A Polyhedral Compilation Library with Explicit Disequality Constraints

Sven Verdoolaege

Cerebras Systems

January 17, 2024

Outline

Outline

A Motivating Example (Kulkarni and Kruse June 2022)

```
for (int i = 0; i < n; i+=1) {
  if (i == p0)
    continue;
  if (i == p1)
    continue;
  if (i == p2)
    continue;
  // ...
  Stmt(i);
}
```

A Motivating Example (Kulkarni and Kruse June 2022)

```
for (int i = 0; i < n; i+=1) {
  if (i == p0)
    continue;
  if (i == p1)
    continue;
  if (i == p2)
    continue;
  // ...
  Stmt(i);
}
```

Instance set: $\{\,\mathrm{Stmt}[i] : 0 \leq i < \mathrm{n} \wedge i \neq \mathrm{p0} \wedge i \neq \mathrm{p1} \wedge i \neq \mathrm{p2} \wedge \ldots\}$

A Motivating Example (Kulkarni and Kruse June 2022)

```
for (int i = 0; i < n; i+=1) {
  if (i == p0)
    continue;
  if (i == p1)
    continue;
  if (i == p2)
    continue;
  // ...
  Stmt(i);
}
```

Instance set: $\{\, \mathrm{Stmt}[i] : 0 \leq i < \mathrm{n} \wedge i \neq \mathrm{p0} \wedge i \neq \mathrm{p1} \wedge i \neq \mathrm{p2} \wedge \ldots \}$

$\{\, \mathrm{Stmt}[i] : 0 \leq i < \mathrm{n} \wedge (i < \mathrm{p0} \vee i > \mathrm{p0}) \wedge (i < \mathrm{p1} \vee i > \mathrm{p1}) \wedge (i < \mathrm{p2} \vee i > \mathrm{p2}) \wedge \ldots \}$

$\Rightarrow$ expansion causes explosion in representation

Another Motivating Example (Klebanov 2015)

card { [r1]; [r2]; [r3]; [r4]; [r5]; [r10] }

Another Motivating Example (Klebanov 2015)

$$\text{card} \, \{ \, [\text{r1}]; [\text{r2}]; [\text{r3}]; [\text{r4}]; [\text{r5}]; [\text{r10}] \, \}$$

$$\begin{cases} 6 & \text{if } \text{r1} \neq \text{r2}, \text{r3}, \text{r4}, \text{r5}, \text{r10} \wedge \text{r2} \neq \text{r3}, \text{r4}, \text{r5}, \text{r10} \wedge \text{r3} \neq \text{r4}, \text{r5}, \text{r10} \wedge \text{r4} \neq \text{r5}, \text{r10} \wedge \text{r5} \neq \text{r10} \\ 5 & \text{if } (\text{r1} = \text{r2} \wedge \text{r2} \neq \text{r3}, \text{r4}, \text{r5}, \text{r10} \wedge \text{r3} \neq \text{r4}, \text{r5}, \text{r10} \wedge \text{r4} \neq \text{r5}, \text{r10} \wedge \text{r5} \neq \text{r10}) \vee \ldots \\ \vdots & \\ 1 & \text{if } \text{r1} = \text{r2} = \text{r3} = \text{r4} = \text{r5} = \text{r10} \end{cases}$$

Another Motivating Example (Klebanov 2015)

$$\text{card}\,\{\,[\text{r1}];[\text{r2}];[\text{r3}];[\text{r4}];[\text{r5}];[\text{r10}]\,\}$$

$$\begin{cases} 6 & \text{if } \text{r1} \neq \text{r2}, \text{r3}, \text{r4}, \text{r5}, \text{r10} \wedge \text{r2} \neq \text{r3}, \text{r4}, \text{r5}, \text{r10} \wedge \text{r3} \neq \text{r4}, \text{r5}, \text{r10} \wedge \text{r4} \neq \text{r5}, \text{r10} \wedge \text{r5} \neq \text{r10} \\ 5 & \text{if } (\text{r1} = \text{r2} \wedge \text{r2} \neq \text{r3}, \text{r4}, \text{r5}, \text{r10} \wedge \text{r3} \neq \text{r4}, \text{r5}, \text{r10} \wedge \text{r4} \neq \text{r5}, \text{r10} \wedge \text{r5} \neq \text{r10}) \vee \ldots \\ \vdots & \\ 1 & \text{if } \text{r1} = \text{r2} = \text{r3} = \text{r4} = \text{r5} = \text{r10} \end{cases}$$

$\Rightarrow$ large representation even with explicit disequality constraints

$\Rightarrow$ a lot worse without

Core Representation of Polyhedral Compilation Library

Conjunction of affine inequality constraints

$$\{\, \mathbf{z} : A\mathbf{z} + \mathbf{a} \geq \mathbf{0} \qquad\qquad \}$$

$+$ unions of such sets

No explicit representation for disequality constraints

This applies to libraries

- not supporting existentially quantified variables:
  - `PolyLib` (Wilde 1993)
  - `PPL` (Bagnara et al. 2008)
- supporting existentially quantified variables:
  - `Omega` (Kelly et al. Nov. 1996)
  - `isl` (V. 2010)
  - `Omega+` (Chen June 2012)
  - `FPL` (Pitchanathan et al. Oct. 2021)

Core Representation of Polyhedral Compilation Library

Conjunction of affine inequality constraints

$$\{\, \mathbf{z} : A\mathbf{z} + \mathbf{a} \geq \mathbf{0} \qquad\qquad \}$$

$+$ unions of such sets

No explicit representation for disequality constraints

How about equality constraints?

Core Representation of Polyhedral Compilation Library

Conjunction of affine inequality constraints

$$\{\, \mathbf{z} : A\mathbf{z} + \mathbf{a} \geq \mathbf{0} \wedge B\mathbf{z} + \mathbf{b} = \mathbf{0} \,\}$$

$+$ unions of such sets

No explicit representation for disequality constraints

How about equality constraints?

Core Representation of Polyhedral Compilation Library

Conjunction of affine inequality constraints

$$\{\, \mathbf{z} : A\mathbf{z} + \mathbf{a} \geq \mathbf{0} \wedge B\mathbf{z} + \mathbf{b} = \mathbf{0} \,\}$$

$+$ unions of such sets

No explicit representation for disequality constraints

How about equality constraints? Not strictly needed but still used

- do not change expressivity
- n equality constraints replace $n + 1$ to $2n$ inequality constraints
- every (independent) equality constraint reduces effective dimensionality

Core Representation of Polyhedral Compilation Library

Conjunction of affine inequality constraints

$$\{\, \mathbf{z} : A\mathbf{z} + \mathbf{a} \geq \mathbf{0} \wedge B\mathbf{z} + \mathbf{b} = \mathbf{0} \,\}$$

$+$ unions of such sets

No explicit representation for disequality constraints

How about equality constraints? Not strictly needed but still used

- do not change expressivity
- n equality constraints replace $n+1$ to $2n$ inequality constraints
- every (independent) equality constraint reduces effective dimensionality

Why not disequality constraints?

- do not change expressivity
- n disequality constraints avoid split into 2 to 2^n disjuncts

Outline

Seater and Wonnacott (2005)

Detect “inert” disequality constraints

$\Rightarrow$ disequality constraints that can be ignored (in terms of emptiness)

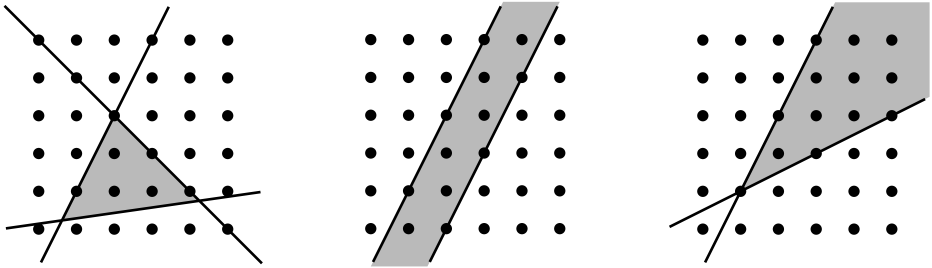
$\Rightarrow$ disequality constraints that involve unbounded direction

Seater and Wonnacott (2005)

Detect "inert" disequality constraints

⇒ disequality constraints that can be ignored (in terms of emptiness)

