isl: An Integer Set Library for the Polyhedral Model

Sven Verdoolaege

Department of Computer Science, Katholieke Universiteit Leuven, Belgium
Team ALCHEMY, INRIA Saclay, France
Sven.Verdoolaege@{cs.kuleuven.be,inria.fr}

September 15, 2010

Outline

Outline

An Integer Set Library

`isl` is an **LGPL thread-safe C** library for manipulating **sets and relations** of **integer tuples** bounded by **affine constraints**

$\rightsquigarrow$ finite unions of projections of parametric lattice polytopes

An Integer Set Library

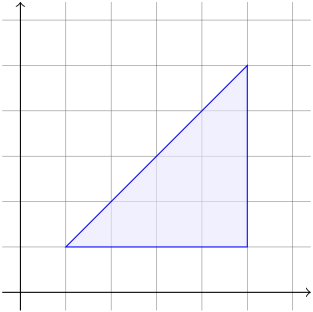
`isl` is an **LGPL thread-safe C** library for manipulating
sets and relations of **integer tuples** bounded by **affine constraints**

$\rightsquigarrow$ finite unions of projections of parametric lattice polytopes

- very similar to `Omega` and `Omega+` libraries
- similar to `polymake`, but different focus/philosophy
    - integer values instead of rational values
    - designed for the polyhedral model for program analysis and transformation (but also useful for other applications)
    - library ("calculator" interface is available too)
      $\Rightarrow$ embeddable in a compiler
    - works best on sets of small dimensions (up to about 10; some operations also work for higher dimensions)
    - self-contained (apart from GMP)
    - closed representation
    - objects may be sets or relations (or piecewise quasipolynomials)

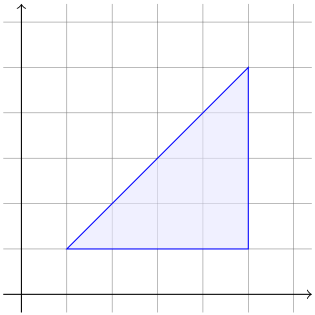
Examples of Sets and Relations

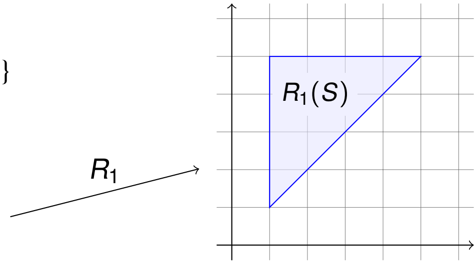
$$S = \{ (x, y) \mid 1 \leq y \leq x \leq 5 \}$$



Examples of Sets and Relations

$S = \{ (x, y) \mid 1 \le y \le x \le 5 \}$

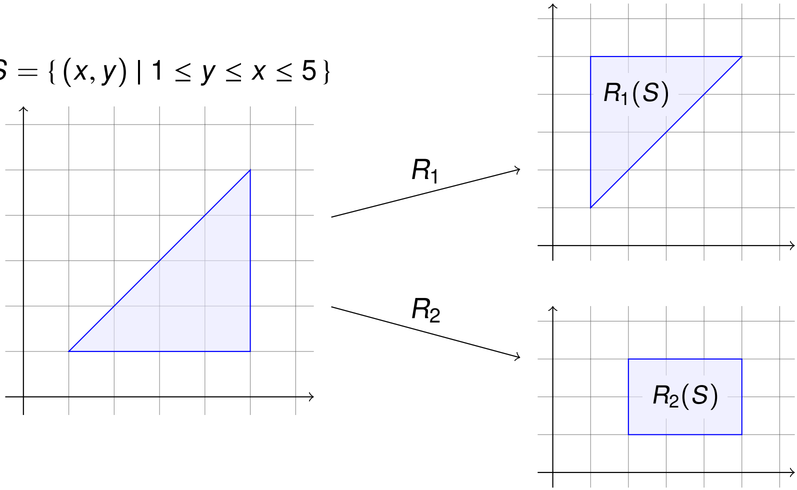




$$R_1 = \{ (x, y) \to (y, x) \} = \{ (x, y) \to (x', y') \mid x' = y \wedge y' = x \}$$

Examples of Sets and Relations

$$S = \{\, (x, y) \mid 1 \leq y \leq x \leq 5 \,\}$$



$$R_1 = \{\, (x, y) \to (y, x) \,\} = \{\, (x, y) \to (x', y') \mid x' = y \wedge y' = x \,\}$$
$$R_2 = \{\, (x, y) \to (x, y') \mid x \geq 2 \wedge 1 \leq y' \leq 3 \,\}$$

Sets and Relations in the Polyhedral Model

```
for (i = 0; i < n; ++i)
    for (j = 0; j < i; ++j)
        f(a[j][i+j][2*i]);
```

Typical sets and relations

- Iteration domain
  ⇒ set of all possible values of the iterators

$$n \to \{ (i,j) \mid 0 \le i < n \wedge 0 \le j < i \}$$

- Access relation
  ⇒ maps iteration vector to array index

$$\{ (i,j) \to (j, i+j, 2i) \}$$

Comparison to Related Libraries

- Compared to double description based libraries (`PolyLib`, `PPL`)
  - All operations are performed on constraints

    Reason: objects in target application domain usually have few constraints, but may have many vertices
  - Full support for parameters
  - Built-in support for existentially quantified variables
  - Built-in support for relations
  - Focus on integer values

Comparison to Related Libraries

- Compared to double description based libraries (`PolyLib`, `PPL`)
  - All operations are performed on constraints

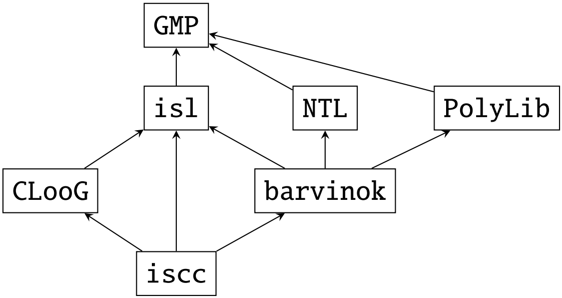
    Reason: objects in target application domain usually have few constraints, but may have many vertices
  - Full support for parameters
  - Built-in support for existentially quantified variables
  - Built-in support for relations
  - Focus on integer values
- Compared to `Omega` and `Omega+`
  - All operations are performed in arbitrary integer arithmetic using GMP
  - Different way of handling existentially quantified variables
  - Named and nested spaces
  - Parametric vertex enumeration

    ⇒ useful for the `barvinok` counting library and for computing bounds
  - Support for piecewise quasipolynomials

    ⇒ results of counting problems

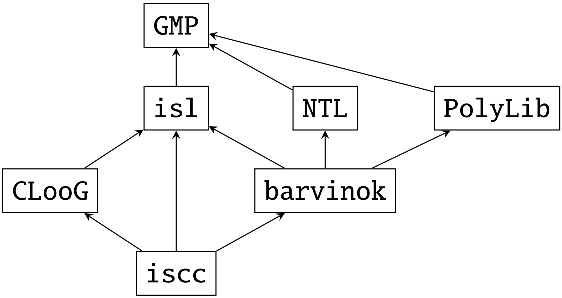
Interaction with Other Libraries and Tools

barvinok: counts elements in parametric affine sets and relations
CLooG: generates code to scan elements in parametric affine sets
iscc: interactive isl calculator (included in barvinok distribution)



Interaction with Other Libraries and Tools

barvinok: counts elements in parametric affine sets and relations
CLooG: generates code to scan elements in parametric affine sets
iscc: interactive isl calculator (included in barvinok distribution)



Future work:

