Constraint Solving

René Thiemann and Fabian Mitterwallner

based on a previous course by Aart Middeldorp

Outline

Example (Application of Linear Integer Arithmetic: Termination Proving)

- consider program
- model loop-iteration as formula φ using pre-variables $\vec{x}$ and post-variables $\vec{x}'$
- prove termination by choosing expression e and integer constant f and show that two LIA problems are unsatisfiable
  - $\varphi \wedge e(\vec{x}) < e(\vec{x}') + 1$ (¬ decrease)
  - $\varphi \wedge e(\vec{x}) < f$ (¬ bounded)
- for certain programs, reasoning over integers is essential

Example (Application of Linear Integer Arithmetic: Termination Proving)

- consider program
- model loop-iteration as formula φ using pre-variables $\vec{x}$ and post-variables $\vec{x}'$
- prove termination by choosing expression e and integer constant f and show that two LIA problems are unsatisfiable
  - $\varphi \wedge e(\vec{x}) < e(\vec{x}') + 1$ (¬ decrease)
  - $\varphi \wedge e(\vec{x}) < f$ (¬ bounded)
- for certain programs, reasoning over integers is essential

Branch-and-Bound Algorithm

- core idea for finding integral solution
  - simplex algorithm is used to find real solution v or detect unsat in $\mathbb{R}$
  - whenever $q := v(x) \notin \mathbb{Z}$, consider two possibilities: add $x \leq \lfloor q \rfloor$ or $\lceil q \rceil \leq x$
- small model property is required for termination: obtain finite search space

Theorem (Small Model Property)

if LIA formula ψ has solution over $\mathbb{Z}$ then it has a solution v with

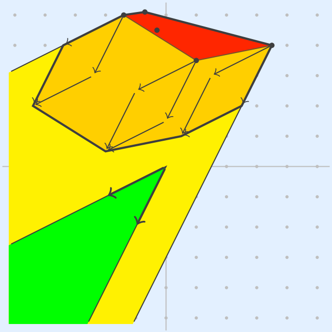
$$|v(x)| \leq bound(\psi) := (n+1) \cdot \sqrt{n^n} \cdot c^n$$

for all x where

- *n: number of variables in ψ*
- *c: maximal absolute value of numbers in ψ*

Proof Idea of Small Model Property

1. convert conjunctive LIA formula ψ into form $A\vec{x} \leq \vec{b}$
2. represent polyhedron $\underbrace{\{\vec{x} \mid A\vec{x} \leq \vec{b}\}}_{\textit{yellow}}$ as polyhedron $P = \underbrace{\textit{hull}(X)}_{\textit{red}} + \underbrace{\textit{cone}(V)}_{\textit{green}}$
3. show that P has small integral solutions (orange), depending on X and V
4. approximate entries of vectors in X and V to obtain small model property



Outline

Tightening

- consider inequality $\sum a_i x_i \bowtie b$ with $\bowtie \in \{<, \leq\}$, and $a_1, \ldots, a_n, b \in \mathbb{Z}$
- tightening preprocesses such inequality in a way that preserves integer solutions (but removes solutions in $\mathbb{R}$)
- impact: tightening helps to obtain unsatisfiability in $\mathbb{R}$

Tightening

- consider inequality $\sum a_i x_i \bowtie b$ with $\bowtie \in \{<, \leq\}$, and $a_1, \dots, a_n, b \in \mathbb{Z}$
- tightening preprocesses such inequality in a way that preserves integer solutions (but removes solutions in $\mathbb{R}$)
- impact: tightening helps to obtain unsatisfiability in $\mathbb{R}$
- last lecture: tightening of strict inequalities $\sum a_i x_i < b$ to $\sum a_i x_i \leq b - 1$
  - $2 < 517x + 2y < 3$ can be tightened to $3 \leq 517x + 2y \leq 2$, trivially unsat via simplex

Tightening

- consider inequality $\sum a_i x_i \bowtie b$ with $\bowtie \in \{<, \leq\}$, and $a_1, \dots, a_n, b \in \mathbb{Z}$
- tightening preprocesses such inequality in a way that preserves integer solutions (but removes solutions in $\mathbb{R}$)
- impact: tightening helps to obtain unsatisfiability in $\mathbb{R}$
- last lecture: tightening of strict inequalities $\sum a_i x_i < b$ to $\sum a_i x_i \leq b - 1$
  - $2 < 517x + 2y < 3$ can be tightened to $3 \leq 517x + 2y \leq 2$, trivially unsat via simplex
- tightening of weak inequalities
  - let $g = \gcd(a_1, \dots, a_n)$

Tightening

- consider inequality $\sum a_i x_i \bowtie b$ with $\bowtie \in \{<, \leq\}$, and $a_1, \dots, a_n, b \in \mathbb{Z}$
- tightening preprocesses such inequality in a way that preserves integer solutions (but removes solutions in $\mathbb{R}$)
- impact: tightening helps to obtain unsatisfiability in $\mathbb{R}$
- last lecture: tightening of strict inequalities $\sum a_i x_i < b$ to $\sum a_i x_i \leq b - 1$
  - $2 < 517x + 2y < 3$ can be tightened to $3 \leq 517x + 2y \leq 2$, trivially unsat via simplex
- tightening of weak inequalities
  - let $g = \gcd(a_1, \dots, a_n)$
  - if b is not divisible by g then tighten $\sum a_i x_i \leq b$ to $\sum \frac{a_i}{g} x_i \leq \lfloor \frac{b}{g} \rfloor$

Tightening

- consider inequality $\sum a_i x_i \bowtie b$ with $\bowtie \in \{<, \leq\}$, and $a_1, \dots, a_n, b \in \mathbb{Z}$
- tightening preprocesses such inequality in a way that preserves integer solutions (but removes solutions in $\mathbb{R}$)
- impact: tightening helps to obtain unsatisfiability in $\mathbb{R}$
- last lecture: tightening of strict inequalities $\sum a_i x_i < b$ to $\sum a_i x_i \leq b - 1$
  - $2 < 517x + 2y < 3$ can be tightened to $3 \leq 517x + 2y \leq 2$, trivially unsat via simplex
- tightening of weak inequalities
  - let $g = \gcd(a_1, \dots, a_n)$
  - if b is not divisible by g then tighten $\sum a_i x_i \leq b$ to $\sum \frac{a_i}{g} x_i \leq \lfloor \frac{b}{g} \rfloor$
- example from last lecture
  - tighten $1 \leq 3x - 3y \leq 2$ to $\lceil \frac{1}{3} \rceil \leq x - y \leq \lfloor \frac{2}{3} \rfloor$
  - result $1 \leq x - y \leq 0$ is unsat by simplex

Outline

Motivation

- searching for integral solutions $\vec{x}$ for polyhedron P described as $A\vec{x} \leq \vec{b}$ via branch-and-bound algorithm is expensive
- idea: use sufficient criterion that sometimes quickly finds integral solution of P

Motivation

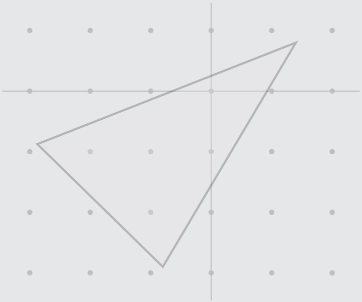
- searching for integral solutions $\vec{x}$ for polyhedron P described as $A\vec{x} \leq \vec{b}$ via branch-and-bound algorithm is expensive
- idea: use sufficient criterion that sometimes quickly finds integral solution of P
- core idea of the cube-test: if there is some cube C with edge-length ≥ 1 that is completely contained in P, then P contains an integral solution

Motivation

- searching for integral solutions $\vec{x}$ for polyhedron P described as $A\vec{x} \leq \vec{b}$ via branch-and-bound algorithm is expensive
- idea: use sufficient criterion that sometimes quickly finds integral solution of P
- core idea of the cube-test: if there is some cube C with edge-length ≥ 1 that is completely contained in P, then P contains an integral solution

Example

- consider polyhedron P, the triangle
- none of the corners is integral, hence BB will require some iterations

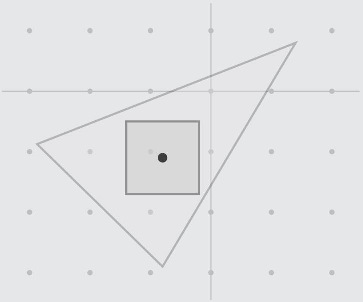


Motivation

- searching for integral solutions $\vec{x}$ for polyhedron P described as $A\vec{x} \leq \vec{b}$ via branch-and-bound algorithm is expensive
- idea: use sufficient criterion that sometimes quickly finds integral solution of P
- core idea of the cube-test: if there is some cube C with edge-length ≥ 1 that is completely contained in P, then P contains an integral solution

Example

- consider polyhedron P, the triangle
- none of the corners is integral, hence BB will require some iterations
- cube C, the square, is contained in P and has edge-length 1.2
- hence C contains an integral solution, which can be calculated from the center point of C