⇒ disequality constraints that involve unbounded direction

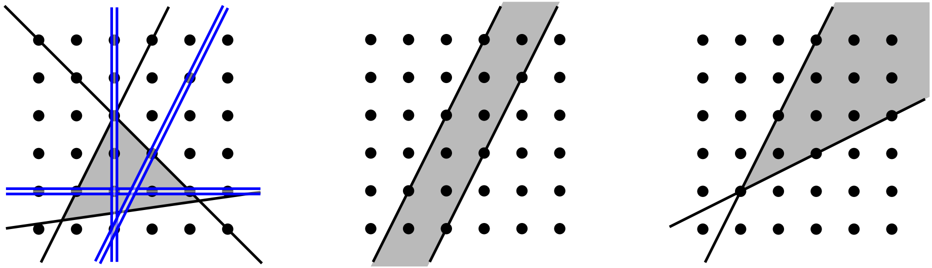


Seater and Wonnacott (2005)

Detect “inert” disequality constraints

⇒ disequality constraints that can be ignored (in terms of emptiness)

⇒ disequality constraints that involve unbounded direction



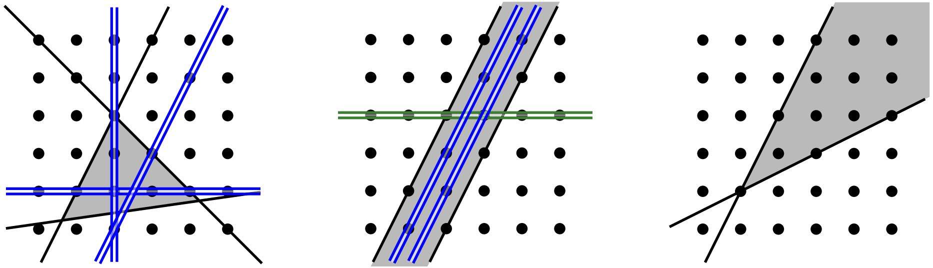
None inert

Seater and Wonnacott (2005)

Detect "inert" disequality constraints

⇒ disequality constraints that can be ignored (in terms of emptiness)

⇒ disequality constraints that involve unbounded direction



None inert

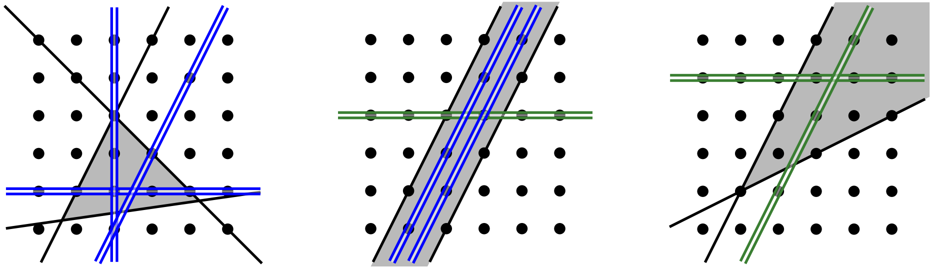
Some inert

Seater and Wonnacott (2005)

Detect "inert" disequality constraints

⇒ disequality constraints that can be ignored (in terms of emptiness)

⇒ disequality constraints that involve unbounded direction



None inert Some inert All inert

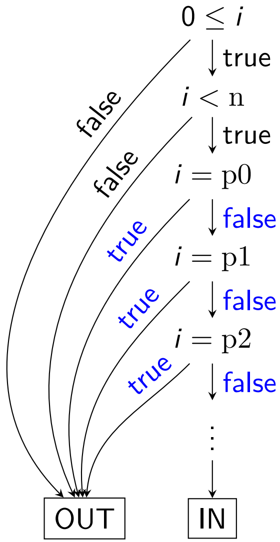
Seater and Wonnacott (2005)

An equivalent approach would be to simply allow negated equality constraints in simplified relations. This approach could be taken even further, to allow more general negated constraints, or other formulas that cannot be handled efficiently

We do not currently have an implementation of our algorithms, and thus we do not have empirical verification that they are either fast or effective in practice. Given the nature of the changes discussed in the previous section, we do not expect to have an implementation any time soon.

Kulkarni and Kruse (June 2022)

$\{ \mathrm{Stmt}[i] : 0 \leq i < \mathrm{n} \wedge i \neq \mathrm{p0} \wedge i \neq \mathrm{p1} \wedge i \neq \mathrm{p2} \wedge \ldots \}$



Polyhedral binary decision diagram, PBDD

- internal nodes: affine (in)equality constraints
- terminal nodes: IN: in set; OUT: not in set

⇒ allows negation of (conjunction of) affine constraints (disequality constraint is special case)

However

- limited number of supported operations (intersection, union, subtraction, complement)
- revert to `isl` (with expansion) for other operations
- no support for existentially quantified variables

Outline

Explicit Disequality Constraints

Main changes:

- extend internal representation
- resolve hidden assumptions
- adjust some core algorithms

Explicit Disequality Constraints

Main changes:

- extend internal representation
- resolve hidden assumptions
- adjust some core algorithms

Changes are (mostly) transparent to user of `isl`

- results of some heuristics-based operations may change

- new expression type in result of AST generation

Explicit Disequality Constraints

Main changes:

- extend internal representation
- resolve hidden assumptions
- adjust some core algorithms

Changes are (mostly) transparent to user of `isl`

- results of some heuristics-based operations may change
  For example: `pet` test case

  ```
  { S_5[i=0:99] -> T[i] : i != 57 } % { S_5[i=0:99] : i != 57 };
  ```

  ⇒ `{ S_5[i] -> T[i] : i >= 58 or i <= 56 };` now: `{ S_5[i] -> T[i] }`
- new expression type in result of AST generation

Explicit Disequality Constraints

Main changes:

- extend internal representation
- resolve hidden assumptions
- adjust some core algorithms

Changes are (mostly) transparent to user of `isl`

- results of some heuristics-based operations may change
  For example: pet test case

  ```
  { S_5[i=0:99] -> T[i] : i != 57 } % { S_5[i=0:99] : i != 57 };
  ```

  ⇒ `{ S_5[i] -> T[i] : i >= 58 or i <= 56 };` now: `{ S_5[i] -> T[i] }`
- new expression type in result of AST generation

  ```
  for (int c0 = 1; c0 <= 9; c0 += 1) {
    if (c0 != 5) {
      for (int c1 = 1; c1 <= 9; c1 += 1)
        s0(c0, c1);
    } else {
  ```

Extend Internal Representation

Basic set:

$$\{\, \mathbf{z} : A\mathbf{z} + \mathbf{a} \geq \mathbf{0} \wedge B\mathbf{z} + \mathbf{b} = \mathbf{0} \qquad\qquad \}$$

$+$ unions of basic sets

Extend Internal Representation

Basic set:

$$\{\, \mathbf{z} : A\mathbf{z} + \mathbf{a} \geq \mathbf{0} \wedge B\mathbf{z} + \mathbf{b} = \mathbf{0} \wedge C\mathbf{z} + \mathbf{c} \neq \mathbf{0} \,\}$$

$+$ unions of basic sets

Extend Internal Representation

Basic set:

$$\{\, \mathbf{z} : A\mathbf{z} + \mathbf{a} \geq \mathbf{0} \wedge B\mathbf{z} + \mathbf{b} = \mathbf{0} \wedge C\mathbf{z} + \mathbf{c} \neq \mathbf{0} \,\}$$

$+$ unions of basic sets

Simplifications:

- $m\, f(\mathbf{z}) + c \neq 0$

  $\Rightarrow$ drop constraint if m does not divide c

Extend Internal Representation

Basic set:

$$\{ \mathbf{z} : A\mathbf{z} + \mathbf{a} \geq \mathbf{0} \wedge B\mathbf{z} + \mathbf{b} = \mathbf{0} \wedge C\mathbf{z} + \mathbf{c} \neq \mathbf{0} \}$$

$+$ unions of basic sets

Simplifications:

- $m\, f(\mathbf{z}) + c \neq 0$
  - $\Rightarrow$ drop constraint if m does not divide c
- $c \neq 0$
  - $\Rightarrow$ drop constraint if c is not zero
  - $\Rightarrow$ mark basic set empty if c is zero

Extend Internal Representation

Basic set:

$$\{ \mathbf{z} : A\mathbf{z} + \mathbf{a} \geq \mathbf{0} \wedge B\mathbf{z} + \mathbf{b} = \mathbf{0} \wedge C\mathbf{z} + \mathbf{c} \neq \mathbf{0} \}$$