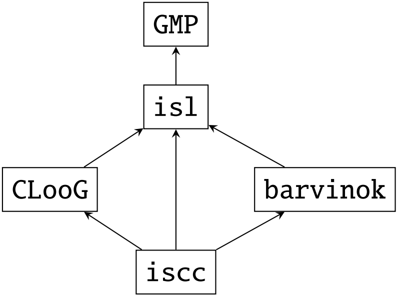
- remove dependence on PolyLib and NTL

Interaction with Other Libraries and Tools

`barvinok`: counts elements in parametric affine sets and relations

`CLooG`: generates code to scan elements in parametric affine sets

`iscc`: interactive `isl` calculator (included in `barvinok` distribution)

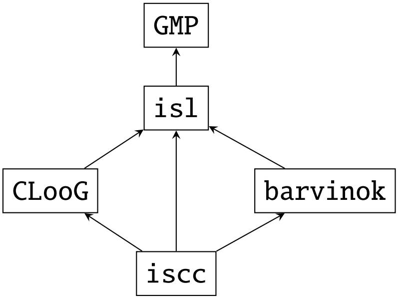


Future work:

- remove dependence on `PolyLib` and `NTL`

Interaction with Other Libraries and Tools

barvinok: counts elements in parametric affine sets and relations
CLooG: generates code to scan elements in parametric affine sets
iscc: interactive isl calculator (included in barvinok distribution)

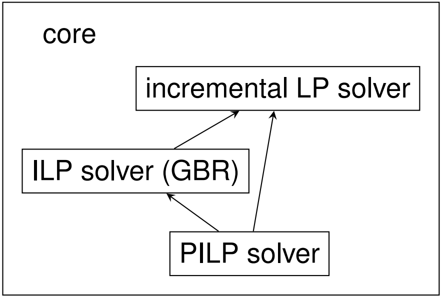


Future work:

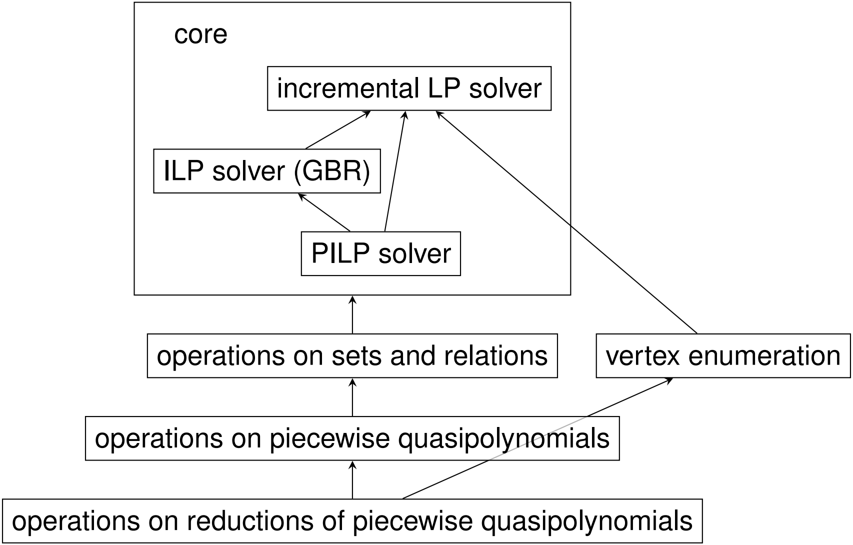
- remove dependence on PolyLib and NTL
- merge barvinok into isl

Outline

Internal Structure



Internal Structure



Internal Representation

$$S(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \exists \mathbf{z} \in \mathbb{Z}^e : A\mathbf{x} + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

$$R(\mathbf{s}) = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z} \in \mathbb{Z}^e : A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

- “basic” types: “convex” sets and maps (relations)
  - equality + inequality constraints
  - parameters **s**
  - (optional) explicit representation of existentially quantified variables as integer divisions
    - ⇒ useful for aligning dimensions when performing set operations (e.g., set difference)
    - ⇒ can be computed using PILP

Internal Representation

$$S(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \exists \mathbf{z} \in \mathbb{Z}^e : A\mathbf{x} + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$
$$R(\mathbf{s}) = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z} \in \mathbb{Z}^e : A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

- "basic" types: "convex" sets and maps (relations)
  - equality + inequality constraints
  - parameters **s**
  - (optional) explicit representation of existentially quantified variables as integer divisions
    - ⇒ useful for aligning dimensions when performing set operations (e.g., set difference)
    - ⇒ can be computed using PILP
- sets and maps
  - ⇒ (disjoint) unions of basic sets/maps

Internal Representation

$$S(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \exists \mathbf{z} \in \mathbb{Z}^e : A\mathbf{x} + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

$$R(\mathbf{s}) = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z} \in \mathbb{Z}^e : A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

- “basic” types: “convex” sets and maps (relations)
  - equality + inequality constraints
  - parameters $\mathbf{s}$
  - (optional) explicit representation of existentially quantified variables as integer divisions
    - ⇒ useful for aligning dimensions when performing set operations (e.g., set difference)
    - ⇒ can be computed using PILP
- sets and maps
  - ⇒ (disjoint) unions of basic sets/maps
- union sets and union maps
  - ⇒ unions of sets/maps in different spaces

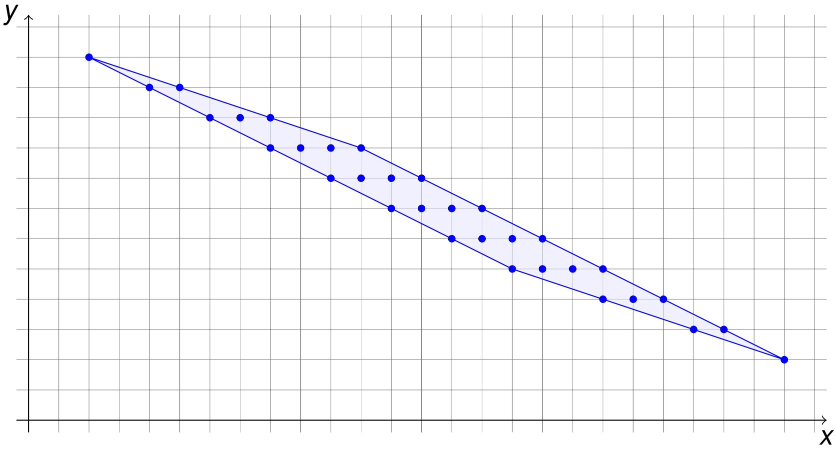
Parametric Integer Linear Programming

$$R(\mathbf{s}) = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z} \in \mathbb{Z}^{e} : A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

Lexicographic minimum of R:

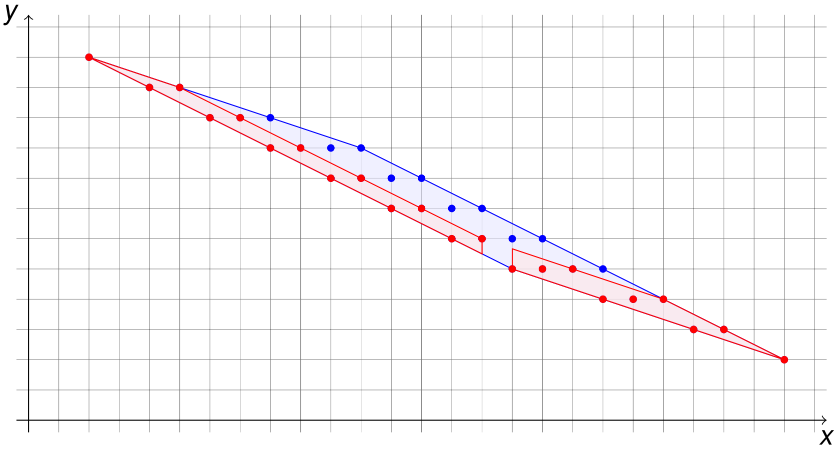
$$\operatorname{lexmin} R = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in R \mid \forall \mathbf{x}_2' \in R(\mathbf{s}, \mathbf{x}_1) : \mathbf{x}_2 \preccurlyeq \mathbf{x}_2' \,\}$$