Definition of Cubes

$\text{cube}_s(\vec{z})$ is the cube with center $\vec{z} \in \mathbb{R}^n$ and size $s \in \mathbb{R}_{\geq 0}$

$$\text{cube}_s(\vec{z}) = \{\vec{x} \in \mathbb{R}^n \mid \forall i \in \{1, \ldots, n\}.\ |x_i - z_i| \leq s\}$$

Definition of Cubes

$\text{cube}_s(\vec{z})$ is the cube with center $\vec{z} \in \mathbb{R}^n$ and size $s \in \mathbb{R}_{\geq 0}$

$$\text{cube}_s(\vec{z}) = \{\vec{x} \in \mathbb{R}^n \mid \forall i \in \{1, \ldots, n\}.\ |x_i - z_i| \leq s\}$$

Lemma

if $s \geq 1/2$ then $\text{cube}_s(\vec{z})$ contains an integral vector $\vec{p}$

Definition of Cubes

$\text{cube}_s(\vec{z})$ is the cube with center $\vec{z} \in \mathbb{R}^n$ and size $s \in \mathbb{R}_{\geq 0}$

$$\text{cube}_s(\vec{z}) = \{\vec{x} \in \mathbb{R}^n \mid \forall i \in \{1, \dots, n\}.\ |x_i - z_i| \leq s\}$$

Lemma

if $s \geq 1/2$ *then* $\text{cube}_s(\vec{z})$ *contains an integral vector* $\vec{p}$

Proof

choose $\vec{p}$ where $p_i = \lfloor z_i \rceil$, i.e., rounding z_i to the nearest integer

Example

the center of the cube on slide 9 is $\vec{z} = (-0.8, -1.1)^t$, so we compute $\vec{p} = (-1, -1)^t$

Cube Inclusion

- consider some polyhedron $P = \{\vec{x} \mid A\vec{x} \leq \vec{b}\}$ for some $A \in \mathbb{R}^{m \times n}$ and $\vec{b} \in \mathbb{R}^m$
- we are interested in whether P contains a cube of size s, formally:

$$\exists \vec{z}.\ \text{cube}_s(\vec{z}) \subseteq \{\vec{x} \mid A\vec{x} \leq \vec{b}\} \tag{1}$$

Cube Inclusion

- consider some polyhedron $P = \{\vec{x} \mid A\vec{x} \leq \vec{b}\}$ for some $A \in \mathbb{R}^{m \times n}$ and $\vec{b} \in \mathbb{R}^m$
- we are interested in whether P contains a cube of size s, formally:

$$\exists \vec{z}.\ \mathrm{cube}_s(\vec{z}) \subseteq \{\vec{x} \mid A\vec{x} \leq \vec{b}\} \tag{1}$$

Lemma (Cube Inclusion for Single Inequality)

For a single inequality $\vec{a} \cdot \vec{x} \leq c$ with $\vec{a} \in \mathbb{R}^n$, $c \in \mathbb{R}$ there is the equivalence:

$$\mathrm{cube}_s(\vec{z}) \subseteq \{\vec{x} \mid \vec{a} \cdot \vec{x} \leq c\} \quad \textit{iff} \quad \vec{a} \cdot \vec{z} \leq c - s \sum_{i=1}^{n} |a_i| \tag{2}$$

Cube Inclusion

- consider some polyhedron $P = \{\vec{x} \mid A\vec{x} \leq \vec{b}\}$ for some $A \in \mathbb{R}^{m \times n}$ and $\vec{b} \in \mathbb{R}^m$
- we are interested in whether P contains a cube of size s, formally:

$$\exists \vec{z}.\ \mathrm{cube}_s(\vec{z}) \subseteq \{\vec{x} \mid A\vec{x} \leq \vec{b}\} \tag{1}$$

Lemma (Cube Inclusion for Single Inequality)

For a single inequality $\vec{a} \cdot \vec{x} \leq c$ with $\vec{a} \in \mathbb{R}^n$, $c \in \mathbb{R}$ there is the equivalence:

$$\mathrm{cube}_s(\vec{z}) \subseteq \{\vec{x} \mid \vec{a} \cdot \vec{x} \leq c\} \quad \textit{iff} \quad \vec{a} \cdot \vec{z} \leq c - s \sum_{i=1}^{n} |a_i| \tag{2}$$

Corollary

Cube inclusion (1) is satisfied iff $A\vec{z} \leq \vec{b} - s \cdot \begin{pmatrix} |A_{11}| + \ldots + |A_{1n}| \\ \ldots \\ |A_{m1}| + \ldots + |A_{mn}| \end{pmatrix}$ *has solution* $\vec{z} \in \mathbb{R}^n$

Proof of One Direction of the Cube Inclusion for Single Inequalities

We assume

$$\text{(A)} \quad \vec{x} \in \mathrm{cube}_s(\vec{z}) \qquad \text{and} \qquad \text{(B)} \quad \vec{a} \cdot \vec{z} \leq c - s \sum_{i=1}^{n} |a_i|$$

and prove $\vec{a} \cdot \vec{x} \leq c$ as follows:

$$\vec{a} \cdot \vec{x}$$

Proof of One Direction of the Cube Inclusion for Single Inequalities

We assume

$$(A) \quad \vec{x} \in \text{cube}_s(\vec{z}) \qquad \text{and} \qquad (B) \quad \vec{a} \cdot \vec{z} \leq c - s \sum_{i=1}^{n} |a_i|$$

and prove $\vec{a} \cdot \vec{x} \leq c$ as follows:

$$\vec{a} \cdot \vec{x} = \vec{a} \cdot (\vec{z} + (\vec{x} - \vec{z})) = \vec{a} \cdot \vec{z} + \vec{a} \cdot (\vec{x} - \vec{z})$$

Proof of One Direction of the Cube Inclusion for Single Inequalities

We assume

$$\text{(A)} \quad \vec{x} \in \mathrm{cube}_s(\vec{z}) \qquad \text{and} \qquad \text{(B)} \quad \vec{a} \cdot \vec{z} \leq c - s \sum_{i=1}^{n} |a_i|$$

and prove $\vec{a} \cdot \vec{x} \leq c$ as follows:

$$\begin{aligned} \vec{a} \cdot \vec{x} &= \vec{a} \cdot (\vec{z} + (\vec{x} - \vec{z})) = \vec{a} \cdot \vec{z} + \vec{a} \cdot (\vec{x} - \vec{z}) \\ &\overset{(B)}{\leq} \left(c - s \sum_{i=1}^{n} |a_i| \right) + \vec{a} \cdot (\vec{x} - \vec{z}) \end{aligned}$$

Proof of One Direction of the Cube Inclusion for Single Inequalities

We assume

$$\text{(A)} \quad \vec{x} \in \text{cube}_s(\vec{z}) \qquad \text{and} \qquad \text{(B)} \quad \vec{a} \cdot \vec{z} \leq c - s \sum_{i=1}^{n} |a_i|$$

and prove $\vec{a} \cdot \vec{x} \leq c$ as follows:

$$\begin{aligned}
\vec{a} \cdot \vec{x} &= \vec{a} \cdot (\vec{z} + (\vec{x} - \vec{z})) = \vec{a} \cdot \vec{z} + \vec{a} \cdot (\vec{x} - \vec{z}) \\
&\overset{(B)}{\leq} \left(c - s \sum_{i=1}^{n} |a_i| \right) + \vec{a} \cdot (\vec{x} - \vec{z}) = c - s \sum_{i=1}^{n} |a_i| + \sum_{i=1}^{n} a_i \cdot (x_i - z_i)
\end{aligned}$$

Proof of One Direction of the Cube Inclusion for Single Inequalities

We assume

$$\text{(A)} \quad \vec{x} \in \text{cube}_s(\vec{z}) \qquad \text{and} \qquad \text{(B)} \quad \vec{a} \cdot \vec{z} \le c - s \sum_{i=1}^{n} |a_i|$$

and prove $\vec{a} \cdot \vec{x} \le c$ as follows:

$$\begin{aligned}
\vec{a} \cdot \vec{x} &= \vec{a} \cdot (\vec{z} + (\vec{x} - \vec{z})) = \vec{a} \cdot \vec{z} + \vec{a} \cdot (\vec{x} - \vec{z}) \\
&\overset{(B)}{\le} \left(c - s \sum_{i=1}^{n} |a_i| \right) + \vec{a} \cdot (\vec{x} - \vec{z}) = c - s \sum_{i=1}^{n} |a_i| + \sum_{i=1}^{n} a_i \cdot (x_i - z_i) \\
&\le c - s \sum_{i=1}^{n} |a_i| + \sum_{i=1}^{n} |a_i| \cdot |x_i - z_i|
\end{aligned}$$