$+$ unions of basic sets

Simplifications:

- $m\, f(\mathbf{z}) + c \neq 0$
  - $\Rightarrow$ drop constraint if m does not divide c
- $c \neq 0$
  - $\Rightarrow$ drop constraint if c is not zero
  - $\Rightarrow$ mark basic set empty if c is zero
- only exact duplicates (or opposites) of disequality constraints can be removed

Extend Internal Representation

Basic set:

$$\{\, \mathbf{z} : A\mathbf{z} + \mathbf{a} \geq \mathbf{0} \wedge B\mathbf{z} + \mathbf{b} = \mathbf{0} \wedge C\mathbf{z} + \mathbf{c} \neq \mathbf{0} \,\}$$

$+$ unions of basic sets

Simplifications:

- $m\, f(\mathbf{z}) + c \neq 0$
  - $\Rightarrow$ drop constraint if m does not divide c
- $c \neq 0$
  - $\Rightarrow$ drop constraint if c is not zero
  - $\Rightarrow$ mark basic set empty if c is zero
- only exact duplicates (or opposites) of disequality constraints can be removed
- $f(\mathbf{z}) + c \neq 0$
  $f(\mathbf{z}) + a \geq 0$
  - $\Rightarrow$ replace by $f(\mathbf{z}) + a - 1 \geq 0$ if $a = c$
  - $\Rightarrow$ drop disequality if $a < c$

Resolve Hidden Assumptions

Main hidden assumption in `isl`: basic set is convex

Implications:

- all integer values between min/max rational values are in basic set
- simple hull operation can convert 1-disjunct set into basic set
  - $\Rightarrow$ introduce special operation for conversion
  - $\Rightarrow$ simple hull operation drops disequality constraints
  - $\Rightarrow$ another operation for shared constraints needed?

Disequality Constraints in Tableau

Introduce a non-zero variable for each disequality constraint

- a non-zero variable does not participate in pivoting
  - ⇒ always a row variable
- if non-zero variable can attain only negative or only positive values
  - ⇒ non-zero variable is redundant and can be removed
- if non-zero variable can obviously only attain zero value
  - zero values for all remaining columns
  - ⇒ tableau is empty
- sample point only valid if all non-zero variables have non-zero value

$$f_1 = x + y \geq 0$$
$$f_2 = x - 10 \geq 0$$
$$f_3 = y - 5 \geq 0$$
$$f_4 = -y + 5 \geq 0$$
$$f_5 = y - 5 \neq 0$$

		x	y
f_1	0	1	1
f_2	-10	1	0
f_3	-5	0	1
f_4	5	0	-1
f_5	-5	0	1

Emptiness and Sampling

- Sampling picks an integer element
- Set is empty if it has no integer elements

Emptiness and Sampling

- Sampling picks an integer element
- Set is empty if it has no integer elements

Procedure

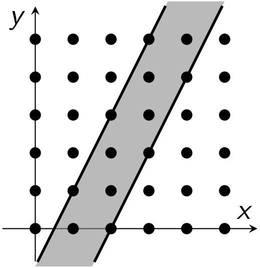
- trivial solution for 0D and 1D sets
- isolate bounded directions
  - compute recession cone (replace constant terms by 0)
  - (implicit) equality constraints determine bounded directions
  - perform unimodular transformation
- perform backtracking search in tableau on bounded dimensions (can fail)
- pick some corresponding value for unbounded dimensions (always succeeds)

Emptiness and Sampling

- Sampling picks an integer element
- Set is empty if it has no integer elements

Procedure

- trivial solution for 0D and 1D sets
- isolate bounded directions
  - compute recession cone (replace constant terms by 0)
  - (implicit) equality constraints determine bounded directions
  - perform unimodular transformation



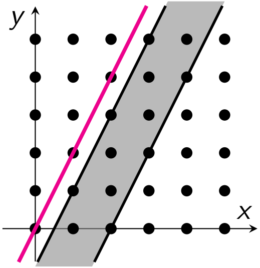
$$\{ [x, y] : 1 \leq 2x - y \leq 4 \}$$

Emptiness and Sampling

- Sampling picks an integer element
- Set is empty if it has no integer elements

Procedure

- trivial solution for 0D and 1D sets
- isolate bounded directions
  - compute recession cone (replace constant terms by 0)
  - (implicit) equality constraints determine bounded directions
  - perform unimodular transformation



$$\{ [x, y] : 1 \leq 2x - y \leq 4 \}$$

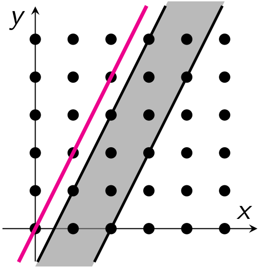
$$\{ [x, y] : 0 \leq 2x - y \leq 0 \}$$

Emptiness and Sampling

- Sampling picks an integer element
- Set is empty if it has no integer elements

Procedure

- trivial solution for 0D and 1D sets
- isolate bounded directions
  - compute recession cone (replace constant terms by 0)
  - (implicit) equality constraints determine bounded directions
  - perform unimodular transformation



$$\{ [x, y] : 1 \le 2x - y \le 4 \}$$

$$\{ [x, y] : 0 \le 2x - y \le 0 \}$$

$$x' = 2x - y$$

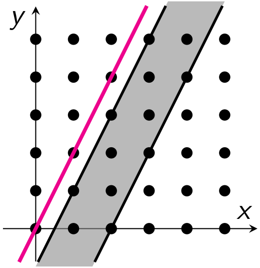
$$y' = y$$

Emptiness and Sampling

- Sampling picks an integer element
- Set is empty if it has no integer elements

Procedure

- trivial solution for 0D and 1D sets
- isolate bounded directions
  - compute recession cone (replace constant terms by 0)
  - (implicit) equality constraints determine bounded directions
  - perform unimodular transformation

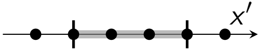


$$\{ [x, y] : 1 \le 2x - y \le 4 \}$$

$$\{ [x, y] : 0 \le 2x - y \le 0 \}$$

$$x' = 2x - y$$

$$y' = y$$



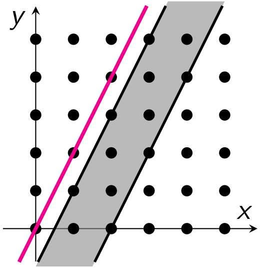
$$\{ [x'] : 1 \le x' \le 4 \}$$

Emptiness and Sampling

- Sampling picks an integer element
- Set is empty if it has no integer elements

Procedure

- trivial solution for 0D and 1D sets
- isolate bounded directions
  - compute recession cone (replace constant terms by 0)
  - (implicit) equality constraints determine bounded directions
  - perform unimodular transformation

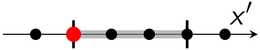


$$\{\, [x, y] : 1 \le 2x - y \le 4 \,\}$$

$$\{\, [x, y] : 0 \le 2x - y \le 0 \,\}$$

$$x' = 2x - y$$

$$y' = y$$



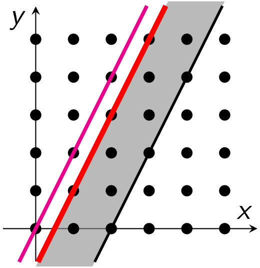
$$\{\, [x'] : 1 \le x' \le 4 \,\}$$

Emptiness and Sampling

- Sampling picks an integer element
- Set is empty if it has no integer elements

Procedure

- trivial solution for 0D and 1D sets
- isolate bounded directions
  - compute recession cone (replace constant terms by 0)
  - (implicit) equality constraints determine bounded directions
  - perform unimodular transformation

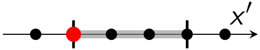


$\{ [x, y] : 1 \leq 2x - y \leq 4 \}$

$\{ [x, y] : 0 \leq 2x - y \leq 0 \}$

$x' = 2x - y$

$y' = y$



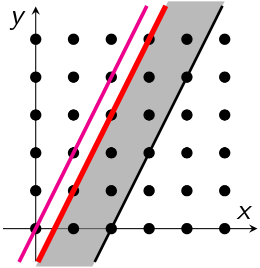
$\{ [x'] : 1 \leq x' \leq 4 \}$

Emptiness and Sampling

- Sampling picks an integer element
- Set is empty if it has no integer elements

Procedure

- trivial solution for 0D and 1D sets
- isolate bounded directions
  - compute recession cone (replace constant terms by 0)
  - (implicit) equality constraints determine bounded directions
  - perform unimodular transformation