Parametric Integer Linear Programming Example



$$R = \{ x \to y \mid 3y \geq 31 - x \wedge 2y \leq 29 - x \wedge 3y \leq 38 - x \wedge 2y \geq 26 - x \}$$

Parametric Integer Linear Programming Example



$R = \{ x \to y \mid 3y \geq 31 - x \wedge 2y \leq 29 - x \wedge 3y \leq 38 - x \wedge 2y \geq 26 - x \}$

$\text{lexmin}\, R = \{ x \to y \mid (x \leq 25 \wedge x \geq 16 \wedge 3y \geq 31 - x \wedge 3y \leq 33 - x \wedge 2y \leq 29 - x) \vee (3y \leq 38 - x \wedge x \leq 15 \wedge x \geq 2 \wedge 2y \geq 26 - x \wedge 2y \leq 27 - x) \}$

Parametric Integer Linear Programming

$$R(\mathbf{s}) = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z} \in \mathbb{Z}^{e} : A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

Lexicographic minimum of R:

$$\operatorname{lexmin} R = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in R \mid \forall \mathbf{x}_2' \in R(\mathbf{s}, \mathbf{x}_1) : \mathbf{x}_2 \preccurlyeq \mathbf{x}_2' \,\}$$

Parametric Integer Linear Programming

$$R(\mathbf{s}) = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z} \in \mathbb{Z}^{e} : A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

Lexicographic minimum of R:

$$\operatorname{lexmin} R = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in R \mid \forall \mathbf{x}_2' \in R(\mathbf{s}, \mathbf{x}_1) : \mathbf{x}_2 \preccurlyeq \mathbf{x}_2' \,\}$$

Parametric integer linear programming computes $\operatorname{lexmin} R$ in the form

$$\operatorname{lexmin} R = \bigcup_i \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z}' \in \mathbb{Z}^{e'} : A_i\mathbf{x}_1 + B_i\mathbf{s} \geq \mathbf{c}_i \wedge \mathbf{z}' = \left\lfloor \frac{P_i\mathbf{x}_1 + Q_i\mathbf{s} + \mathbf{r}_i}{m} \right\rfloor \wedge \mathbf{x}_2 = T_i\mathbf{x}_1 + U_i\mathbf{s} + V_i\mathbf{z}' + \mathbf{w}_i \,\}$$

- explicit representation of existentially quantified variables
- explicit representation of range variables

Technique: dual simplex + Gomory cuts

Parametric Integer Linear Programming

$$R(\mathbf{s}) = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z} \in \mathbb{Z}^{e} : A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

Lexicographic minimum of R:

$$\operatorname{lexmin} R = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in R \mid \forall \mathbf{x}_2' \in R(\mathbf{s}, \mathbf{x}_1) : \mathbf{x}_2 \preccurlyeq \mathbf{x}_2' \,\}$$

Parametric integer linear programming computes $\operatorname{lexmin} R$ in the form

$$\operatorname{lexmin} R = \bigcup_i \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z}' \in \mathbb{Z}^{e'} : A_i\mathbf{x}_1 + B_i\mathbf{s} \geq \mathbf{c}_i \wedge \mathbf{z}' = \left\lfloor \frac{P_i\mathbf{x}_1 + Q_i\mathbf{s} + \mathbf{r}_i}{m} \right\rfloor \wedge \mathbf{x}_2 = T_i\mathbf{x}_1 + U_i\mathbf{s} + V_i\mathbf{z}' + \mathbf{w}_i \,\}$$

- explicit representation of existentially quantified variables
- explicit representation of range variables

Technique: dual simplex + Gomory cuts

Parametric Integer Linear Programming

$$R(\mathbf{s}) = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z} \in \mathbb{Z}^{e} : A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

Lexicographic minimum of R:

$$\operatorname{lexmin} R = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in R \mid \forall \mathbf{x}_2' \in R(\mathbf{s}, \mathbf{x}_1) : \mathbf{x}_2 \preccurlyeq \mathbf{x}_2' \,\}$$

Parametric integer linear programming computes $\operatorname{lexmin} R$ in the form

$$\operatorname{lexmin} R = \bigcup_i \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z}' \in \mathbb{Z}^{e'} : A_i\mathbf{x}_1 + B_i\mathbf{s} \geq \mathbf{c}_i \wedge \mathbf{z}' = \left\lfloor \frac{P_i\mathbf{x}_1 + Q_i\mathbf{s} + \mathbf{r}_i}{m} \right\rfloor \wedge \mathbf{x}_2 = T_i\mathbf{x}_1 + U_i\mathbf{s} + V_i\mathbf{z}' + \mathbf{w}_i \,\}$$

- explicit representation of existentially quantified variables
- explicit representation of range variables

Technique: dual simplex + Gomory cuts

PILP Example: Dataflow Analysis

Given a read from an array element, what was the last write to the same array element before the read?

Simple case: array written through a single access

```
for (i = 0; i < N; ++i)
    for (j = 0; j < N - i; ++j)
        a[i+j] = f(a[i+j]);
for (i = 0; i < N; ++i)
    Write(a[i]);
```

PILP Example: Dataflow Analysis

Given a read from an array element, what was the last write to the same array element before the read?

Simple case: array written through a single access

```
for (i = 0; i < N; ++i)
    for (j = 0; j < N - i; ++j)
        a[i+j] = f(a[i+j]);
for (i = 0; i < N; ++i)
    Write(a[i]);
```

PILP Example: Dataflow Analysis

Given a read from an array element, what was the last write to the same array element before the read?

Simple case: array written through a single access

```
for (i = 0; i < N; ++i)
    for (j = 0; j < N - i; ++j)
        a[i+j] = f(a[i+j]);
for (i = 0; i < N; ++i)
    Write(a[i]);
```

Access relations:

$A_1 = \{(i, j) \to (i + j) \mid 0 \le i < N \wedge 0 \le j < N - i\}$

$A_2 = \{(i) \to (i) \mid 0 \le i < N\}$

PILP Example: Dataflow Analysis

Given a read from an array element, what was the last write to the same array element before the read?

Simple case: array written through a single access

```
for (i = 0; i < N; ++i)
    for (j = 0; j < N - i; ++j)
        a[i+j] = f(a[i+j]);
for (i = 0; i < N; ++i)
    Write(a[i]);
```

Access relations:

$A_1 = \{(i,j) \to (i+j) \mid 0 \leq i < N \wedge 0 \leq j < N - i\}$

$A_2 = \{(i) \to (i) \mid 0 \leq i < N\}$

Map to all writes: $R' = A_1{}^{-1} \circ A_2 = \{(i) \to (i', i - i') \mid 0 \leq i' \leq i < N\}$

PILP Example: Dataflow Analysis

Given a read from an array element, what was the last write to the same array element before the read?

Simple case: array written through a single access

```
for (i = 0; i < N; ++i)
    for (j = 0; j < N - i; ++j)
        a[i+j] = f(a[i+j]);
for (i = 0; i < N; ++i)
    Write(a[i]);
```

Access relations:

$A_1 = \{(i, j) \to (i + j) \mid 0 \leq i < N \wedge 0 \leq j < N - i\}$

$A_2 = \{(i) \to (i) \mid 0 \leq i < N\}$

Map to all writes: $R' = {A_1}^{-1} \circ A_2 = \{(i) \to (i', i - i') \mid 0 \leq i' \leq i < N\}$

Last write: $R = \text{lexmax}\, R' = \{(i) \to (i, 0) \mid 0 \leq i < N\}$

PILP Example: Dataflow Analysis

Given a read from an array element, what was the last write to the same array element before the read?