Proof of One Direction of the Cube Inclusion for Single Inequalities

We assume

$$\text{(A)} \quad \vec{x} \in \text{cube}_s(\vec{z}) \qquad \text{and} \qquad \text{(B)} \quad \vec{a} \cdot \vec{z} \leq c - s \sum_{i=1}^{n} |a_i|$$

and prove $\vec{a} \cdot \vec{x} \leq c$ as follows:

$$\begin{aligned}
\vec{a} \cdot \vec{x} &= \vec{a} \cdot (\vec{z} + (\vec{x} - \vec{z})) = \vec{a} \cdot \vec{z} + \vec{a} \cdot (\vec{x} - \vec{z}) \\
&\overset{(B)}{\leq} \left(c - s \sum_{i=1}^{n} |a_i| \right) + \vec{a} \cdot (\vec{x} - \vec{z}) = c - s \sum_{i=1}^{n} |a_i| + \sum_{i=1}^{n} a_i \cdot (x_i - z_i) \\
&\leq c - s \sum_{i=1}^{n} |a_i| + \sum_{i=1}^{n} |a_i| \cdot |x_i - z_i| \\
&\overset{(A)}{\leq} c - s \sum_{i=1}^{n} |a_i| + \sum_{i=1}^{n} |a_i| \cdot s = c
\end{aligned}$$

The Unit-Cube Test of Bromberger and Weidenbach

- a unit-cube has edge-length 1, i.e., $s = 1/2$
- testing the unit-cube inclusion is possible via one invocation of simplex by checking existence of $\vec{z}$ for inequalities

$$A\vec{z} \leq \vec{b} - \frac{1}{2} \cdot \begin{pmatrix} |A_{11}| + \ldots + |A_{1n}| \\ \ldots \\ |A_{m1}| + \ldots + |A_{mn}| \end{pmatrix}$$

The Unit-Cube Test of Bromberger and Weidenbach

- a unit-cube has edge-length 1, i.e., $s = 1/2$
- testing the unit-cube inclusion is possible via one invocation of simplex by checking existence of $\vec{z}$ for inequalities

$$A\vec{z} \leq \vec{b} - \frac{1}{2} \cdot \begin{pmatrix} |A_{11}| + \ldots + |A_{1n}| \\ \ldots \\ |A_{m1}| + \ldots + |A_{mn}| \end{pmatrix}$$

Remarks

- the unit-cube test is just a sufficient criterion for integral solution of $A\vec{x} \leq \vec{b}$

The Unit-Cube Test of Bromberger and Weidenbach

- a unit-cube has edge-length 1, i.e., $s = 1/2$
- testing the unit-cube inclusion is possible via one invocation of simplex by checking existence of $\vec{z}$ for inequalities

$$A\vec{z} \leq \vec{b} - \frac{1}{2} \cdot \begin{pmatrix} |A_{11}| + \ldots + |A_{1n}| \\ \ldots \\ |A_{m1}| + \ldots + |A_{mn}| \end{pmatrix}$$

Remarks

- the unit-cube test is just a sufficient criterion for integral solution of $A\vec{x} \leq \vec{b}$
- the unit-cube test always fails on constraints that contain (or imply) equalities, e.g., $\ldots \wedge y' = y - 1 + x_1 \wedge \ldots \wedge x_1 \geq 2x_2 \wedge x_2 \geq x_1 \wedge x_1 \geq 0 \wedge \ldots$

The Unit-Cube Test of Bromberger and Weidenbach

- a unit-cube has edge-length 1, i.e., $s = 1/2$
- testing the unit-cube inclusion is possible via one invocation of simplex by checking existence of $\vec{z}$ for inequalities

$$A\vec{z} \leq \vec{b} - \frac{1}{2} \cdot \begin{pmatrix} |A_{11}| + \ldots + |A_{1n}| \\ \ldots \\ |A_{m1}| + \ldots + |A_{mn}| \end{pmatrix}$$

Remarks

- the unit-cube test is just a sufficient criterion for integral solution of $A\vec{x} \leq \vec{b}$
- the unit-cube test always fails on constraints that contain (or imply) equalities, e.g., $\ldots \wedge y' = y - 1 + x_1 \wedge \ldots \wedge x_1 \geq 2x_2 \wedge x_2 \geq x_1 \wedge x_1 \geq 0 \wedge \ldots$
- increase applicability as follows
  - first detect all implied equalities
  - then eliminate equalities (or detect unsat purely from equalities, e.g., from $2x = 1$)
  - afterwards achieve higher success rate of unit-cube test (and lower bounds for BB)

Outline

Definition (Implied Equalities)

- consider set of inequalities $A\vec{x} \leq \vec{b}$ where the i-th inequality has form $\vec{a}_i \cdot \vec{x} \leq b_i$
- $A\vec{x} \leq \vec{b}$ implies equality $\vec{c} \cdot \vec{x} = d$ if every solution $\vec{x} \in \mathbb{R}^n$ of $A\vec{x} \leq \vec{b}$ satisfies $\vec{c} \cdot \vec{x} = d$

Definition (Implied Equalities)

- consider set of inequalities $A\vec{x} \leq \vec{b}$ where the i-th inequality has form $\vec{a}_i \cdot \vec{x} \leq b_i$
- $A\vec{x} \leq \vec{b}$ implies equality $\vec{c} \cdot \vec{x} = d$ if every solution $\vec{x} \in \mathbb{R}^n$ of $A\vec{x} \leq \vec{b}$ satisfies $\vec{c} \cdot \vec{x} = d$

Observation

if $A\vec{x} < \vec{b}$ is satisfiable, then no equality $\vec{a}_i \cdot \vec{x} = b_i$ is implied

Definition (Implied Equalities)

- consider set of inequalities $A\vec{x} \leq \vec{b}$ where the i-th inequality has form $\vec{a}_i \cdot \vec{x} \leq b_i$
- $A\vec{x} \leq \vec{b}$ implies equality $\vec{c} \cdot \vec{x} = d$ if every solution $\vec{x} \in \mathbb{R}^n$ of $A\vec{x} \leq \vec{b}$ satisfies $\vec{c} \cdot \vec{x} = d$

Observation

if $A\vec{x} < \vec{b}$ is satisfiable, then no equality $\vec{a}_i \cdot \vec{x} = b_i$ is implied

Further Results

- interestingly, also the other direction is satisfied
  - assume $A\vec{x} < \vec{b}$ is unsatisfiable
  - then there is some minimal unsatisfiable subset I such that $\bigwedge_{i \in I} \vec{a}_i \cdot \vec{x} < b_i$ is unsatisfiable (obtained by a single simplex invocation)
  - lemma: for every $i \in I$, the i-th equality $\vec{a}_i \cdot \vec{x} = b_i$ is implied

Definition (Implied Equalities)

- consider set of inequalities $A\vec{x} \leq \vec{b}$ where the i-th inequality has form $\vec{a}_i \cdot \vec{x} \leq b_i$
- $A\vec{x} \leq \vec{b}$ implies equality $\vec{c} \cdot \vec{x} = d$ if every solution $\vec{x} \in \mathbb{R}^n$ of $A\vec{x} \leq \vec{b}$ satisfies $\vec{c} \cdot \vec{x} = d$

Observation

if $A\vec{x} < \vec{b}$ is satisfiable, then no equality $\vec{a}_i \cdot \vec{x} = b_i$ is implied

Further Results

- interestingly, also the other direction is satisfied
  - assume $A\vec{x} < \vec{b}$ is unsatisfiable
  - then there is some minimal unsatisfiable subset I such that $\bigwedge_{i \in I} \vec{a}_i \cdot \vec{x} < b_i$ is unsatisfiable (obtained by a single simplex invocation)
  - lemma: for every $i \in I$, the i-th equality $\vec{a}_i \cdot \vec{x} = b_i$ is implied
- overall: given $A\vec{x} \leq \vec{b}$ with one simplex invocation it is possible to
  - either get access to an implied equality
  - or figure out that no such equality exists

Equality Basis Algorithm of Bromberger and Weidenbach

- the task is to iteratively deduce all equalities for (satisfiable) constraints $A\vec{x} \leq \vec{b}$
- in each iteration, one equality is added and one variable is eliminated

Equality Basis Algorithm of Bromberger and Weidenbach

- the task is to iteratively deduce all equalities for (satisfiable) constraints $A\vec{x} \leq \vec{b}$
- in each iteration, one equality is added and one variable is eliminated
- throughout the algorithm E stores equalities, and $A'\vec{x} \leq \vec{b}'$ is a set of inequalities
- initialize $E := \emptyset$
- delete all trivial inequalities from $A\vec{x} \leq \vec{b}$, the result is the initial A' and $\vec{b}'$

Equality Basis Algorithm of Bromberger and Weidenbach

- the task is to iteratively deduce all equalities for (satisfiable) constraints $A\vec{x} \leq \vec{b}$
- in each iteration, one equality is added and one variable is eliminated
- throughout the algorithm E stores equalities, and $A'\vec{x} \leq \vec{b}'$ is a set of inequalities
- initialize $E := \emptyset$
- delete all trivial inequalities from $A\vec{x} \leq \vec{b}$, the result is the initial A' and $\vec{b}'$
- while $\mathit{simplex}(A'\vec{x} < \vec{b}')$ delivers minimal unsat core I
  - choose any equality $\vec{a}'_i \cdot \vec{x} = b'_i$ for $i \in I$

