mathbb{Q}$ and $\preceq_i$, $\succeq_i$ are either $<$ or $\leq$.

$S$ can be converted into a problem $S'$ of the form

$$Ax = 0 \land \ell' \leq x \leq u'$$

where $\ell', u' \in \mathbb{Q}_\delta$ as follows:

- $x_i > \ell_i \rightarrow x_i \geq \ell'_i$ with $\ell'_i = \ell_i + \delta$
- $x_i < u_i \rightarrow x_i \leq u'_i$ with $u'_i = u_i - \delta$
**THEOREM.** $S$ and $S'$ are equisatisfiable.

**PROOF:** Let us see $S'$ sat in $Q_\delta$ implies $S$ sat in $Q$.

Let $\beta'$ be a satisfying assignment for $S'$.

The inequalities $\ell'_j \leq \beta'(x_j) \leq u'_j$ are satisfied in $Q_\delta$.

Let $\beta'(x_j) = p_j + q_j \delta$, $\ell'_j = \ell_j + k_j \delta$, $u'_j = u_j + h_j \delta$ where $k_j \in \{0, 1\}$, $k_j = 0$ iff $\bowtie_i^-$ is $\leq$,

$h_j \in \{0, -1\}$, $h_j = 0$ iff $\bowtie_i^+$ is $\leq$.

By the previous lemma, there is $\delta_0 \in \mathbb{R}$, $\delta_0 > 0$ such that

$$\ell_j + k_j \delta_0 \leq p_j + q_j \delta_0 \leq u_j + h_j \delta_0$$

Let us define $\beta(x_j) = p_j + q_j \delta_0$ for all $x_j$.

Then $\beta$ satisfies both $\ell \bowtie x \bowtie u$ as well as $Ax = 0$
Strict Inequalities and Disequalities

PROOF (continued):

Let us see $S$ sat in $Q$ implies $S'$ sat in $Q_\delta$.

Trivial: any satisfying assignment $\beta$ for $S$ in $Q$ is a satisfying assignment for $S'$ in $Q_\delta$. 
Overview of the Lecture

- De Moura & Dutertre’s Algorithm for LRA
- LIA
SMT(LIA)

State-of-the-art SMT solvers for LIA use:

- Branch & Bound
- Cutting Planes
- GCD Test

Strict inequalities are transformed into non-strict ones:

\[
\begin{align*}
  x > a &\quad \rightarrow \quad x \geq a + 1 \\
  x < a &\quad \rightarrow \quad x \leq a - 1
\end{align*}
\]

So in what follows, all constraints will be non-strict
Branch & Bound (Feasibility)

\[ S := \{ P_0 \} \]

/* set of pending problems */

while \( S \neq \emptyset \) do

- remove \( P \) from \( S \); solve \( \text{LP}(P) \)

  - if \( \text{LP}(P) \) is feasible then
    - Let \( \beta \) be basic solution obtained after solving \( \text{LP}(P) \)
    - if \( \beta \) satisfies integrality constraints then
      - return SATISFIABLE
    - else
      - Let \( x_j \) be integer variable such that \( \beta_j \notin \mathbb{Z} \)
      - \( S := S \cup \{ P \land x_j \leq \lfloor \beta_j \rfloor, \ P \land x_j \geq \lceil \beta_j \rceil \} \)

  - return UNSATISFIABLE
Splitting on Demand

Two ways to implement Branch & Bound in SMT:

1. Branch & Bound is internal to the theory solver
   - Modular and flexible
   - Lots of code are repeated in SAT/theory solvers: splitting heuristics, stack, etc.

2. Delegate splits to SAT solver: splitting on demand
   - Whenever theory solver needs to split on $x_j$, it invents new lit $l$ and asks SAT solver to split on it
   - Internal meaning of the literal for theory solver:
     - $l \equiv x_j \leq \lfloor \beta_j \rfloor$
     - $\neg l \equiv x_j \geq \lceil \beta_j \rceil$
   - Implementation of theory solver can be simplified
GCD Test

- Quick test which, if positive, ensures problem is UNSAT
- Let us consider an equation $\sum_{i=1}^{n} a_i x_i = b$ where $a_i, b \in \mathbb{Z}$
- Notation: $c \mid d$ means “$c$ divides $d$”. 
  GCD$(x, y)$ is the greatest common divisor of $x$ and $y$.
- Let $g = \gcd(a_1, \ldots, a_n)$. If $g \nmid b$ then equation is UNSAT
**GCD Test**

- Quick test which, if positive, ensures problem is UNSAT
- Let us consider an equation $\sum_{i=1}^{n} a_i x_i = b$ where $a_i, b \in \mathbb{Z}$
- Notation: $c \mid d$ means “$c$ divides $d$”.
  GCD($x, y$) is the greatest common divisor of $x$ and $y$.
- Let $g = \text{GCD}(a_1, \ldots, a_n)$. If $g \nmid b$ then equation is UNSAT

**PROOF:** If $x_i \in \mathbb{Z}$ satisfy the equation then $g \mid a_i$ implies $g \mid a_i x_i$, and hence $g \mid \sum_{i=1}^{n} a_i x_i = b$
GCD Test

- Quick test which, if positive, ensures problem is UNSAT
- Let us consider an equation \( \sum_{i=1}^{n} a_i x_i = b \) where \( a_i, b \in \mathbb{Z} \)
- Notation: \( c \mid d \) means “\( c \) divides \( d \)”.
  GCD(\( x, y \)) is the greatest common divisor of \( x \) and \( y \).
- Let \( g = \text{GCD}(a_1, \ldots, a_n) \). If \( g \nmid b \) then equation is UNSAT

**PROOF:** If \( x_i \in \mathbb{Z} \) satisfy the equation then \( g \mid a_i \) implies \( g \mid a_i x_i \), and hence \( g \mid \sum_{i=1}^{n} a_i x_i = b \)

- In theory solver GCD test can be applied to tableau rows
Bibliography - Further Reading