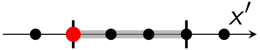


$$\{\, [x, y] : 1 \leq 2x - y \leq 4 \,\}$$

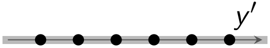
$$\{\, [x, y] : 0 \leq 2x - y \leq 0 \,\}$$

$$x' = 2x - y$$

$$y' = y$$



$$\{\, [x'] : 1 \leq x' \leq 4 \,\}$$



Picking Element in Unbounded Set

Rational element can easily be picked in tableau (sample value, possibly non-integer values)

$\Rightarrow$ restrict set to points that have entire unit cube included in original set

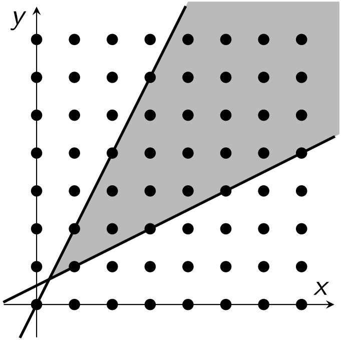
$\Rightarrow$ pick rational element in restricted set

$\Rightarrow$ round up

Picking Element in Unbounded Set

Rational element can easily be picked in tableau (sample value, possibly non-integer values)

- $\Rightarrow$ restrict set to points that have entire unit cube included in original set
- $\Rightarrow$ pick rational element in restricted set
- $\Rightarrow$ round up

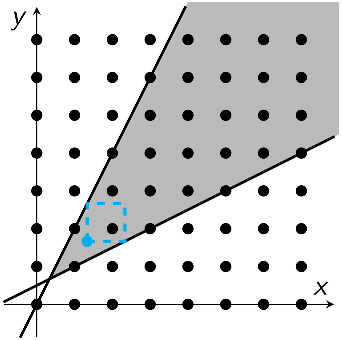


$$\{\,[x, y] : y \leq 2x \wedge x \leq 2y - 1\,\}$$

Picking Element in Unbounded Set

Rational element can easily be picked in tableau (sample value, possibly non-integer values)

- $\Rightarrow$ restrict set to points that have entire unit cube included in original set
- $\Rightarrow$ pick rational element in restricted set
- $\Rightarrow$ round up

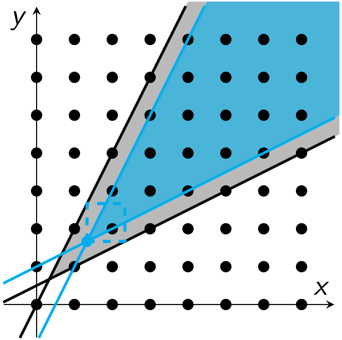


$$\{\, [x, y] : y \leq 2x \wedge x \leq 2y - 1 \,\}$$

Picking Element in Unbounded Set

Rational element can easily be picked in tableau (sample value, possibly non-integer values)

- $\Rightarrow$ restrict set to points that have entire unit cube included in original set
- $\Rightarrow$ pick rational element in restricted set
- $\Rightarrow$ round up



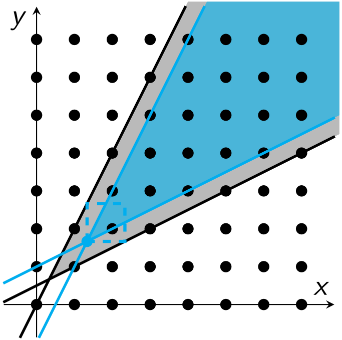
$$\{ [x, y] : y \leq 2x \wedge x \leq 2y - 1 \}$$

$$\{ [x, y] : y \leq 2x - 1 \wedge x \leq 2y - 2 \}$$

Picking Element in Unbounded Set

Rational element can easily be picked in tableau (sample value, possibly non-integer values)

- $\Rightarrow$ restrict set to points that have entire unit cube included in original set
- $\Rightarrow$ pick rational element in restricted set $(4/3, 5/3)$
- $\Rightarrow$ round up



$$\{\, [x, y] : y \leq 2x \wedge x \leq 2y - 1 \,\}$$

$$\{\, [x, y] : y \leq 2x - 1 \wedge x \leq 2y - 2 \,\}$$

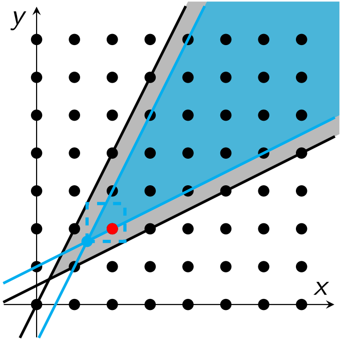
Picking Element in Unbounded Set

Rational element can easily be picked in tableau (sample value, possibly non-integer values)

$\Rightarrow$ restrict set to points that have entire unit cube included in original set

$\Rightarrow$ pick rational element in restricted set $(4/3, 5/3)$

$\Rightarrow$ round up $(2, 2)$



$$\{\, [x, y] : y \leq 2x \wedge x \leq 2y - 1 \,\}$$

$$\{\, [x, y] : y \leq 2x - 1 \wedge x \leq 2y - 2 \,\}$$

Emptiness and Sampling

- Sampling picks an integer element
- Set is empty if it has no integer elements

Procedure

- trivial solution for 0D and 1D sets
- isolate bounded directions
  - compute recession cone (replace constant terms by 0)
  - (implicit) equality constraints determine bounded directions
  - perform unimodular transformation
- perform backtracking search in tableau on bounded dimensions (can fail)
- pick some corresponding value for unbounded dimensions (always succeeds)

Emptiness and Sampling

- Sampling picks an integer element
- Set is empty if it has no integer elements

Procedure

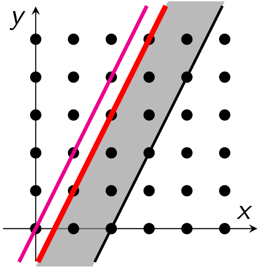
- trivial solution for 0D and 1D sets
- isolate bounded directions
  - compute recession cone (replace constant terms by 0) ignoring disequality constraints
  - (implicit) equality constraints determine bounded directions
  - perform unimodular transformation
- perform backtracking search in tableau on bounded dimensions (can fail)
  - drop disequality constraints involving unbounded dimension ("inert")
  - skip values violating any other disequality constraint
- pick some corresponding value for unbounded dimensions (always succeeds)

Emptiness and Sampling

- Sampling picks an integer element
- Set is empty if it has no integer elements

Procedure

- trivial solution for 0D and 1D sets
- isolate bounded directions
  - compute recession cone (replace constant terms by 0) ignoring disequality constraints
  - (implicit) equality constraints determine bounded directions
  - perform unimodular transformation

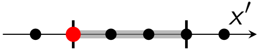


$$\{\, [x, y] : 1 \le 2x - y \le 4 \,\}$$

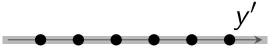
$$\{\, [x, y] : 0 \le 2x - y \le 0 \,\}$$

$$x' = 2x - y$$

$$y' = y$$



$$\{\, [x'] : 1 \le x' \le 4 \,\}$$

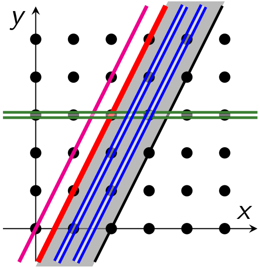


Emptiness and Sampling

- Sampling picks an integer element
- Set is empty if it has no integer elements

Procedure

- trivial solution for 0D and 1D sets
- isolate bounded directions
  - compute recession cone (replace constant terms by 0) ignoring disequality constraints
  - (implicit) equality constraints determine bounded directions
  - perform unimodular transformation

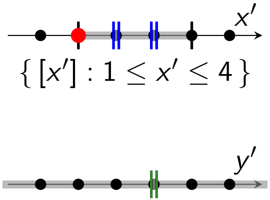


$\{ [x, y] : 1 \leq 2x - y \leq 4 \}$

$\{ [x, y] : 0 \leq 2x - y \leq 0 \}$

$x' = 2x - y$

$y' = y$



$\{ [x'] : 1 \leq x' \leq 4 \}$

Picking Element in Unbounded Set

Rational element can easily be picked in tableau (sample value, possibly non-integer values)

$\Rightarrow$ restrict set to points that have entire unit cube included in original set