Simple case: array written through a single access

```
for (i = 0; i < N; ++i)
    for (j = 0; j < N - i; ++j)
        a[i+j] = f(a[i+j]);
for (i = 0; i < N; ++i)
    Write(a[i]);
```

Access relations:

$A_1 = \{(i,j) \to (i+j) \mid 0 \leq i < N \wedge 0 \leq j < N - i\}$

$A_2 = \{(i) \to (i) \mid 0 \leq i < N\}$

Map to all writes: $R' = {A_1}^{-1} \circ A_2 = \{(i) \to (i', i - i') \mid 0 \leq i' \leq i < N\}$

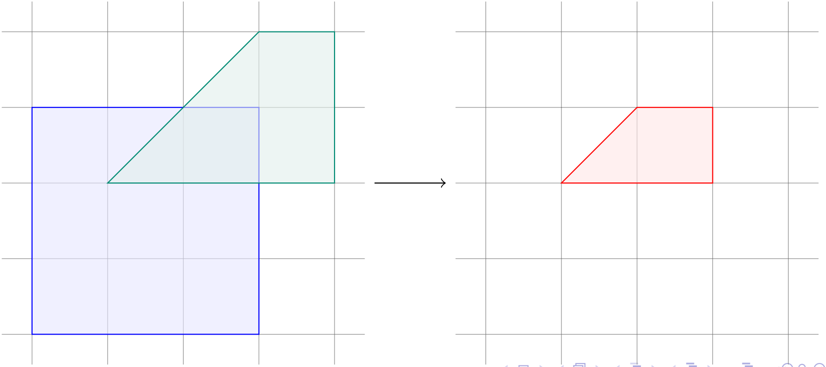
Last write: $R = \operatorname{lexmax} R' = \{(i) \to (i, 0) \mid 0 \leq i < N\}$

In general: impose lexicographical order on shared iterators

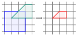
Outline

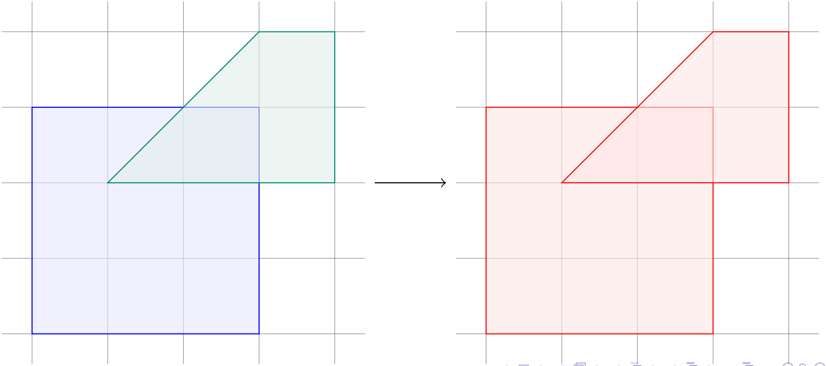
Supported Operations

- Intersection

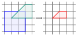
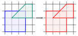


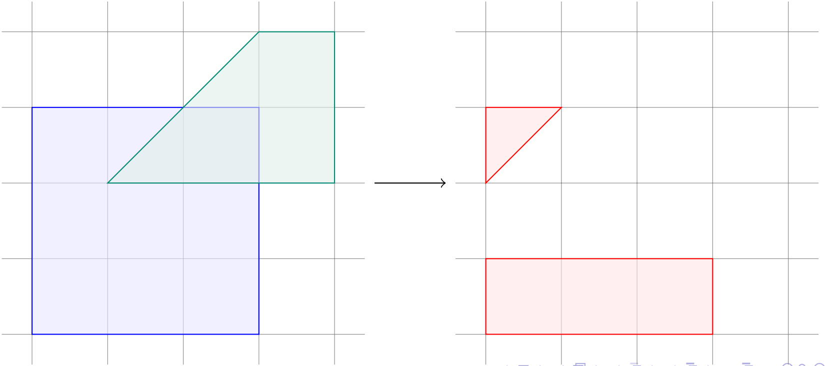
Supported Operations

- Intersection 
- Union



Supported Operations

- Intersection 
- Union 
- Set difference



isl Operation: Set Difference

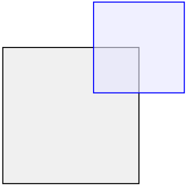
$$S(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \exists \mathbf{z} \in \mathbb{Z}^e : A\mathbf{x} + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

Set difference $S_1 \setminus S_2$

- no existentially quantified variables

$$S_2(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \bigwedge_i \langle \mathbf{a}_i, \mathbf{x} \rangle + \langle \mathbf{b}_i, \mathbf{s} \rangle \geq c_i \,\}$$

$$S_1 \setminus S_2 = \bigcup_i \left(S_1 \cap \{\, \mathbf{x} \mid \neg(\langle \mathbf{a}_i, \mathbf{x} \rangle + \langle \mathbf{b}_i, \mathbf{s} \rangle \geq c_i) \,\} \right)$$



isl Operation: Set Difference

$$S(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \exists \mathbf{z} \in \mathbb{Z}^e : A\mathbf{x} + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

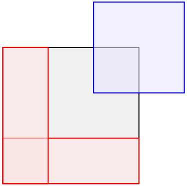
Set difference $S_1 \setminus S_2$

- no existentially quantified variables

$$S_2(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \bigwedge_i \langle \mathbf{a}_i, \mathbf{x} \rangle + \langle \mathbf{b}_i, \mathbf{s} \rangle \geq c_i \,\}$$

$$S_1 \setminus S_2 = \bigcup_i \left(S_1 \cap \{\, \mathbf{x} \mid \neg(\langle \mathbf{a}_i, \mathbf{x} \rangle + \langle \mathbf{b}_i, \mathbf{s} \rangle \geq c_i) \,\}\right)$$

$$= \bigcup_i \left(S_1 \cap \{\, \mathbf{x} \mid \langle \mathbf{a}_i, \mathbf{x} \rangle + \langle \mathbf{b}_i, \mathbf{s} \rangle \leq c_i - 1 \,\}\right)$$



isl Operation: Set Difference

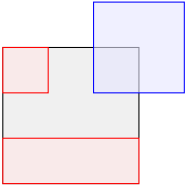
$$S(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \exists \mathbf{z} \in \mathbb{Z}^e : A\mathbf{x} + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

Set difference $S_1 \setminus S_2$

- no existentially quantified variables

$$S_2(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \bigwedge_i \langle \mathbf{a}_i, \mathbf{x} \rangle + \langle \mathbf{b}_i, \mathbf{s} \rangle \geq c_i \,\}$$

$$\begin{aligned} S_1 \setminus S_2 &= \bigcup_i \left(S_1 \cap \{\, \mathbf{x} \mid \neg(\langle \mathbf{a}_i, \mathbf{x} \rangle + \langle \mathbf{b}_i, \mathbf{s} \rangle \geq c_i) \,\} \right) \\ &= \bigcup_i \left(S_1 \cap \{\, \mathbf{x} \mid \langle \mathbf{a}_i, \mathbf{x} \rangle + \langle \mathbf{b}_i, \mathbf{s} \rangle \leq c_i - 1 \,\} \right) \\ &= \bigcup_i \Big(S_1 \cap \bigcap_{j<i} \{\, \mathbf{x} \mid \langle \mathbf{a}_j, \mathbf{x} \rangle + \langle \mathbf{b}_j, \mathbf{s} \rangle \geq c_j \,\} \\ &\qquad \cap \{\, \mathbf{x} \mid \langle \mathbf{a}_i, \mathbf{x} \rangle + \langle \mathbf{b}_i, \mathbf{s} \rangle \leq c_i - 1 \,\} \Big) \end{aligned}$$



isl Operation: Set Difference

$$S(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \exists \mathbf{z} \in \mathbb{Z}^e : A\mathbf{x} + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

Set difference $S_1 \setminus S_2$

- no existentially quantified variables

$$S_2(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \bigwedge_i \langle \mathbf{a}_i, \mathbf{x} \rangle + \langle \mathbf{b}_i, \mathbf{s} \rangle \geq c_i \,\}$$

$$S_1 \setminus S_2 = \bigcup_i \Big(S_1 \cap \bigcap_{j<i} \{\, \mathbf{x} \mid \big\langle \mathbf{a}_j, \mathbf{x} \big\rangle + \big\langle \mathbf{b}_j, \mathbf{s} \big\rangle \geq c_j \,\}$$

$$\cap \{\, \mathbf{x} \mid \langle \mathbf{a}_i, \mathbf{x} \rangle + \langle \mathbf{b}_i, \mathbf{s} \rangle \leq c_i - 1 \,\} \Big)$$

isl Operation: Set Difference

$$S(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \exists \mathbf{z} \in \mathbb{Z}^e : A\mathbf{x} + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