Equality Basis Algorithm of Bromberger and Weidenbach

- the task is to iteratively deduce all equalities for (satisfiable) constraints $A\vec{x} \leq \vec{b}$
- in each iteration, one equality is added and one variable is eliminated
- throughout the algorithm E stores equalities, and $A'\vec{x} \leq \vec{b}'$ is a set of inequalities
- initialize $E := \emptyset$
- delete all trivial inequalities from $A\vec{x} \leq \vec{b}$, the result is the initial A' and $\vec{b}'$
- while $simplex(A'\vec{x} < \vec{b}')$ delivers minimal unsat core I
  - choose any equality $\vec{a}'_i \cdot \vec{x} = b'_i$ for $i \in I$
  - choose any j such that $A'_{ij} \neq 0$
  - reorder equation to obtain equation (e_j) of the form $x_j = \frac{b_i}{A'_{ij}} - \sum_{k \neq j} \frac{A'_{ik}}{A'_{ij}} x_k$

Equality Basis Algorithm of Bromberger and Weidenbach

- the task is to iteratively deduce all equalities for (satisfiable) constraints $A\vec{x} \leq \vec{b}$
- in each iteration, one equality is added and one variable is eliminated
- throughout the algorithm E stores equalities, and $A'\vec{x} \leq \vec{b}'$ is a set of inequalities
- initialize $E := \emptyset$
- delete all trivial inequalities from $A\vec{x} \leq \vec{b}$, the result is the initial A' and $\vec{b}'$
- while $\mathit{simplex}(A'\vec{x} < \vec{b}')$ delivers minimal unsat core I
  - choose any equality $\vec{a}'_i \cdot \vec{x} = b'_i$ for $i \in I$
  - choose any j such that $A'_{ij} \neq 0$
  - reorder equation to obtain equation (e_j) of the form $x_j = \frac{b_i}{A'_{ij}} - \sum_{k \neq j} \frac{A'_{ik}}{A'_{ij}} x_k$
  - eliminate x_j from $A'\vec{x} \leq \vec{b}'$ and from E by using (e_j) as substitution
  - add (e_j) to E
  - remove trivial equations from $A'\vec{x} \leq \vec{b}'$ after simplifications

Equality Basis Algorithm of Bromberger and Weidenbach

- the task is to iteratively deduce all equalities for (satisfiable) constraints $A\vec{x} \leq \vec{b}$
- in each iteration, one equality is added and one variable is eliminated
- throughout the algorithm E stores equalities, and $A'\vec{x} \leq \vec{b}'$ is a set of inequalities
- initialize $E := \emptyset$
- delete all trivial inequalities from $A\vec{x} \leq \vec{b}$, the result is the initial A' and $\vec{b}'$
- while $\mathit{simplex}(A'\vec{x} < \vec{b}')$ delivers minimal unsat core I
  - choose any equality $\vec{a}'_i \cdot \vec{x} = b'_i$ for $i \in I$
  - choose any j such that $A'_{ij} \neq 0$
  - reorder equation to obtain equation (e_j) of the form $x_j = \frac{b_i}{A'_{ij}} - \sum_{k \neq j} \frac{A'_{ik}}{A'_{ij}} x_k$
  - eliminate x_j from $A'\vec{x} \leq \vec{b}'$ and from E by using (e_j) as substitution
  - add (e_j) to E
  - remove trivial equations from $A'\vec{x} \leq \vec{b}'$ after simplifications
- return E and the final $A'\vec{x} \leq \vec{b}'$

Lemma

- $A\vec{x} \leq \vec{b}$ is equivalent to $E \cup A'\vec{x} \leq \vec{b}'$ throughout the algorithm
- the final set E is even an equality basis, i.e., every implied equality $\vec{c} \cdot \vec{x} = d$ of $A\vec{x} \leq \vec{b}$ is a linear combination of equations in E

Example

- initial constraints

$$
\begin{aligned}
y' - y - x_1 &\leq -1 \\
y - y' + x_1 &\leq 1 \\
2x_2 - x_1 &\leq 0 \\
x_1 - x_2 &\leq 0 \\
-x_1 &\leq 0 \\
z - y + 2y' - x_2 &\leq 5
\end{aligned}
$$

Example

- initial constraints

$$\begin{aligned} y' - y - x_1 &\leq -1 \\ y - y' + x_1 &\leq 1 \\ 2x_2 - x_1 &\leq 0 \\ x_1 - x_2 &\leq 0 \\ -x_1 &\leq 0 \\ z - y + 2y' - x_2 &\leq 5 \end{aligned}$$

- simplex on strict inequalities detects equality $y' - y - x_1 = -1$, so $y' = y + x_1 - 1$

Example

- initial constraints

$$\begin{aligned} y' - y - x_1 &\leq -1 \\ y - y' + x_1 &\leq 1 \\ 2x_2 - x_1 &\leq 0 \\ x_1 - x_2 &\leq 0 \\ -x_1 &\leq 0 \\ z - y + 2y' - x_2 &\leq 5 \end{aligned}$$

- simplex on strict inequalities detects equality $y' - y - x_1 = -1$, so $y' = y + x_1 - 1$
- hence $E = \{y' = y + x_1 - 1\}$ and inequalities become

$(y + x_1 - 1) - y - x_1 \leq -1$	is simplified and deleted	$-1 \leq -1$
$y - (y + x_1 - 1) + x_1 \leq 1$	is simplified and deleted	$1 \leq 1$
$2x_2 - x_1 \leq 0$		
$x_1 - x_2 \leq 0$		
$-x_1 \leq 0$		
$z - y + 2(y + x_1 - 1) - x_2 \leq 5$	is simplified to	$z + y + 2x_1 - x_2 \leq 7$

Example (continued)

- $E = \{y' = y + x_1 - 1\}$ and inequalities $\{2x_2 - x_1 \leq 0,\ x_1 - x_2 \leq 0,\ -x_1 \leq 0,\ z + y + 2x_1 - x_2 \leq 7\}$

Example (continued)

- $E = \{y' = y + x_1 - 1\}$ and inequalities $\{2x_2 - x_1 \leq 0,\ x_1 - x_2 \leq 0,\ -x_1 \leq 0,\ z + y + 2x_1 - x_2 \leq 7\}$
- simplex on strict inequalities detects equality $x_1 - x_2 = 0$, so $x_1 = x_2$

Example (continued)

- $E = \{y' = y + x_1 - 1\}$ and
  inequalities $\{2x_2 - x_1 \leq 0,\ x_1 - x_2 \leq 0,\ -x_1 \leq 0,\ z + y + 2x_1 - x_2 \leq 7\}$
- simplex on strict inequalities detects equality $x_1 - x_2 = 0$, so $x_1 = x_2$
- after substitution and simplification get
  $E = \{y' = y + x_2 - 1, x_1 = x_2\}$ and
  inequalities $\{x_2 \leq 0,\ -x_2 \leq 0,\ z + y + x_2 \leq 7\}$

Example (continued)

- $E = \{y' = y + x_1 - 1\}$ and
  inequalities $\{2x_2 - x_1 \leq 0,\ x_1 - x_2 \leq 0,\ -x_1 \leq 0,\ z + y + 2x_1 - x_2 \leq 7\}$
- simplex on strict inequalities detects equality $x_1 - x_2 = 0$, so $x_1 = x_2$
- after substitution and simplification get
  $E = \{y' = y + x_2 - 1, x_1 = x_2\}$ and
  inequalities $\{x_2 \leq 0,\ -x_2 \leq 0,\ z + y + x_2 \leq 7\}$
- simplex on strict inequalities detects equality $x_2 = 0$

Example (continued)

- $E = \{y' = y + x_1 - 1\}$ and inequalities $\{2x_2 - x_1 \leq 0,\ x_1 - x_2 \leq 0,\ -x_1 \leq 0,\ z + y + 2x_1 - x_2 \leq 7\}$
- simplex on strict inequalities detects equality $x_1 - x_2 = 0$, so $x_1 = x_2$
- after substitution and simplification get $E = \{y' = y + x_2 - 1, x_1 = x_2\}$ and inequalities $\{x_2 \leq 0,\ -x_2 \leq 0,\ z + y + x_2 \leq 7\}$
- simplex on strict inequalities detects equality $x_2 = 0$
- after substitution and simplification get $E = \{y' = y - 1, x_1 = 0, x_2 = 0\}$ and inequalities $\{z + y \leq 7\}$

Example (continued)

- $E = \{y' = y + x_1 - 1\}$ and
  inequalities $\{2x_2 - x_1 \leq 0,\ x_1 - x_2 \leq 0,\ -x_1 \leq 0,\ z + y + 2x_1 - x_2 \leq 7\}$
- simplex on strict inequalities detects equality $x_1 - x_2 = 0$, so $x_1 = x_2$
- after substitution and simplification get
  $E = \{y' = y + x_2 - 1, x_1 = x_2\}$ and
  inequalities $\{x_2 \leq 0,\ -x_2 \leq 0,\ z + y + x_2 \leq 7\}$
- simplex on strict inequalities detects equality $x_2 = 0$
- after substitution and simplification get
  $E = \{y' = y - 1, x_1 = 0, x_2 = 0\}$ and
  inequalities $\{z + y \leq 7\}$
- no further equalities are detected