$\Rightarrow$ pick rational element in restricted set

$\Rightarrow$ round up

Picking Element in Unbounded Set

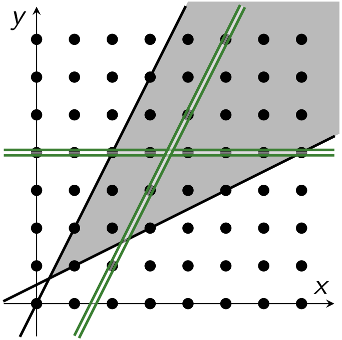
Rational element can easily be picked in tableau (sample value, possibly non-integer values)

- ⇒ restrict set to points that have entire $(1 + n^{\neq})$-cube included in original set
- ⇒ pick rational element in restricted set
- ⇒ round up (skipping violated disequality constraints)

Picking Element in Unbounded Set

Rational element can easily be picked in tableau (sample value, possibly non-integer values)

- ⇒ restrict set to points that have entire $(1 + n^{\neq})$-cube included in original set
- ⇒ pick rational element in restricted set
- ⇒ round up (skipping violated disequality constraints)

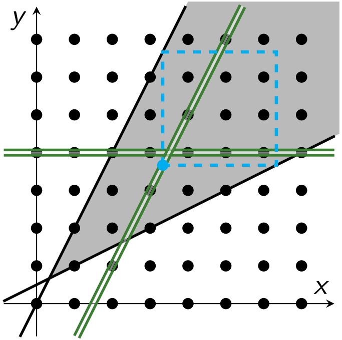


$$\{ [x, y] : y \leq 2x \land x \leq 2y - 1 \}$$

Picking Element in Unbounded Set

Rational element can easily be picked in tableau (sample value, possibly non-integer values)

- $\Rightarrow$ restrict set to points that have entire $(1 + n^{\neq})$-cube included in original set
- $\Rightarrow$ pick rational element in restricted set
- $\Rightarrow$ round up (skipping violated disequality constraints)

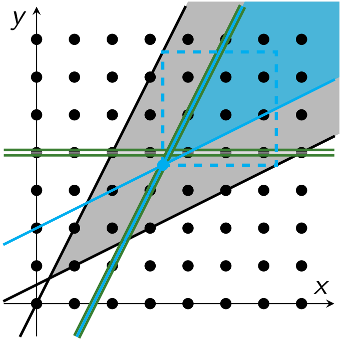


$$\{\,[x, y] : y \leq 2x \wedge x \leq 2y - 1\,\}$$

Picking Element in Unbounded Set

Rational element can easily be picked in tableau (sample value, possibly non-integer values)

- $\Rightarrow$ restrict set to points that have entire $(1 + n^{\neq})$-cube included in original set
- $\Rightarrow$ pick rational element in restricted set
- $\Rightarrow$ round up (skipping violated disequality constraints)



$$\{\, [x, y] : y \leq 2x \wedge x \leq 2y - 1 \,\}$$

$$\{\, [x, y] : y \leq 2x - 3 \wedge x \leq 2y - 4 \,\}$$

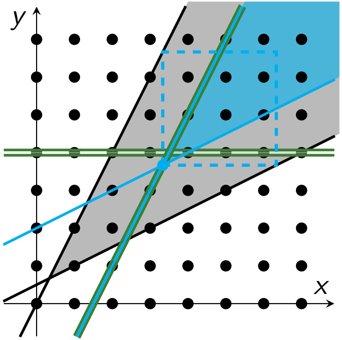
Picking Element in Unbounded Set

Rational element can easily be picked in tableau (sample value, possibly non-integer values)

$\Rightarrow$ restrict set to points that have entire $(1+n^{\neq})$-cube included in original set

$\Rightarrow$ pick rational element in restricted set $(10/3, 11/3)$

$\Rightarrow$ round up (skipping violated disequality constraints)



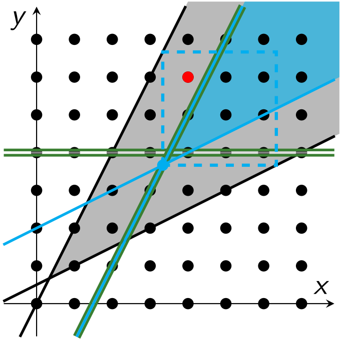
$$\{\,[x,y] : y \le 2x \wedge x \le 2y - 1\,\}$$

$$\{\,[x,y] : y \le 2x - 3 \wedge x \le 2y - 4\,\}$$

Picking Element in Unbounded Set

Rational element can easily be picked in tableau (sample value, possibly non-integer values)

- $\Rightarrow$ restrict set to points that have entire $(1 + n^{\neq})$-cube included in original set
- $\Rightarrow$ pick rational element in restricted set $(10/3, 11/3)$
- $\Rightarrow$ round up (skipping violated disequality constraints) $(4, 6)$



$$\{ [x, y] : y \leq 2x \wedge x \leq 2y - 1 \}$$

$$\{ [x, y] : y \leq 2x - 3 \wedge x \leq 2y - 4 \}$$

Redundant Local Variables

Basic set:

$$\{\, \mathbf{x} : \exists \boldsymbol{\alpha} : A_1 \mathbf{x} + A_2 \boldsymbol{\alpha} + \mathbf{a} \geq \mathbf{0} \,\}$$

Some local variable $\boldsymbol{\alpha}$ may be redundant

Some of these can be detected based purely on constraints

- consider all pairs of lower and upper bounds on variable α
- if each pair admits an integer value
  $\Rightarrow$ α can be eliminated (using Fourier-Motzkin)

Redundant Local Variables

Basic set:

$$\{\, \mathbf{x} : \exists \boldsymbol{\alpha} : A_1\mathbf{x} + A_2\boldsymbol{\alpha} + \mathbf{a} \geq \mathbf{0} \,\}$$

Some local variable $\boldsymbol{\alpha}$ may be redundant

Some of these can be detected based purely on constraints

- consider all pairs of lower and upper bounds on variable α
- α is involved in n disequality constraints
- if each pair admits $1 + n$ integer values

  $\Rightarrow$ α can be eliminated (using Fourier-Motzkin)

Redundant Local Variables

Basic set:

$$\{\, \mathbf{x} : \exists \boldsymbol{\alpha} : A_1\mathbf{x} + A_2\boldsymbol{\alpha} + \mathbf{a} \geq \mathbf{0} \,\}$$

Some local variable $\boldsymbol{\alpha}$ may be redundant

Some of these can be detected based purely on constraints

- consider all pairs of lower and upper bounds on variable α
- α is involved in n disequality constraints
- if each pair admits $1 + n$ integer values
  - $\Rightarrow$ α can be eliminated (using Fourier-Motzkin)

$\Rightarrow$ potential trade-off between number of disjuncts and dimensionality of disjuncts

Parametric Integer Programming

Compute lexicographic minimum of some variables **x** in terms of other variables **n**

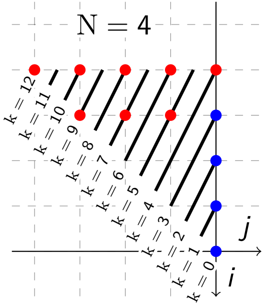
Two tableaux:

- main tableau in **x** and **n**
- context tableau **n**

Pivoting in main tableau depends on sign of symbolic constant term in context tableau

$\Rightarrow$ requires context splits if constant term can attain both positive and negative values

$$R = \{ [i,j] : 0 \le -i \le \mathrm{N} \wedge 0 \le -j \le -i \wedge 0 \le \mathrm{k} \le 3\mathrm{N} \wedge \mathrm{k} = -i - 2j \}$$



lexmin R =

if $\mathrm{k} < \mathrm{N}$

$[-\mathrm{k}, 0]$

else

if $3\left\lfloor \frac{\mathrm{k}+\mathrm{N}}{2} \right\rfloor \ge 2\mathrm{k}$

$\left[\mathrm{k} - 2\left\lfloor \frac{\mathrm{k}+\mathrm{N}}{2} \right\rfloor, -\mathrm{k} + \left\lfloor \frac{\mathrm{k}+\mathrm{N}}{2} \right\rfloor\right]$

Parametric Integer Programming

Compute lexicographic minimum of some variables $\mathbf{x}$ in terms of other variables $\mathbf{n}$

Two tableaux:

- main tableau in $\mathbf{x}$ and $\mathbf{n}$
- context tableau $\mathbf{n}$

Pivoting in main tableau depends on sign of symbolic constant term in context tableau