Set difference $S_1 \setminus S_2$

- no existentially quantified variables

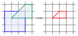
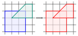
$$S_2(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \bigwedge_i \langle \mathbf{a}_i, \mathbf{x} \rangle + \langle \mathbf{b}_i, \mathbf{s} \rangle \geq c_i \,\}$$

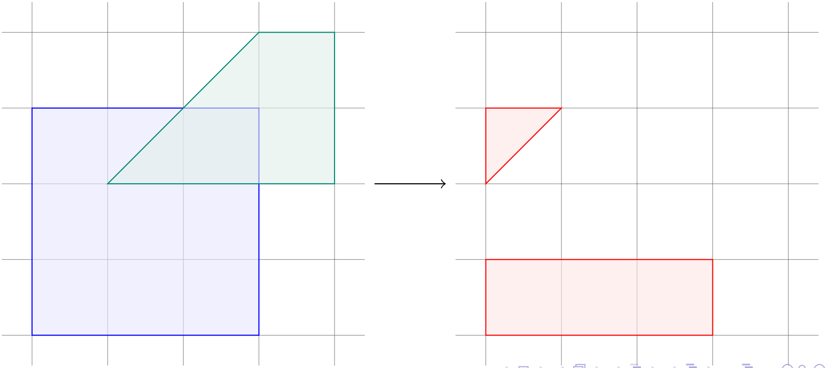
$$S_1 \setminus S_2 = \bigcup_i \Big(S_1 \cap \bigcap_{j<i} \{\, \mathbf{x} \mid \big\langle \mathbf{a}_j, \mathbf{x} \big\rangle + \big\langle \mathbf{b}_j, \mathbf{s} \big\rangle \geq c_j \,\} \\ \cap \{\, \mathbf{x} \mid \langle \mathbf{a}_i, \mathbf{x} \rangle + \langle \mathbf{b}_i, \mathbf{s} \rangle \leq c_i - 1 \,\}\Big)$$

- with existentially quantified variables
  $\Rightarrow$ compute explicit representation

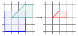
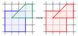
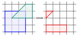
$$S_2(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \bigwedge_i \langle \mathbf{a}_i, \mathbf{x} \rangle + \langle \mathbf{b}_i, \mathbf{s} \rangle + \left\langle \mathbf{d}_i, \left\lfloor \frac{\langle \mathbf{p}, \mathbf{x} \rangle + \langle \mathbf{q}_i, \mathbf{s} \rangle + r}{m} \right\rfloor \right\rangle \geq c_i \,\}$$

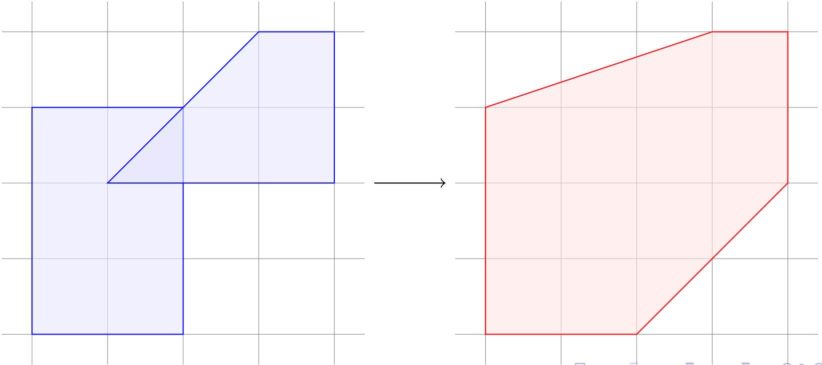
Supported Operations

- Intersection 
- Union 
- Set difference

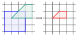
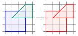
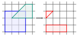
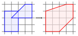


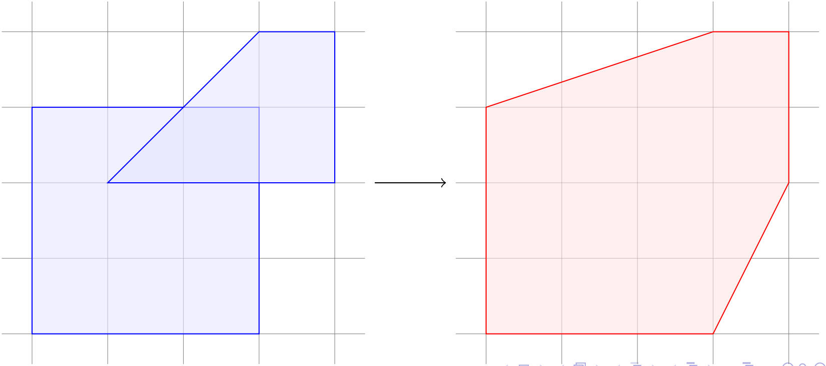
Supported Operations

- Intersection 
- Union 
- Set difference 
- Closed convex hull (“wrapping”, FLL2000)



Supported Operations

- Intersection 
- Union 
- Set difference 
- Closed convex hull (“wrapping”, FLL2000) 
- Coalescing



isl Operation: Set Coalescing

After many applications of projection, set difference, union,
a set may be represented as a union of many basic sets
⇒ try to combine several basic sets into a single basic set

isl Operation: Set Coalescing

After many applications of projection, set difference, union, a set may be represented as a union of many basic sets
⇒ try to combine several basic sets into a single basic set

$$S_1 = \{\, \mathbf{x} \mid A\mathbf{x} \geq \mathbf{c} \,\} \qquad S_2 = \{\, \mathbf{x} \mid B\mathbf{x} \geq \mathbf{d} \,\}$$

PolyLib way:

1. Compute $H = \mathrm{conv.hull}(S_1 \cup S_2)$
2. Replace $S_1 \cup S_2$ by $H \setminus (H \setminus (S_1 \cup S_2))$

`isl` Operation: Set Coalescing

After many applications of projection, set difference, union, a set may be represented as a union of many basic sets
⇒ try to combine several basic sets into a single basic set

$$S_1 = \{ \mathbf{x} \mid A\mathbf{x} \geq \mathbf{c} \} \qquad S_2 = \{ \mathbf{x} \mid B\mathbf{x} \geq \mathbf{d} \}$$

PolyLib way:

1. Compute $H = \mathrm{conv.hull}(S_1 \cup S_2)$
2. Replace $S_1 \cup S_2$ by $H \setminus (H \setminus (S_1 \cup S_2))$

isl way:

1. Classify constraints
   - redundant: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over remaining constraints
   - valid: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over S_2
   - separating: $\max \langle \mathbf{a}_i, \mathbf{x} \rangle < c_i$ over S_2
     special cases:
     - ★ adjacent to equality: $\langle \mathbf{a}_i, \mathbf{x} \rangle = c_i - 1$ over S_2
     - ★ adjacent to inequality: $\left\langle (\mathbf{a}_i + \mathbf{b}_j), \mathbf{x} \right\rangle = (c_i + d_j) - 1$ over S_2
   - cut: otherwise