Example (continued)

- $E = \{y' = y + x_1 - 1\}$ and
  inequalities $\{2x_2 - x_1 \leq 0,\ x_1 - x_2 \leq 0,\ -x_1 \leq 0,\ z + y + 2x_1 - x_2 \leq 7\}$
- simplex on strict inequalities detects equality $x_1 - x_2 = 0$, so $x_1 = x_2$
- after substitution and simplification get
  $E = \{y' = y + x_2 - 1, x_1 = x_2\}$ and
  inequalities $\{x_2 \leq 0,\ -x_2 \leq 0,\ z + y + x_2 \leq 7\}$
- simplex on strict inequalities detects equality $x_2 = 0$
- after substitution and simplification get
  $E = \{y' = y - 1, x_1 = 0, x_2 = 0\}$ and
  inequalities $\{z + y \leq 7\}$
- no further equalities are detected
- finally simplex can be used to find solution of $z + y \leq 7$ for variables y and z, and E determines the values for all other variables

Final Remarks

- the algorithm detects equalities over $\mathbb{R}$
- given the final set E and final inequalities I, one can always easily transform real solutions of I to real solutions of the initial constraints by using E

Outline

Current Situation

- given LIA or LRA constraints φ over variables X, compute set of equations E and inequalities ψ such that
  - $X = Y \uplus Z$

Current Situation

- given LIA or LRA constraints φ over variables X, compute set of equations E and inequalities ψ such that
  - $X = Y \uplus Z$
  - ψ only uses variables in Z
  - every equation in E has form $y = e_y$ with $y \in Y$ and e_y linear expression over variables Z
  - for every $y \in Y$ there is exactly one equation $y = e_y$ in E

Current Situation

- given LIA or LRA constraints φ over variables X, compute set of equations E and inequalities ψ such that
  - $X = Y \uplus Z$
  - ψ only uses variables in Z
  - every equation in E has form $y = e_y$ with $y \in Y$ and e_y linear expression over variables Z
  - for every $y \in Y$ there is exactly one equation $y = e_y$ in E
  - φ is equivalent to $\psi \wedge \bigwedge_{y=e_y \in E} y = e_y$

Current Situation

- given LIA or LRA constraints φ over variables X, compute set of equations E and inequalities ψ such that
  - $X = Y \uplus Z$
  - ψ only uses variables in Z
  - every equation in E has form $y = e_y$ with $y \in Y$ and e_y linear expression over variables Z
  - for every $y \in Y$ there is exactly one equation $y = e_y$ in E
  - φ is equivalent to $\psi \wedge \bigwedge_{y=e_y \in E} y = e_y$
- remarks
  - e_y does not necessarily have integer coefficients, e.g., $y = \frac{3}{2}z + 5$
  - consequently, integer solutions of ψ cannot always be extended to integer solutions of φ

Current Situation

- given LIA or LRA constraints φ over variables X, compute set of equations E and inequalities ψ such that
  - $X = Y \uplus Z$
  - ψ only uses variables in Z
  - every equation in E has form $y = e_y$ with $y \in Y$ and e_y linear expression over variables Z
  - for every $y \in Y$ there is exactly one equation $y = e_y$ in E
  - φ is equivalent to $\psi \wedge \bigwedge_{y=e_y \in E} y = e_y$
- remarks
  - e_y does not necessarily have integer coefficients, e.g., $y = \frac{3}{2}z + 5$
  - consequently, integer solutions of ψ cannot always be extended to integer solutions of φ
    - if $v(z) = 4$ then $v(y) = \frac{3}{2} \cdot 4 + 5 = 9 \in \mathbb{Z}$

Current Situation

- given LIA or LRA constraints φ over variables X, compute set of equations E and inequalities ψ such that
  - $X = Y \uplus Z$
  - ψ only uses variables in Z
  - every equation in E has form $y = e_y$ with $y \in Y$ and e_y linear expression over variables Z
  - for every $y \in Y$ there is exactly one equation $y = e_y$ in E
  - φ is equivalent to $\psi \wedge \bigwedge_{y=e_y \in E} y = e_y$
- remarks
  - e_y does not necessarily have integer coefficients, e.g., $y = \frac{3}{2}z + 5$
  - consequently, integer solutions of ψ cannot always be extended to integer solutions of φ
    - if $v(z) = 4$ then $v(y) = \frac{3}{2} \cdot 4 + 5 = 9 \in \mathbb{Z}$
    - if $v(z) = 1$ then $v(y) = \frac{3}{2} + 9 \notin \mathbb{Z}$

Current Situation

- given LIA or LRA constraints φ over variables X, compute set of equations E and inequalities ψ such that
  - $X = Y \uplus Z$
  - ψ only uses variables in Z
  - every equation in E has form $y = e_y$ with $y \in Y$ and e_y linear expression over variables Z
  - for every $y \in Y$ there is exactly one equation $y = e_y$ in E
  - φ is equivalent to $\psi \wedge \bigwedge_{y=e_y \in E} y = e_y$
- remarks
  - e_y does not necessarily have integer coefficients, e.g., $y = \frac{3}{2}z + 5$
  - consequently, integer solutions of ψ cannot always be extended to integer solutions of φ
    - if $v(z) = 4$ then $v(y) = \frac{3}{2} \cdot 4 + 5 = 9 \in \mathbb{Z}$
    - if $v(z) = 1$ then $v(y) = \frac{3}{2} + 9 \notin \mathbb{Z}$
- upcoming: algorithm to convert E in a way that
  - integer solutions can always be extended via resulting equations, or
  - it is detected that E itself is not solvable in the integers
  - remark: additional variables may be required, hence only obtain equisatisfiability

Diophantine Equation Solver of Griggio – Setup

- input: set E of linear equalities
- output: "unsat" or "sat" with list of equations in solved form
- solved form
  - list of equations of the shape $x = e_x$ with e_x linear expression with integer coefficients
  - no cyclic dependencies: e_x in list $[\dots, x = e_x, \dots, y = e_y, \dots]$ does neither contain x nor y

Diophantine Equation Solver of Griggio – Setup

- input: set E of linear equalities
- output: “unsat” or “sat” with list of equations in solved form
- solved form
  - list of equations of the shape $x = e_x$ with e_x linear expression with integer coefficients
  - no cyclic dependencies: e_x in list $[\dots, x = e_x, \dots, y = e_y, \dots]$ does neither contain x nor y
- S is list of equations that will become part of solution, initially $S = [\,]$

Diophantine Equation Solver of Griggio – Setup

- input: set E of linear equalities
- output: "unsat" or "sat" with list of equations in solved form
- solved form
  - list of equations of the shape $x = e_x$ with e_x linear expression with integer coefficients
  - no cyclic dependencies: e_x in list $[\dots, x = e_x, \dots, y = e_y, \dots]$ does neither contain x nor y
- S is list of equations that will become part of solution, initially $S = [\,]$
- F is a set of equations that need to be processed, initially $F = E$

Diophantine Equation Solver of Griggio – Setup

- input: set E of linear equalities
- output: “unsat” or “sat” with list of equations in solved form
- solved form
  - list of equations of the shape $x = e_x$ with e_x linear expression with integer coefficients
  - no cyclic dependencies: e_x in list $[\dots, x = e_x, \dots, y = e_y, \dots]$ does neither contain x nor y
- S is list of equations that will become part of solution, initially $S = [\,]$
- F is a set of equations that need to be processed, initially $F = E$
- processing is done equation by equation, and the selected equation is marked; initially no equation is marked

Diophantine Equation Solver of Griggio – Setup

- input: set E of linear equalities
- output: "unsat" or "sat" with list of equations in solved form
- solved form
  - list of equations of the shape $x = e_x$ with e_x linear expression with integer coefficients
  - no cyclic dependencies: e_x in list $[\dots, x = e_x, \dots, y = e_y, \dots]$ does neither contain x nor y
- S is list of equations that will become part of solution, initially $S = [\,]$
- F is a set of equations that need to be processed, initially $F = E$
- processing is done equation by equation, and the selected equation is marked; initially no equation is marked
- *normalize* is a sub-algorithm that transforms each equation into form $\sum a_i x_i = b$ with $a_1, \dots, a_n, b \in \mathbb{Z}$, $\gcd(a_1, \dots, a_n, b) = 1$
  - $\mathit{normalize}(2x - 6y - 14 = 0) = (x - 3y = 7)$
  - $\mathit{normalize}(x = \frac{1}{2}y + \frac{2}{3}) = (6x - 3y = 4)$

Diophantine Equation Solver of Griggio – Algorithm

1. if $F = \emptyset$ then return "sat" with solved form S

Diophantine Equation Solver of Griggio – Algorithm

1. if $F = \emptyset$ then return "sat" with solved form S
2. $F := \mathit{normalize}(F)$, each equation has form $\sum a_i x_i = b$ with $a_1, \dots, a_n, b \in \mathbb{Z}$