$\Rightarrow$ requires context splits if constant term can attain both positive and negative values

Keep track of disequality constraints in tableaux

If disequality constraint $g(\mathbf{n}, \mathbf{x}) \neq 0$ may be violated by potential solution

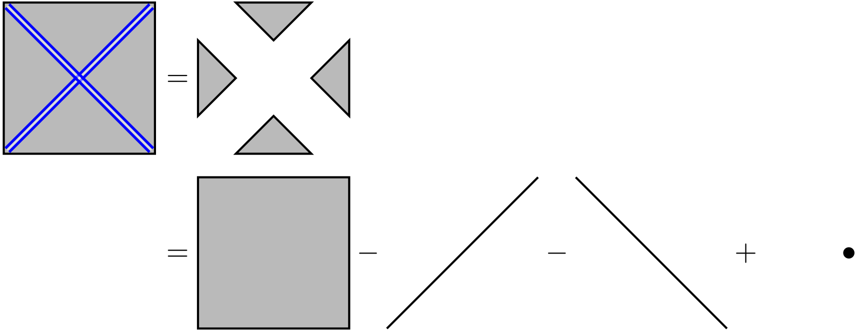
$\Rightarrow$ split context into 2 cases

- $f(\mathbf{n}) \neq 0$ (implying $g(\mathbf{n}, \mathbf{x}) \neq 0$ is not violated)
  $\Rightarrow$ proceed with other disequality constraints
- $f(\mathbf{n}) = 0$ (implying $g(\mathbf{n}, \mathbf{x}) \neq 0$ is violated)
  - $\Rightarrow$ compute two solutions, for $g(\mathbf{n}, \mathbf{x}) \geq 1$ and $g(\mathbf{n}, \mathbf{x}) \leq -1$
  - $\Rightarrow$ take minimum of two solutions

Splitting $g(\mathbf{n}, \mathbf{x}) \neq 0$ up front computes same minimum but then cost is always incurred

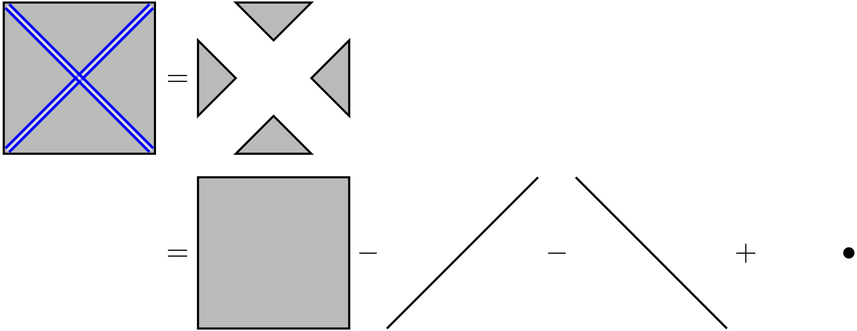
Some Other Operations

- Preparation for counting using `barvinok` (V., Seghir, et al. June 2007)



Some Other Operations

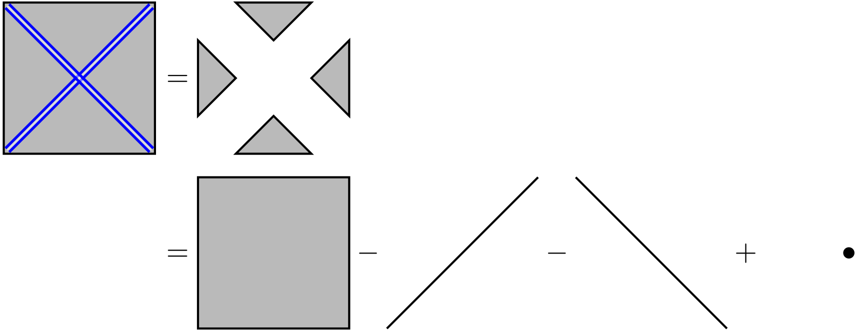
- Preparation for counting using `barvinok`



- Transitive closure approximation
  Basic sets do not have to be split but result may be less accurate

Some Other Operations

- Preparation for counting using `barvinok`

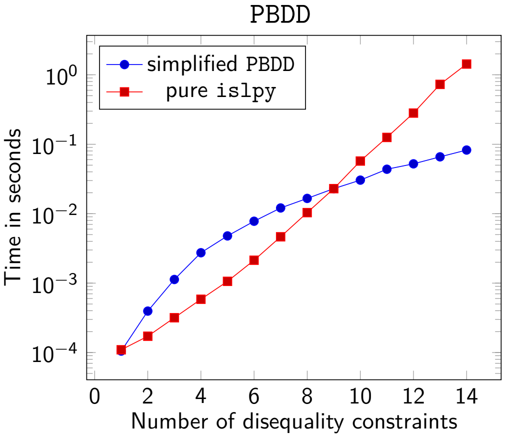


- Transitive closure approximation
  Basic sets do not have to be split but result may be less accurate
- Scheduling
  Disequality constraints essentially ignored

Outline

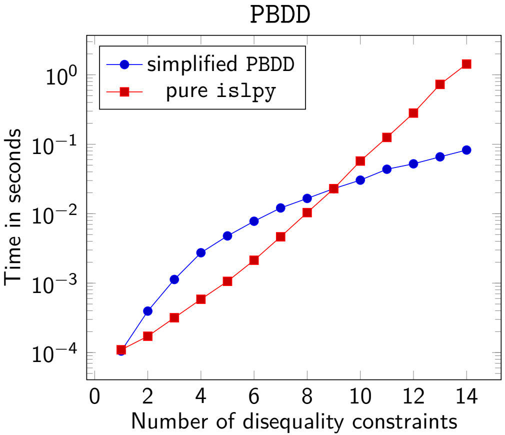
PBDD versus isl with Explicit Disequality Constraints

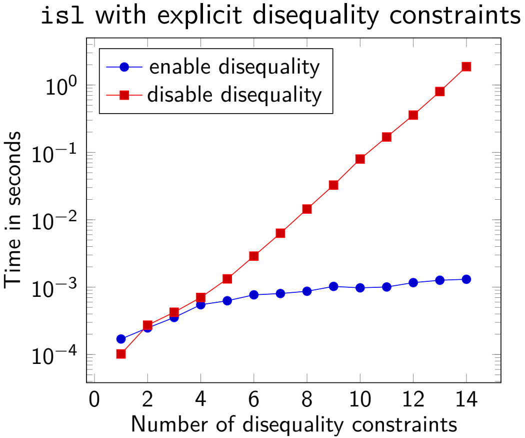
$$\{\, [i] : \bigwedge_{j \leq n^{\neq}} i \neq \mathrm{p}_j \,\}$$



PBDD versus isl with Explicit Disequality Constraints

$$\{\, [i] : \bigwedge_{j \leq n^{\neq}} i \neq \mathrm{p}_j \,\}$$





Note: construction times with PBDD and isl not directly comparable

Full Polyhedral Compilation Flow

```
for (int i = 0; i < n; ++i) {
  if (i == p0 || i == p1 || i == p2)
    continue;
  A[i] = i;
}
for (int i = 0; i < n; ++i) {
  if (i == p0 || i == p1 || i == p2)
    continue;
  B[i] = A[i];
}
```

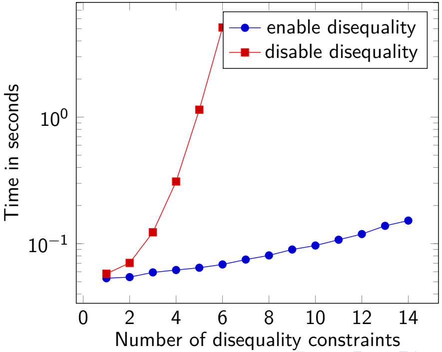
PPCG (V., Juega, et al. 2013) output:

```
for (int c0 = 0; c0 < n; c0 += 1)
  if (c0 != p0 && c0 != p1 && c0 != p2) {
    A[c0] = (c0);
    B[c0] = A[c0];
  }
```

(No changes required to PPCG)

Involves

- construction of polyhedral model
- dependence analysis
- scheduling
- AST generation



Outline

Conclusion

Supporting explicit disequality constraints in a polyhedral compilation library is feasible

- requires only conceptually minor adjustments
- in some cases simply delaying split to where it becomes relevant
- can dramatically reduce size of representation
- mostly transparent to the user