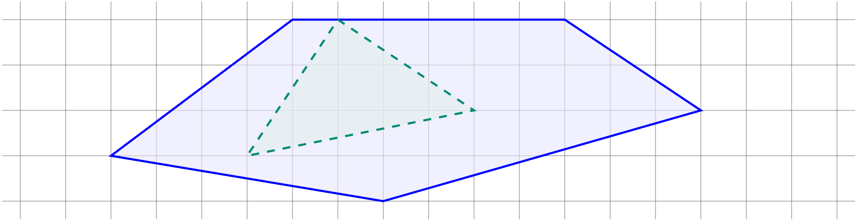
isl Operation: Set Coalescing

1. Classify constraints
   - redundant: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over remaining constraints
   - valid: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over S_2
   - separating: $\max \langle \mathbf{a}_i, \mathbf{x} \rangle < c_i$ over S_2
     special cases:
     - adjacent to equality: $\langle \mathbf{a}_i, \mathbf{x} \rangle = c_i - 1$ over S_2
     - adjacent to inequality: $\left\langle (\mathbf{a}_i + \mathbf{b}_j), \mathbf{x} \right\rangle = (c_i + d_j) - 1$ over S_2
   - cut: otherwise

isl Operation: Set Coalescing

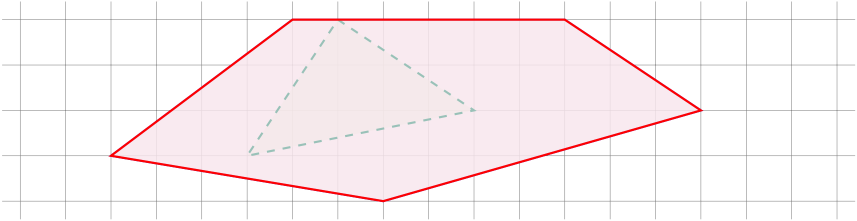
1. Classify constraints
   - redundant: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over remaining constraints
   - valid: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over S_2
   - separating: $\max \langle \mathbf{a}_i, \mathbf{x} \rangle < c_i$ over S_2
     special cases:
     - adjacent to equality: $\langle \mathbf{a}_i, \mathbf{x} \rangle = c_i - 1$ over S_2
     - adjacent to inequality: $\left\langle (\mathbf{a}_i + \mathbf{b}_j), \mathbf{x} \right\rangle = (c_i + d_j) - 1$ over S_2
   - cut: otherwise
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
      $\Rightarrow S_2$ can be dropped

isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
      $\Rightarrow S_2$ can be dropped

isl Operation: Set Coalescing

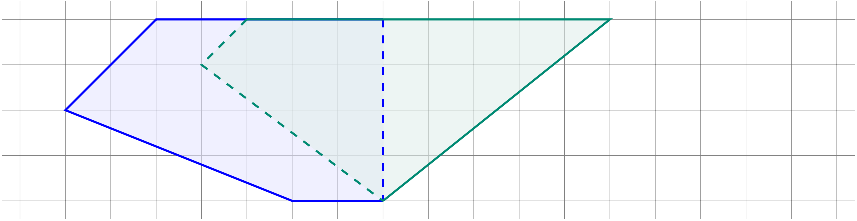


2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
      $\Rightarrow S_2$ can be dropped

isl Operation: Set Coalescing

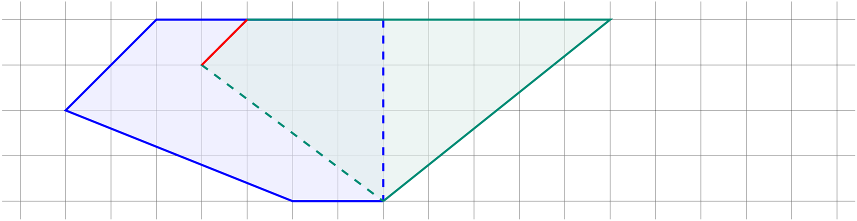
1. Classify constraints
   - redundant: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over remaining constraints
   - valid: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over S_2
   - separating: $\max \langle \mathbf{a}_i, \mathbf{x} \rangle < c_i$ over S_2
     special cases:
     - adjacent to equality: $\langle \mathbf{a}_i, \mathbf{x} \rangle = c_i - 1$ over S_2
     - adjacent to inequality: $\left\langle (\mathbf{a}_i + \mathbf{b}_j), \mathbf{x} \right\rangle = (c_i + d_j) - 1$ over S_2
   - cut: otherwise
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
      $\Rightarrow$ replace S_1 and S_2 by basic set with all valid constraints

isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
      $\Rightarrow$ replace S_1 and S_2 by basic set with all valid constraints

isl Operation: Set Coalescing

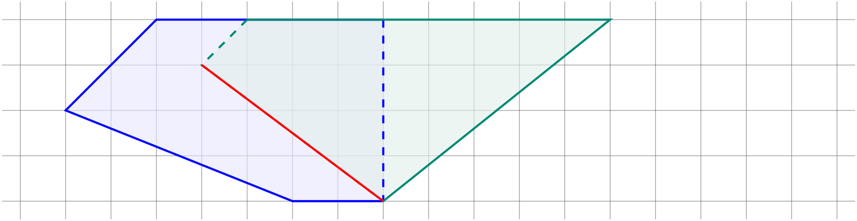


2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
      $\Rightarrow$ replace S_1 and S_2 by basic set with all valid constraints

isl Operation: Set Coalescing

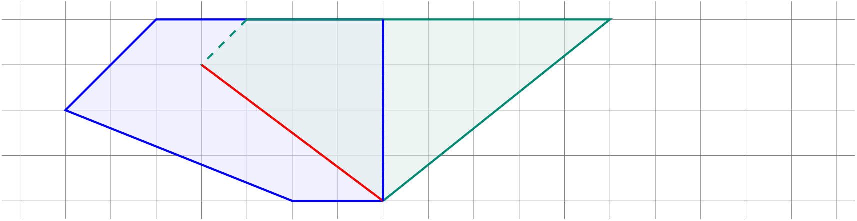
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
      $\Rightarrow$ replace S_1 and S_2 by basic set with all valid constraints

isl Operation: Set Coalescing



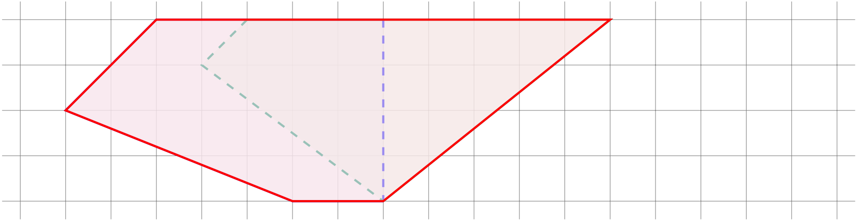
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
      $\Rightarrow$ replace S_1 and S_2 by basic set with all valid constraints

isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
      $\Rightarrow$ replace S_1 and S_2 by basic set with all valid constraints

isl Operation: Set Coalescing

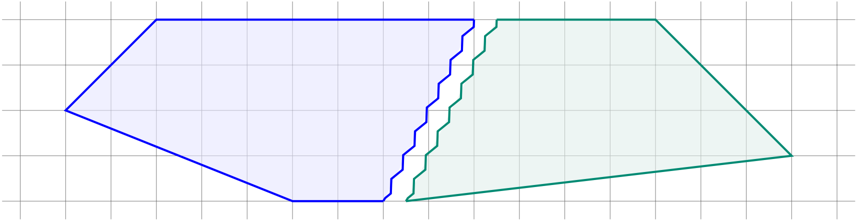


2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
      $\Rightarrow$ replace S_1 and S_2 by basic set with all valid constraints