Diophantine Equation Solver of Griggio – Algorithm

1. if $F = \emptyset$ then return "sat" with solved form S
2. $F := \mathit{normalize}(F)$, each equation has form $\sum a_i x_i = b$ with $a_1, \ldots, a_n, b \in \mathbb{Z}$
3. if for some $\sum a_i x_i = b \in F$ the $\gcd(a_1, \ldots, a_n)$ does not divide b, return "unsat"

Diophantine Equation Solver of Griggio – Algorithm

1. if $F = \emptyset$ then return “sat” with solved form S
2. $F := \mathit{normalize}(F)$, each equation has form $\sum a_i x_i = b$ with $a_1, \dots, a_n, b \in \mathbb{Z}$
3. if for some $\sum a_i x_i = b \in F$ the $\gcd(a_1, \dots, a_n)$ does not divide b, return “unsat”
4. selection: if no equation is marked, then mark one; assume it is $\sum a_i x_i = b$

Diophantine Equation Solver of Griggio – Algorithm

1. if $F = \emptyset$ then return "sat" with solved form S
2. $F := \mathit{normalize}(F)$, each equation has form $\sum a_i x_i = b$ with $a_1, \dots, a_n, b \in \mathbb{Z}$
3. if for some $\sum a_i x_i = b \in F$ the $\gcd(a_1, \dots, a_n)$ does not divide b, return "unsat"
4. selection: if no equation is marked, then mark one; assume it is $\sum a_i x_i = b$
5. choose k such that a_k has smallest absolute value among all a_i

Diophantine Equation Solver of Griggio – Algorithm

1. if $F = \emptyset$ then return "sat" with solved form S
2. $F := \mathit{normalize}(F)$, each equation has form $\sum a_i x_i = b$ with $a_1, \ldots, a_n, b \in \mathbb{Z}$
3. if for some $\sum a_i x_i = b \in F$ the $\gcd(a_1, \ldots, a_n)$ does not divide b, return "unsat"
4. selection: if no equation is marked, then mark one; assume it is $\sum a_i x_i = b$
5. choose k such that a_k has smallest absolute value among all a_i
6. if $|a_k| = 1$

Diophantine Equation Solver of Griggio – Algorithm

1. if $F = \emptyset$ then return "sat" with solved form S
2. $F := normalize(F)$, each equation has form $\sum a_i x_i = b$ with $a_1, \ldots, a_n, b \in \mathbb{Z}$
3. if for some $\sum a_i x_i = b \in F$ the $\gcd(a_1, \ldots, a_n)$ does not divide b, return "unsat"
4. selection: if no equation is marked, then mark one; assume it is $\sum a_i x_i = b$
5. choose k such that a_k has smallest absolute value among all a_i
6. if $|a_k| = 1$
   - remove marked equation from F

Diophantine Equation Solver of Griggio – Algorithm

1. if $F = \emptyset$ then return “sat” with solved form S
2. $F := \mathit{normalize}(F)$, each equation has form $\sum a_i x_i = b$ with $a_1, \ldots, a_n, b \in \mathbb{Z}$
3. if for some $\sum a_i x_i = b \in F$ the $\gcd(a_1, \ldots, a_n)$ does not divide b, return “unsat”
4. selection: if no equation is marked, then mark one; assume it is $\sum a_i x_i = b$
5. choose k such that a_k has smallest absolute value among all a_i
6. if $|a_k| = 1$
   - remove marked equation from F
   - reorder equation to have shape $(eq): \; x_k = \pm(b - \sum_{i \neq k} a_i x_i)$

Diophantine Equation Solver of Griggio – Algorithm

1. if $F = \emptyset$ then return "sat" with solved form S
2. $F := \mathit{normalize}(F)$, each equation has form $\sum a_i x_i = b$ with $a_1, \ldots, a_n, b \in \mathbb{Z}$
3. if for some $\sum a_i x_i = b \in F$ the $\gcd(a_1, \ldots, a_n)$ does not divide b, return "unsat"
4. selection: if no equation is marked, then mark one; assume it is $\sum a_i x_i = b$
5. choose k such that a_k has smallest absolute value among all a_i
6. if $|a_k| = 1$
   - remove marked equation from F
   - reorder equation to have shape $(eq):\ x_k = \pm(b - \sum_{i \neq k} a_i x_i)$
   - $S := (eq) : S$

Diophantine Equation Solver of Griggio – Algorithm

1. if $F = \emptyset$ then return "sat" with solved form S
2. $F := \mathit{normalize}(F)$, each equation has form $\sum a_i x_i = b$ with $a_1, \ldots, a_n, b \in \mathbb{Z}$
3. if for some $\sum a_i x_i = b \in F$ the $\gcd(a_1, \ldots, a_n)$ does not divide b, return "unsat"
4. selection: if no equation is marked, then mark one; assume it is $\sum a_i x_i = b$
5. choose k such that a_k has smallest absolute value among all a_i
6. if $|a_k| = 1$
   - remove marked equation from F
   - reorder equation to have shape $(eq):\ x_k = \pm(b - \sum_{i \neq k} a_i x_i)$
   - $S := (eq) : S$
   - use (eq) as substitution to eliminate x_k in F

Diophantine Equation Solver of Griggio – Algorithm

1. if $F = \emptyset$ then return "sat" with solved form S
2. $F := \mathit{normalize}(F)$, each equation has form $\sum a_i x_i = b$ with $a_1, \ldots, a_n, b \in \mathbb{Z}$
3. if for some $\sum a_i x_i = b \in F$ the $\gcd(a_1, \ldots, a_n)$ does not divide b, return "unsat"
4. selection: if no equation is marked, then mark one; assume it is $\sum a_i x_i = b$
5. choose k such that a_k has smallest absolute value among all a_i
6. if $|a_k| = 1$
   - remove marked equation from F
   - reorder equation to have shape $(eq): \; x_k = \pm(b - \sum_{i \neq k} a_i x_i)$
   - $S := (eq) : S$
   - use (eq) as substitution to eliminate x_k in F
   - continue with step 1

Diophantine Equation Solver of Griggio – Algorithm

1. if $F = \emptyset$ then return "sat" with solved form S
2. $F := normalize(F)$, each equation has form $\sum a_i x_i = b$ with $a_1, \dots, a_n, b \in \mathbb{Z}$
3. if for some $\sum a_i x_i = b \in F$ the $\gcd(a_1, \dots, a_n)$ does not divide b, return "unsat"
4. selection: if no equation is marked, then mark one; assume it is $\sum a_i x_i = b$
5. choose k such that a_k has smallest absolute value among all a_i
6. if $|a_k| = 1$
   - remove marked equation from F
   - reorder equation to have shape $(eq):\ x_k = \pm(b - \sum_{i \neq k} a_i x_i)$
   - $S := (eq) : S$
   - use (eq) as substitution to eliminate x_k in F
   - continue with step 1
7. otherwise, handle case $|a_k| > 1$ (cf. slide 26) and continue with step 1

Example

- $F = \{x = \frac{1}{6}y + 1, 4z + y = 2\}, S = [\,]$

Example

- $F = \{x = \frac{1}{6}y + 1, 4z + y = 2\}, S = [\,]$
- normalization: $F = \{6x - y = 6, 4z + y = 2\}$

Example

- $F = \{x = \frac{1}{6}y + 1, 4z + y = 2\}, S = [\,]$
- normalization: $F = \{6x - y = 6, 4z + y = 2\}$
- selection: $6x - y = 6$ gets marked

Example

- $F = \{x = \frac{1}{6}y + 1, 4z + y = 2\}$, $S = [\,]$
- normalization: $F = \{6x - y = 6, 4z + y = 2\}$
- selection: $6x - y = 6$ gets marked
- reordering: $y = 6x - 6$

Example

- $F = \{x = \frac{1}{6}y + 1, 4z + y = 2\}$, $S = [\,]$
- normalization: $F = \{6x - y = 6, 4z + y = 2\}$
- selection: $6x - y = 6$ gets marked
- reordering: $y = 6x - 6$
- updates: $S = [y = 6x - 6]$, $F = \{4z + (6x - 6) = 2\}$

Example

- $F = \{x = \frac{1}{6}y + 1, 4z + y = 2\}$, $S = [\,]$
- normalization: $F = \{6x - y = 6, 4z + y = 2\}$
- selection: $6x - y = 6$ gets marked
- reordering: $y = 6x - 6$
- updates: $S = [y = 6x - 6]$, $F = \{4z + (6x - 6) = 2\}$
- normalization: $F = \{2z + 3x = 4\}$

Example

- $F = \{x = \frac{1}{6}y + 1, 4z + y = 2\}$, $S = [\,]$
- normalization: $F = \{6x - y = 6, 4z + y = 2\}$
- selection: $6x - y = 6$ gets marked
- reordering: $y = 6x - 6$
- updates: $S = [y = 6x - 6]$, $F = \{4z + (6x - 6) = 2\}$
- normalization: $F = \{2z + 3x = 4\}$
- selection: $2z + 3x = 4$ gets marked