Some trade-offs involved, e.g.,

- elimination of redundant local variables
- accuracy of transitive closure approximation

Perhaps useful to consider other explicit constraints, e.g.,

- lexicographic constraints

Outline

Incremental LP Solver

Core representation: tableau

Given

$$\{\, \mathbf{z} : A\mathbf{z} + \mathbf{a} \geq \mathbf{0} \,\}$$

with n variables $\mathbf{z}$ and m constraints $A\mathbf{z} + \mathbf{a} \geq \mathbf{0}$

- introduce a non-negative variable f_i for each affine expression
- tableau writes m variables in terms of n variables
- initially, $\mathbf{f}$ in terms of $\mathbf{z}$

$$f_1 = x + y \geq 0$$
$$f_2 = x - 10 \geq 0$$
$$f_3 = y - 5 \geq 0$$
$$f_4 = -y + 5 \geq 0$$

		x	y
f_1	0	1	1
f_2	-10	1	0
f_3	-5	0	1
f_4	5	0	-1

Incremental LP Solver

Core representation: tableau

Given

$$\{\, \mathbf{z} : A\mathbf{z} + \mathbf{a} \geq \mathbf{0} \,\}$$

with n variables $\mathbf{z}$ and m constraints $A\mathbf{z} + \mathbf{a} \geq \mathbf{0}$

- introduce a non-negative variable f_i for each affine expression
- tableau writes m variables in terms of n variables
- initially, $\mathbf{f}$ in terms of $\mathbf{z}$

$$f_1 = x + y \geq 0$$
$$f_2 = x - 10 \geq 0$$
$$f_3 = y - 5 \geq 0$$
$$f_4 = -y + 5 \geq 0$$

		x	y
f_1	0	1	1
f_2	-10	1	0
f_3	-5	0	1
f_4	5	0	-1

- sample value: assign zero to all column variables
  $x = 0, y = 0, f_1 = 0, f_2 = -10, f_3 = -5, f_4 = 5$

Incremental LP Solver

$$f_1 = x + y \geq 0$$
$$f_2 = x - 10 \geq 0$$
$$f_3 = y - 5 \geq 0$$
$$f_4 = -y + 5 \geq 0$$

		x	y
f_1	0	1	1
f_2	−10	1	0
f_3	−5	0	1
f_4	5	0	−1

- a pivot interchanges a row and a column variable

Incremental LP Solver

$f_1 = x + y \geq 0$

$f_2 = x - 10 \geq 0$

$f_3 = y - 5 \geq 0$

$f_4 = -y + 5 \geq 0$

		x	y
f_1	0	1	1
f_2	−10	1	0
f_3	−5	0	1
f_4	5	0	−1

- a pivot interchanges a row and a column variable

Incremental LP Solver

$$f_1 = x + y \geq 0$$
$$f_2 = x - 10 \geq 0$$
$$f_3 = y - 5 \geq 0$$
$$f_4 = -y + 5 \geq 0$$

		x	y
f_1	0	1	1
f_2	-10	1	0
f_3	-5	0	1
f_4	5	0	-1

- a pivot interchanges a row and a column variable
  $f_3 = -5 + y \Rightarrow y = 5 + f_3,\ f_1 = 5 + x + f_3,\ f_4 = -f_3$

Incremental LP Solver

$f_1 = x + y \geq 0$

$f_2 = x - 10 \geq 0$

$f_3 = y - 5 \geq 0$

$f_4 = -y + 5 \geq 0$

		x	y
f_1	0	1	1
f_2	−10	1	0
f_3	−5	0	1
f_4	5	0	−1

		x	f_3
f_1	5	1	1
f_2	−10	1	0
y	5	0	1
f_4	0	0	−1

- a pivot interchanges a row and a column variable
  $f_3 = -5 + y \Rightarrow y = 5 + f_3,\ f_1 = 5 + x + f_3,\ f_4 = -f_3$

Incremental LP Solver

$f_1 = x + y \geq 0$

$f_2 = x - 10 \geq 0$

$f_3 = y - 5 \geq 0$

$f_4 = -y + 5 \geq 0$

		x	y
f_1	0	1	1
f_2	-10	1	0
f_3	-5	0	1
f_4	5	0	-1

		x	f_3
f_1	5	1	1
f_2	-10	1	0
y	5	0	1
f_4	0	0	-1

- a pivot interchanges a row and a column variable
  $f_3 = -5 + y \Rightarrow y = 5 + f_3,\ f_1 = 5 + x + f_3,\ f_4 = -f_3$
- a column variable that is known to be zero (e.g., f_3) can be “killed”
  - $\Rightarrow$ fixed to zero
  - $\Rightarrow$ no longer participates in pivoting

Incremental LP Solver

$$f_1 = x + y \geq 0$$
$$f_2 = x - 10 \geq 0$$
$$f_3 = y - 5 \geq 0$$
$$f_4 = -y + 5 \geq 0$$

		x	y
f_1	0	1	1
f_2	−10	1	0
f_3	−5	0	1
f_4	5	0	−1

		x	f_3
f_1	5	1	1
f_2	−10	1	0
y	5	0	1
f_4	0	0	−1

		f_3	x
f_1	5	1	1
f_2	−10	0	1
y	5	1	0
f_4	0	−1	0

- a pivot interchanges a row and a column variable
  $f_3 = -5 + y \Rightarrow y = 5 + f_3,\ f_1 = 5 + x + f_3,\ f_4 = -f_3$
- a column variable that is known to be zero (e.g., f_3) can be "killed"
  - $\Rightarrow$ fixed to zero
  - $\Rightarrow$ no longer participates in pivoting

Incremental LP Solver

$f_1 = x + y \geq 0$

$f_2 = x - 10 \geq 0$

$f_3 = y - 5 \geq 0$

$f_4 = -y + 5 \geq 0$

		x	y
f_1	0	1	1
f_2	-10	1	0
f_3	-5	0	1
f_4	5	0	-1

		x	f_3
f_1	5	1	1
f_2	-10	1	0
y	5	0	1
f_4	0	0	-1

		f_3	x
f_1	5	1	1
f_2	-10	0	1
y	5	1	0
f_4	0	-1	0

- a pivot interchanges a row and a column variable
  $f_3 = -5 + y \Rightarrow y = 5 + f_3,\ f_1 = 5 + x + f_3,\ f_4 = -f_3$
- a column variable that is known to be zero (e.g., f_3) can be "killed"
  - $\Rightarrow$ fixed to zero
  - $\Rightarrow$ no longer participates in pivoting
- sample value is "valid" if
  - all non-negative variables have non-negative value
  - all variables have integer value

$\Rightarrow$ current sample value not valid because $f_2 = -10$

Disequality Constraints in Tableau

Introduce a non-zero variable for each disequality constraint

- a non-zero variable does not participate in pivoting
  - ⇒ always a row variable
- if non-zero variable can attain only negative or only positive values (while non-negative variables have non-negative values)
  - ⇒ non-zero variable is redundant and can be removed
- if non-zero variable can obviously only attain zero value
  - zero sample value
  - all coefficients in non-killed columns are zero
  - ⇒ tableau is empty

Disequality Constraints in Tableau

Introduce a non-zero variable for each disequality constraint

- a non-zero variable does not participate in pivoting
  - ⇒ always a row variable
- if non-zero variable can attain only negative or only positive values (while non-negative variables have non-negative values)
  - ⇒ non-zero variable is redundant and can be removed
- if non-zero variable can obviously only attain zero value
  - zero sample value
  - all coefficients in non-killed columns are zero
  - ⇒ tableau is empty

$$f_1 = x + y \geq 0$$
$$f_2 = x - 10 \geq 0$$
$$f_3 = y - 5 \geq 0$$
$$f_4 = -y + 5 \geq 0$$

		x	y
f_1	0	1	1
f_2	-10	1	0
f_3	-5	0	1
f_4	5	0	-1

		x	f_3
f_1	5	1	1
f_2	-10	1	0
y	5	0	1
f_4	0	0	-1

		f_3	x
f_1	5	1	1
f_2	-10	0	1
y	5	1	0
f_4	0	-1	0

Disequality Constraints in Tableau

Introduce a non-zero variable for each disequality constraint

- a non-zero variable does not participate in pivoting
  - ⇒ always a row variable
- if non-zero variable can attain only negative or only positive values (while non-negative variables have non-negative values)
  - ⇒ non-zero variable is redundant and can be removed
- if non-zero variable can obviously only attain zero value
  - ▸ zero sample value
  - ▸ all coefficients in non-killed columns are zero
  - ⇒ tableau is empty