`isl` Operation: Set Coalescing

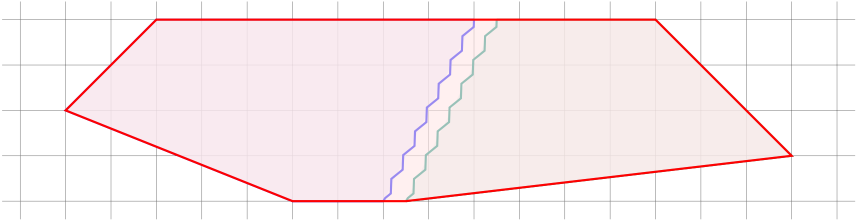
1. Classify constraints
   - redundant: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over remaining constraints
   - valid: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over S_2
   - separating: $\max \langle \mathbf{a}_i, \mathbf{x} \rangle < c_i$ over S_2
     special cases:
     - adjacent to equality: $\langle \mathbf{a}_i, \mathbf{x} \rangle = c_i - 1$ over S_2
     - adjacent to inequality: $\left\langle (\mathbf{a}_i + \mathbf{b}_j), \mathbf{x} \right\rangle = (c_i + d_j) - 1$ over S_2
   - cut: otherwise
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
      $\Rightarrow$ replace S_1 and S_2 by basic set with all valid constraints

isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
      $\Rightarrow$ replace S_1 and S_2 by basic set with all valid constraints

isl Operation: Set Coalescing

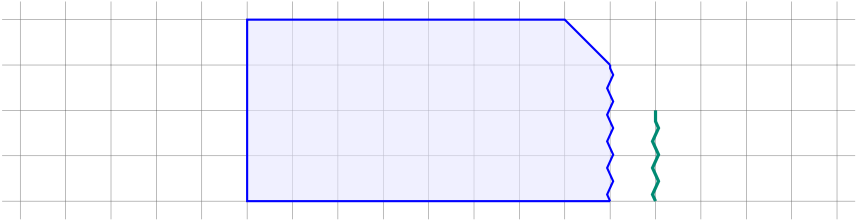


2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
      $\Rightarrow$ replace S_1 and S_2 by basic set with all valid constraints

isl Operation: Set Coalescing

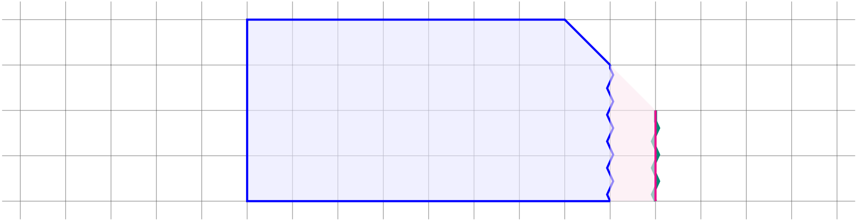
1. Classify constraints
   - redundant: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over remaining constraints
   - valid: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over S_2
   - separating: $\max \langle \mathbf{a}_i, \mathbf{x} \rangle < c_i$ over S_2
     special cases:
     - adjacent to equality: $\langle \mathbf{a}_i, \mathbf{x} \rangle = c_i - 1$ over S_2
     - adjacent to inequality: $\left\langle (\mathbf{a}_i + \mathbf{b}_j), \mathbf{x} \right\rangle = (c_i + d_j) - 1$ over S_2
   - cut: otherwise
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2)
      + other constraints of S_1 are valid
      + constraints of S_2 valid for facet of relaxed inequality
      $\Rightarrow$ drop S_2 and relax adjacent inequality of S_1

isl Operation: Set Coalescing



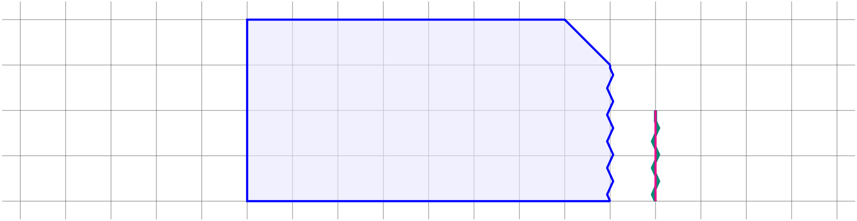
2. Case distinction
    1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
    2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
    3. single pair of adjacent inequalities (other constraints valid)
    4. single adjacent pair of an inequality (S_1) and an equality (S_2)
       + other constraints of S_1 are valid
       + constraints of S_2 valid for facet of relaxed inequality
       $\Rightarrow$ drop S_2 and relax adjacent inequality of S_1

isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2)
      + other constraints of S_1 are valid
      + constraints of S_2 valid for facet of relaxed inequality
      $\Rightarrow$ drop S_2 and relax adjacent inequality of S_1

isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2)
      + other constraints of S_1 are valid
      + constraints of S_2 valid for facet of relaxed inequality
      $\Rightarrow$ drop S_2 and relax adjacent inequality of S_1

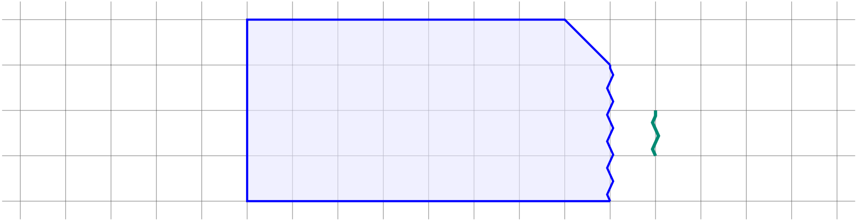
isl Operation: Set Coalescing

2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2)
      + other constraints of S_1 are valid
      + constraints of S_2 valid for facet of relaxed inequality
      $\Rightarrow$ drop S_2 and relax adjacent inequality of S_1

isl Operation: Set Coalescing

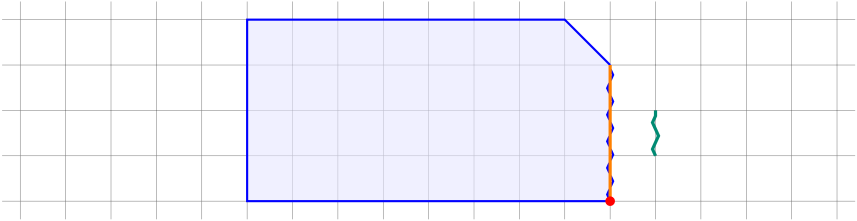
1. Classify constraints
   - redundant: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over remaining constraints
   - valid: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over S_2
   - separating: $\max \langle \mathbf{a}_i, \mathbf{x} \rangle < c_i$ over S_2
     special cases:
     - ★ adjacent to equality: $\langle \mathbf{a}_i, \mathbf{x} \rangle = c_i - 1$ over S_2
     - ★ adjacent to inequality: $\left\langle (\mathbf{a}_i + \mathbf{b}_j), \mathbf{x} \right\rangle = (c_i + d_j) - 1$ over S_2
   - cut: otherwise
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2)
      + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2)
      + other constraints of S_1 are valid
      + inequality and equality can be wrapped to include union
      $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



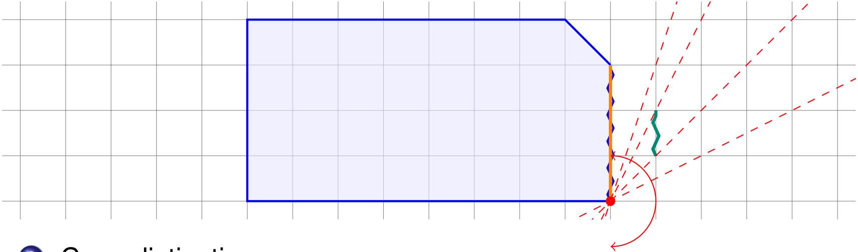
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + other constraints of S_1 are valid + inequality and equality can be wrapped to include union $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



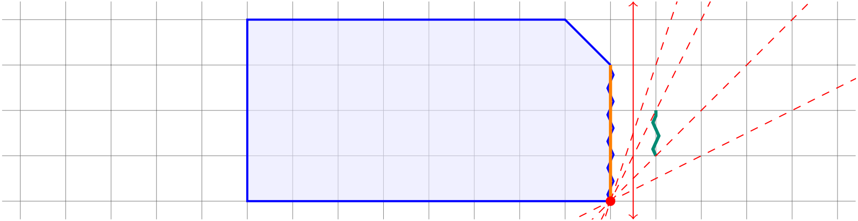
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + other constraints of S_1 are valid + inequality and equality can be wrapped to include union $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