Example

- $F = \{x = \frac{1}{6}y + 1, 4z + y = 2\}$, $S = [\,]$
- normalization: $F = \{6x - y = 6, 4z + y = 2\}$
- selection: $6x - y = 6$ gets marked
- reordering: $y = 6x - 6$
- updates: $S = [y = 6x - 6]$, $F = \{4z + (6x - 6) = 2\}$
- normalization: $F = \{2z + 3x = 4\}$
- selection: $2z + 3x = 4$ gets marked
- step 7 required

Diophantine Equation Solver – Case $|a_k| > 1$

- we can write any integer number c as $c^q a_k + c^r$ where c^q and c^r are quotient and remainder when dividing c by a_k
- hence

$$\begin{aligned}\sum a_i x_i = b &\equiv a_k x_k + \sum_{i \neq k} a_i x_i = b \\ &\equiv a_k x_k + \sum_{i \neq k} (a_i^q a_k + a_i^r) x_i = b^q a_k + b^r\end{aligned}$$

Diophantine Equation Solver – Case $|a_k| > 1$

- we can write any integer number c as $c^q a_k + c^r$ where c^q and c^r are quotient and remainder when dividing c by a_k
- hence

$$\begin{aligned}\sum a_i x_i = b &\equiv a_k x_k + \sum_{i \neq k} a_i x_i = b \\ &\equiv a_k x_k + \sum_{i \neq k} (a_i^q a_k + a_i^r) x_i = b^q a_k + b^r \\ &\equiv a_k \left(x_k + (\sum_{i \neq k} a_i^q x_i) - b^q\right) + \sum_{i \neq k} a_i^r x_i = b^r\end{aligned}$$

Diophantine Equation Solver – Case $|a_k| > 1$

- we can write any integer number c as $c^q a_k + c^r$ where c^q and c^r are quotient and remainder when dividing c by a_k
- hence
$$\begin{aligned} \sum a_i x_i = b &\equiv a_k x_k + \sum_{i \neq k} a_i x_i = b \\ &\equiv a_k x_k + \sum_{i \neq k} (a_i^q a_k + a_i^r) x_i = b^q a_k + b^r \\ &\equiv a_k \Big(x_k + \big(\sum_{i \neq k} a_i^q x_i\big) - b^q \Big) + \sum_{i \neq k} a_i^r x_i = b^r \end{aligned}$$
- introduce fresh variable x_t to obtain equation (eq) that is always solvable

$$x_k = -\Big(\big(\sum_{i \neq k} a_i^q x_i\big) - b^q\Big) + x_t \qquad \text{(eq)}$$

Diophantine Equation Solver – Case $|a_k| > 1$

- we can write any integer number c as $c^q a_k + c^r$ where c^q and c^r are quotient and remainder when dividing c by a_k
- hence

$$\begin{aligned}\sum a_i x_i = b &\equiv a_k x_k + \sum_{i \neq k} a_i x_i = b \\ &\equiv a_k x_k + \sum_{i \neq k} (a_i^q a_k + a_i^r) x_i = b^q a_k + b^r \\ &\equiv a_k \Big(x_k + \big(\sum_{i \neq k} a_i^q x_i\big) - b^q \Big) + \sum_{i \neq k} a_i^r x_i = b^r\end{aligned}$$

- introduce fresh variable x_t to obtain equation (eq) that is always solvable

$$x_k = -\Big(\big(\sum_{i \neq k} a_i^q x_i\big) - b^q\Big) + x_t \tag{eq}$$

- update $S := (eq) : S$ and eliminate x_k in F by substituting with (eq) as in step 6 (note that the marker stays on the previously marked equation)

Example (continued)

- marked equation is $2z + 3x = 4$, i.e., $2(z + (x - 2)) + x = 0$

Example (continued)

- marked equation is $2z + 3x = 4$, i.e., $2(z + (x - 2)) + x = 0$
- introduce fresh variable u and add equation to S, so
  $S = [z = -(x - 2) + u,\ y = 6x - 6]$

Example (continued)

- marked equation is $2z + 3x = 4$, i.e., $2(z + (x - 2)) + x = 0$
- introduce fresh variable u and add equation to S, so
  $S = [z = -(x - 2) + u,\; y = 6x - 6]$
- substituting in F delivers $2(-(x - 2) + (x - 2) + u) + x = 0$, i.e., $F = \{2u + x = 0\}$
  (and the marker is still on this equation)

Example (continued)

- marked equation is $2z + 3x = 4$, i.e., $2(z + (x - 2)) + x = 0$
- introduce fresh variable u and add equation to S, so $S = [z = -(x - 2) + u,\ y = 6x - 6]$
- substituting in F delivers $2(-(x - 2) + (x - 2) + u) + x = 0$, i.e., $F = \{2u + x = 0\}$ (and the marker is still on this equation)
- reorder $2u + x = 0$ to $x = -2u$ and update S and F, so $S = [x = -2u,\ z = -(x - 2) + u,\ y = 6x - 6]$ and $F = \emptyset$

Example (continued)

- marked equation is $2z + 3x = 4$, i.e., $2(z + (x - 2)) + x = 0$
- introduce fresh variable u and add equation to S, so
  $S = [z = -(x - 2) + u,\ y = 6x - 6]$
- substituting in F delivers $2(-(x - 2) + (x - 2) + u) + x = 0$, i.e., $F = \{2u + x = 0\}$
  (and the marker is still on this equation)
- reorder $2u + x = 0$ to $x = -2u$ and update S and F, so
  $S = [x = -2u,\ z = -(x - 2) + u,\ y = 6x - 6]$ and $F = \emptyset$
- final result
  - initial equations $\{x = \frac{1}{6}y + 1, 4z + y = 2\}$ are equisatisfiable to final S,
  - S is trivially solvable as it is in solved form

Example (continued)

- marked equation is $2z + 3x = 4$, i.e., $2(z + (x - 2)) + x = 0$
- introduce fresh variable u and add equation to S, so
  $S = [z = -(x - 2) + u,\ y = 6x - 6]$
- substituting in F delivers $2(-(x - 2) + (x - 2) + u) + x = 0$, i.e., $F = \{2u + x = 0\}$
  (and the marker is still on this equation)
- reorder $2u + x = 0$ to $x = -2u$ and update S and F, so
  $S = [x = -2u,\ z = -(x - 2) + u,\ y = 6x - 6]$ and $F = \emptyset$
- final result
  - initial equations $\{x = \frac{1}{6}y + 1, 4z + y = 2\}$ are equisatisfiable to final S,
  - S is trivially solvable as it is in solved form
  - note that S can be applied in two ways
    - on symbolic values: translate constraints (e.g., inequalities)
      apply equations starting with end of S

Example (continued)

- marked equation is $2z + 3x = 4$, i.e., $2(z + (x - 2)) + x = 0$
- introduce fresh variable u and add equation to S, so
  $S = [z = -(x - 2) + u,\ y = 6x - 6]$
- substituting in F delivers $2(-(x - 2) + (x - 2) + u) + x = 0$, i.e., $F = \{2u + x = 0\}$
  (and the marker is still on this equation)
- reorder $2u + x = 0$ to $x = -2u$ and update S and F, so
  $S = [x = -2u,\ z = -(x - 2) + u,\ y = 6x - 6]$ and $F = \emptyset$
- final result
  - initial equations $\{x = \frac{1}{6}y + 1, 4z + y = 2\}$ are equisatisfiable to final S,
  - S is trivially solvable as it is in solved form
  - note that S can be applied in two ways
    - on symbolic values: translate constraints (e.g., inequalities)
      apply equations starting with end of S
    - on concrete value: translate solution (e.g., from a solution of inequalities)
      compute values by equations starting at beginning of S

Theorem

Let E be a set of linear equations

- *the Diophantine equation solver terminates on input E*

Theorem

Let E be a set of linear equations

- *the Diophantine equation solver terminates on input E*
- *if the result on input E is "unsat", then E is not solvable in $\mathbb{Z}$*

Theorem

Let E be a set of linear equations

- *the Diophantine equation solver terminates on input E*
- *if the result on input E is "unsat", then E is not solvable in $\mathbb{Z}$*
- *if the result on input E is "sat" with answer S, then*
  - *E and S are equisatisfiable in $\mathbb{Z}$,*
  - *S is in solved form, and*
  - *constraints and solutions can be translated via S*

Application: Combine Algorithms for an Improved LIA Solver

input: set of linear inequalities

Application: Combine Algorithms for an Improved LIA Solver

input: set of linear inequalities

1. tighten bounds if possible (Section 2)

Application: Combine Algorithms for an Improved LIA Solver

input: set of linear inequalities

1. tighten bounds if possible (Section 2)
2. split inequalities into equalities E and inequalities I (Section 4)

Application: Combine Algorithms for an Improved LIA Solver

input: set of linear inequalities

1. tighten bounds if possible (Section 2)
2. split inequalities into equalities E and inequalities I (Section 4)
3. if Diophantine equation solver on E returns “unsat”, return “unsat” (Section 5)

Application: Combine Algorithms for an Improved LIA Solver

input: set of linear inequalities

1. tighten bounds if possible (Section 2)
2. split inequalities into equalities E and inequalities I (Section 4)
3. if Diophantine equation solver on E returns “unsat”, return “unsat” (Section 5)
4. otherwise obtain solution S and apply substitution S on I (symbolically)