$$f_1 = x + y \geq 0$$
$$f_2 = x - 10 \geq 0$$
$$f_3 = y - 5 \geq 0$$
$$f_4 = -y + 5 \geq 0$$
$$f_5 = y - 5 \neq 0$$

		x	y
f_1	0	1	1
f_2	-10	1	0
f_3	-5	0	1
f_4	5	0	-1

		x	f_3
f_1	5	1	1
f_2	-10	1	0
y	5	0	1
f_4	0	0	-1

		f_3	x
f_1	5	1	1
f_2	-10	0	1
y	5	1	0
f_4	0	-1	0

Disequality Constraints in Tableau

Introduce a non-zero variable for each disequality constraint

- a non-zero variable does not participate in pivoting
  - $\Rightarrow$ always a row variable
- if non-zero variable can attain only negative or only positive values (while non-negative variables have non-negative values)
  - $\Rightarrow$ non-zero variable is redundant and can be removed
- if non-zero variable can obviously only attain zero value
  - zero sample value
  - all coefficients in non-killed columns are zero
  - $\Rightarrow$ tableau is empty

$$f_1 = x + y \geq 0$$
$$f_2 = x - 10 \geq 0$$
$$f_3 = y - 5 \geq 0$$
$$f_4 = -y + 5 \geq 0$$
$$f_5 = y - 5 \neq 0$$

		x	y
f_1	0	1	1
f_2	-10	1	0
f_3	-5	0	1
f_4	5	0	-1
f_5	-5	0	1

		x	f_3
f_1	5	1	1
f_2	-10	1	0
y	5	0	1
f_4	0	0	-1

		f_3	x
f_1	5	1	1
f_2	-10	0	1
y	5	1	0
f_4	0	-1	0

Disequality Constraints in Tableau

Introduce a non-zero variable for each disequality constraint

- a non-zero variable does not participate in pivoting
  - $\Rightarrow$ always a row variable
- if non-zero variable can attain only negative or only positive values (while non-negative variables have non-negative values)
  - $\Rightarrow$ non-zero variable is redundant and can be removed
- if non-zero variable can obviously only attain zero value
  - zero sample value
  - all coefficients in non-killed columns are zero
  - $\Rightarrow$ tableau is empty

$$f_1 = x + y \geq 0$$
$$f_2 = x - 10 \geq 0$$
$$f_3 = y - 5 \geq 0$$
$$f_4 = -y + 5 \geq 0$$
$$f_5 = y - 5 \neq 0$$

		x	y
f_1	0	1	1
f_2	-10	1	0
f_3	-5	0	1
f_4	5	0	-1
f_5	-5	0	1

		x	f_3
f_1	5	1	1
f_2	-10	1	0
y	5	0	1
f_4	0	0	-1
f_5	0	0	1

		f_3	x
f_1	5	1	1
f_2	-10	0	1
y	5	1	0
f_4	0	-1	0

Disequality Constraints in Tableau

Introduce a non-zero variable for each disequality constraint

- a non-zero variable does not participate in pivoting
  - $\Rightarrow$ always a row variable
- if non-zero variable can attain only negative or only positive values (while non-negative variables have non-negative values)
  - $\Rightarrow$ non-zero variable is redundant and can be removed
- if non-zero variable can obviously only attain zero value
  - zero sample value
  - all coefficients in non-killed columns are zero
  - $\Rightarrow$ tableau is empty

$$f_1 = x + y \geq 0$$
$$f_2 = x - 10 \geq 0$$
$$f_3 = y - 5 \geq 0$$
$$f_4 = -y + 5 \geq 0$$
$$f_5 = y - 5 \neq 0$$

		x	y
f_1	0	1	1
f_2	-10	1	0
f_3	-5	0	1
f_4	5	0	-1
f_5	-5	0	1

		x	f_3
f_1	5	1	1
f_2	-10	1	0
y	5	0	1
f_4	0	0	-1
f_5	0	0	1

		f_3	x
f_1	5	1	1
f_2	-10	0	1
y	5	1	0
f_4	0	-1	0
f_5	0	1	0

Disequality Constraints in Tableau

Introduce a non-zero variable for each disequality constraint

- a non-zero variable does not participate in pivoting
  - $\Rightarrow$ always a row variable
- if non-zero variable can attain only negative or only positive values (while non-negative variables have non-negative values)
  - $\Rightarrow$ non-zero variable is redundant and can be removed
- if non-zero variable can obviously only attain zero value
  - zero sample value
  - all coefficients in non-killed columns are zero
  - $\Rightarrow$ tableau is empty

Effect on sample value validity

- sample value is "valid" if
  - all non-zero variables have non-zero value
  - all non-negative variables have non-negative value
  - all variables have integer value

References I

- Bagnara, Roberto, Patricia M. Hill, and Enea Zaffanella (2008). "The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems". In: *Science of Computer Programming* 72.1–2, pp. 3–21. DOI: 10.1016/j.scico.2007.08.001.
- Chen, Chun (June 2012). "Polyhedra scanning revisited". In: *SIGPLAN Not.* 47.6, pp. 499–508. DOI: 10.1145/2345156.2254123.
- Kelly, Wayne, Vadim Maslov, William Pugh, Evan Rosser, Tatiana Shpeisman, and David Wonnacott (Nov. 1996). *The Omega Library*. Tech. rep. University of Maryland.
- Klebanov, Vladimir (2015). *Personal communication*.
- Klöckner, Andreas (2014). "Loo.Py: Transformation-based Code Generation for GPUs and CPUs". In: *Proceedings of ACM SIGPLAN International Workshop on Libraries, Languages, and Compilers for Array Programming*. ARRAY'14. Edinburgh, United Kingdom: ACM, 82:82–82:87. DOI: 10.1145/2627373.2627387.

References II

- Kulkarni, Shubhang and Michael Kruse (June 2022). "Polyhedral Binary Decision Diagrams for Representing Non-Convex Polyhedra". In: *12th International Workshop on Polyhedral Compilation Techniques (IMPACT 2022)*. Budapest, Hungary.
- Pitchanathan, Arjun, Christian Ulmann, Michel Weber, Torsten Hoefler, and Tobias Grosser (Oct. 2021). "FPL: Fast Presburger Arithmetic through Transprecision". In: *Proc. ACM Program. Lang.* 5.OOPSLA. DOI: 10.1145/3485539.
- Seater, Robert and David Wonnacott (2005). "Efficient Manipulation of Disequalities During Dependence Analysis". In: *Proceedings of the 15th International Conference on Languages and Compilers for Parallel Computing*. LCPC'02. College Park, MD: Springer-Verlag, pp. 295–308. DOI: 10.1007/11596110_20.
- V., Sven (2010). "isl: An Integer Set Library for the Polyhedral Model". In: *Mathematical Software - ICMS 2010*. Ed. by Komei Fukuda, Joris Hoeven, Michael Joswig, and Nobuki Takayama. Vol. 6327. Lecture Notes in Computer Science. Springer, pp. 299–302. DOI: 10.1007/978-3-642-15582-6_49.

References III

- V., Sven (Jan. 2024). "A Polyhedral Compilation Library with Explicit Disequality Constraints". In: *14th International Workshop on Polyhedral Compilation Techniques (IMPACT 2024)*. Munich, Geermany. DOI: 10.5281/zenodo.10511210.
- V., Sven and Tobias Grosser (Jan. 2012). "Polyhedral Extraction Tool". In: *Second International Workshop on Polyhedral Compilation Techniques (IMPACT'12)*. Paris, France. DOI: 10.13140/RG.2.1.4213.4562.
- V., Sven, Juan Carlos Juega, Albert Cohen, José Ignacio Gómez, Christian Tenllado, and Francky Catthoor (2013). "Polyhedral parallel code generation for CUDA". In: *ACM Trans. Archit. Code Optim.* 9.4, p. 54. DOI: 10.1145/2400682.2400713.
- V., Sven, Rachid Seghir, Kristof Beyls, Vincent Loechner, and Maurice Bruynooghe (June 2007). "Counting integer points in parametric polytopes using Barvinok's rational functions". In: *Algorithmica* 48.1, pp. 37–66. DOI: 10.1007/s00453-006-1231-0.
- Wilde, Doran K. (1993). *A Library for doing polyhedral operations*. Tech. rep. 785. IRISA, Rennes, France, 45 p.