Application: Combine Algorithms for an Improved LIA Solver

input: set of linear inequalities

1. tighten bounds if possible (Section 2)
2. split inequalities into equalities E and inequalities I (Section 4)
3. if Diophantine equation solver on E returns “unsat”, return “unsat” (Section 5)
4. otherwise obtain solution S and apply substitution S on I (symbolically)
5. try to tighten I again

Application: Combine Algorithms for an Improved LIA Solver

input: set of linear inequalities

1. tighten bounds if possible (Section 2)
2. split inequalities into equalities E and inequalities I (Section 4)
3. if Diophantine equation solver on E returns "unsat", return "unsat" (Section 5)
4. otherwise obtain solution S and apply substitution S on I (symbolically)
5. try to tighten I again
6. potentially detect new equalities and go back to step 2

Application: Combine Algorithms for an Improved LIA Solver

input: set of linear inequalities

1. tighten bounds if possible (Section 2)
2. split inequalities into equalities E and inequalities I (Section 4)
3. if Diophantine equation solver on E returns “unsat”, return “unsat” (Section 5)
4. otherwise obtain solution S and apply substitution S on I (symbolically)
5. try to tighten I again
6. potentially detect new equalities and go back to step 2
7. if simplex on I returns “unsat”, return “unsat”

Application: Combine Algorithms for an Improved LIA Solver

input: set of linear inequalities

1. tighten bounds if possible (Section 2)
2. split inequalities into equalities E and inequalities I (Section 4)
3. if Diophantine equation solver on E returns "unsat", return "unsat" (Section 5)
4. otherwise obtain solution S and apply substitution S on I (symbolically)
5. try to tighten I again
6. potentially detect new equalities and go back to step 2
7. if simplex on I returns "unsat", return "unsat"
8. if unit-cube test on I finds solution then store it and go to step 10 (Section 3)

Application: Combine Algorithms for an Improved LIA Solver

input: set of linear inequalities

1. tighten bounds if possible (Section 2)
2. split inequalities into equalities E and inequalities I (Section 4)
3. if Diophantine equation solver on E returns "unsat", return "unsat" (Section 5)
4. otherwise obtain solution S and apply substitution S on I (symbolically)
5. try to tighten I again
6. potentially detect new equalities and go back to step 2
7. if simplex on I returns "unsat", return "unsat"
8. if unit-cube test on I finds solution then store it and go to step 10 (Section 3)
9. apply branch-and-bound on I and either store solution or return "unsat"

Application: Combine Algorithms for an Improved LIA Solver

input: set of linear inequalities

1. tighten bounds if possible (Section 2)
2. split inequalities into equalities E and inequalities I (Section 4)
3. if Diophantine equation solver on E returns “unsat”, return “unsat” (Section 5)
4. otherwise obtain solution S and apply substitution S on I (symbolically)
5. try to tighten I again
6. potentially detect new equalities and go back to step 2
7. if simplex on I returns “unsat”, return “unsat”
8. if unit-cube test on I finds solution then store it and go to step 10 (Section 3)
9. apply branch-and-bound on I and either store solution or return “unsat”
10. output solution where S is used to compute values of further variables

Example (for improved LIA solver)

- input

$$\begin{aligned} 2x_1 &\leq 5x_3 \\ 5x_3 &\leq 2x_1 \\ x_2 &= 3x_4 \\ 2x_1 + x_2 + x_3 &\geq 7 \\ 2x_1 + x_2 + x_3 &\leq 8 \end{aligned}$$

Example (for improved LIA solver)

- input

$$\begin{aligned} 2x_1 &\leq 5x_3 \\ 5x_3 &\leq 2x_1 \\ x_2 &= 3x_4 \\ 2x_1 + x_2 + x_3 &\geq 7 \\ 2x_1 + x_2 + x_3 &\leq 8 \end{aligned}$$

- tightening is not applicable on input constraints

Example (for improved LIA solver)

- input

$$\begin{aligned} 2x_1 &\leq 5x_3 \\ 5x_3 &\leq 2x_1 \\ x_2 &= 3x_4 \\ 2x_1 + x_2 + x_3 &\geq 7 \\ 2x_1 + x_2 + x_3 &\leq 8 \end{aligned}$$

- tightening is not applicable on input constraints
- equation detection splits constraints into

$$E = \{2x_1 = 5x_3, x_2 = 3x_4\}$$

and

$$I = \{7 \leq 2x_1 + x_2 + x_3 \leq 8\}$$

Example (continued)

- Diophantine equation solver on $E = \{2x_1 = 5x_3, x_2 = 3x_4\}$ delivers solved form S and introduces new variable x_5

$$S = [x_3 = 2x_5,\ x_1 = 2x_3 + x_5,\ x_2 = 3x_4]$$

Example (continued)

- Diophantine equation solver on $E = \{2x_1 = 5x_3, x_2 = 3x_4\}$ delivers solved form S and introduces new variable x_5

$$S = [x_3 = 2x_5,\ x_1 = 2x_3 + x_5,\ x_2 = 3x_4]$$

- S is used as substitution to change $I = \{7 \leq 2x_1 + x_2 + x_3 \leq 8\}$ into

$$\begin{aligned} I &= \{7 \leq 2(2(2x_5) + x_5) + 3x_4 + 2x_5 \leq 8\} \\ &= \{7 \leq 12x_5 + 3x_4 \leq 8\} \end{aligned}$$

Example (continued)

- Diophantine equation solver on $E = \{2x_1 = 5x_3, x_2 = 3x_4\}$ delivers solved form S and introduces new variable x_5

$$S = [x_3 = 2x_5,\ x_1 = 2x_3 + x_5,\ x_2 = 3x_4]$$

- S is used as substitution to change $I = \{7 \leq 2x_1 + x_2 + x_3 \leq 8\}$ into

$$\begin{aligned} I &= \{7 \leq 2(2(2x_5) + x_5) + 3x_4 + 2x_5 \leq 8\} \\ &= \{7 \leq 12x_5 + 3x_4 \leq 8\} \end{aligned}$$

- tightening is now applicable and yields

$$I = \{3 \leq 4x_5 + x_4 \leq 2\}$$

Example (continued)

- Diophantine equation solver on $E = \{2x_1 = 5x_3, x_2 = 3x_4\}$ delivers solved form S and introduces new variable x_5

$$S = [x_3 = 2x_5,\ x_1 = 2x_3 + x_5,\ x_2 = 3x_4]$$

- S is used as substitution to change $I = \{7 \leq 2x_1 + x_2 + x_3 \leq 8\}$ into

$$\begin{aligned} I &= \{7 \leq 2(2(2x_5) + x_5) + 3x_4 + 2x_5 \leq 8\} \\ &= \{7 \leq 12x_5 + 3x_4 \leq 8\} \end{aligned}$$

- tightening is now applicable and yields

$$I = \{3 \leq 4x_5 + x_4 \leq 2\}$$

- simplex now easily detects unsat of I

Example (continued)

- Diophantine equation solver on $E = \{2x_1 = 5x_3, x_2 = 3x_4\}$ delivers solved form S and introduces new variable x_5

$$S = [x_3 = 2x_5,\ x_1 = 2x_3 + x_5,\ x_2 = 3x_4]$$

- S is used as substitution to change $I = \{7 \leq 2x_1 + x_2 + x_3 \leq 8\}$ into

$$\begin{aligned} I &= \{7 \leq 2(2(2x_5) + x_5) + 3x_4 + 2x_5 \leq 8\} \\ &= \{7 \leq 12x_5 + 3x_4 \leq 8\} \end{aligned}$$

- tightening is now applicable and yields

$$I = \{3 \leq 4x_5 + x_4 \leq 2\}$$

- simplex now easily detects unsat of I
- (solution of I for x_4 and x_5 would be extensible to x_3, x_1, x_2 via S)

Final Remarks

- easy to see: T-solver for LIA is much more complex than one for LRA

Final Remarks

- easy to see: T-solver for LIA is much more complex than one for LRA
- not discussed
  - how to make LIA-solver incremental in DPLL(T) approach
  - how to generate small unsatisfiable cores for DPLL(T) approach

Final Remarks

- easy to see: T-solver for LIA is much more complex than one for LRA
- not discussed
  - how to make LIA-solver incremental in DPLL(T) approach
  - how to generate small unsatisfiable cores for DPLL(T) approach
- strategy used in DPLL(T) solvers for LIA
  - utilize (incomplete) LRA-reasoning as much as possible, and
  - only invoke full LIA-solver when a complete Boolean model has been found

Outline

Further Reading

- Alberto Griggio
  A Practical Approach to Satisfiability Modulo Linear Integer Arithmetic
  Journal on Satisfiability, Boolean Modeling and Computation, volume 8, pages 1–27, 2012.

- Martin Bromberger and Christoph Weidenbach
  New techniques for linear arithmetic: cubes and equalities
  Formal Methods in System Design, volume 51, pages 433–461, 2017.

Important Concepts

- cube
- Diophantine equation solver
- equality basis algorithm
- implied equality
- tightening