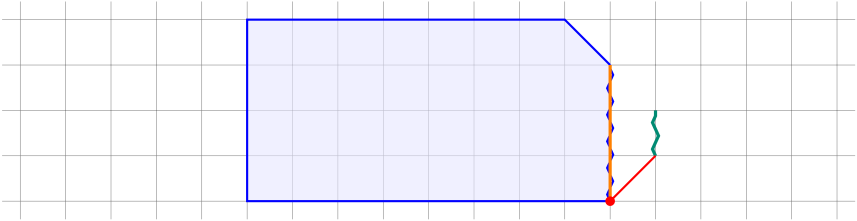
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2)
      + other constraints of S_1 are valid
      + inequality and equality can be wrapped to include union
      $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



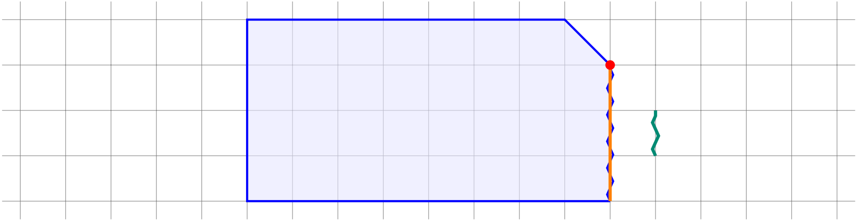
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + other constraints of S_1 are valid + inequality and equality can be wrapped to include union $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



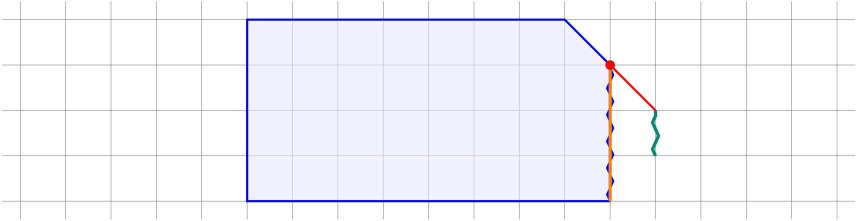
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2)
      + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2)
      + other constraints of S_1 are valid
      + inequality and equality can be wrapped to include union
      $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



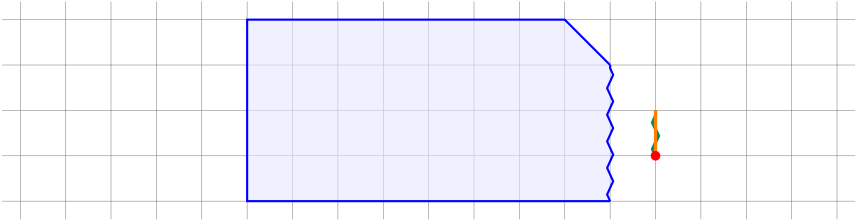
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2)
      + other constraints of S_1 are valid
      + inequality and equality can be wrapped to include union
      $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



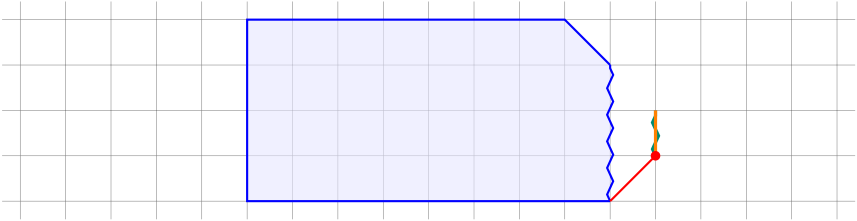
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2)
      + other constraints of S_1 are valid
      + inequality and equality can be wrapped to include union
      $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



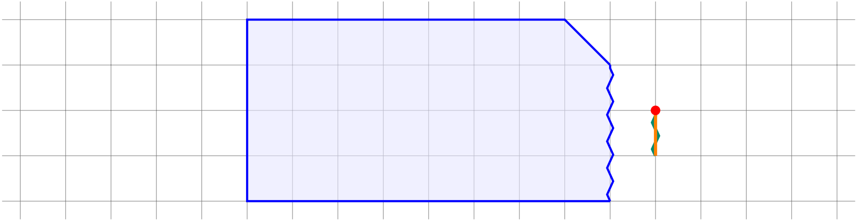
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2)
      + other constraints of S_1 are valid
      + inequality and equality can be wrapped to include union
      $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



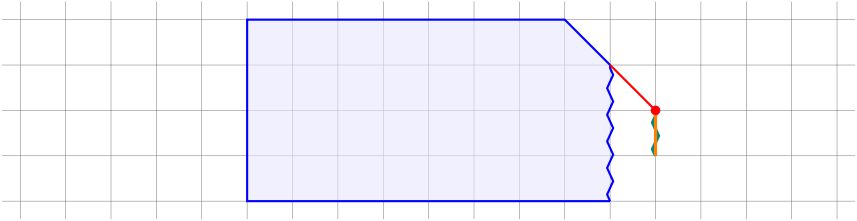
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + other constraints of S_1 are valid + inequality and equality can be wrapped to include union $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + other constraints of S_1 are valid + inequality and equality can be wrapped to include union $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2)
      + other constraints of S_1 are valid
      + inequality and equality can be wrapped to include union
      $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

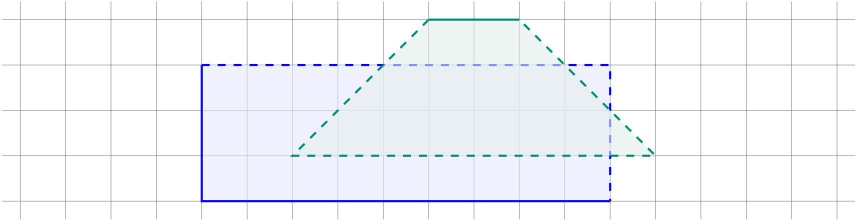
isl Operation: Set Coalescing

2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2)
      + other constraints of S_1 are valid
      + inequality and equality can be wrapped to include union
      $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing

1. Classify constraints
   - redundant: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over remaining constraints
   - valid: $\min \langle \mathbf{a}_i, \mathbf{x} \rangle > c_i - 1$ over S_2
   - separating: $\max \langle \mathbf{a}_i, \mathbf{x} \rangle < c_i$ over S_2
     special cases:
     - ★ adjacent to equality: $\langle \mathbf{a}_i, \mathbf{x} \rangle = c_i - 1$ over S_2
     - ★ adjacent to inequality: $\left\langle (\mathbf{a}_i + \mathbf{b}_j), \mathbf{x} \right\rangle = (c_i + d_j) - 1$ over S_2
   - cut: otherwise
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union
      $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

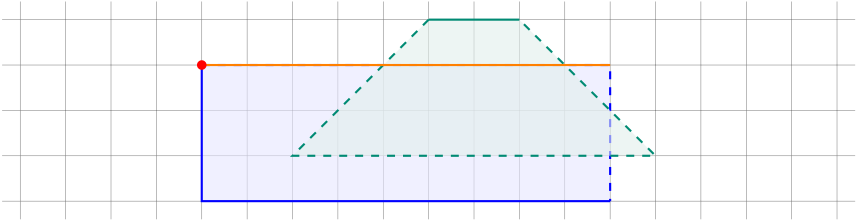
isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union

   $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

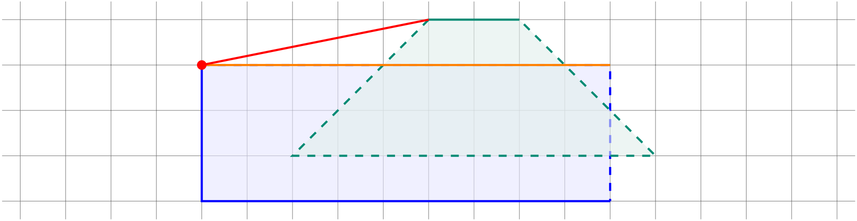
isl Operation: Set Coalescing



2. Case distinction
    1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
    2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
    3. single pair of adjacent inequalities (other constraints valid)
    4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
    5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
    6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union

    $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

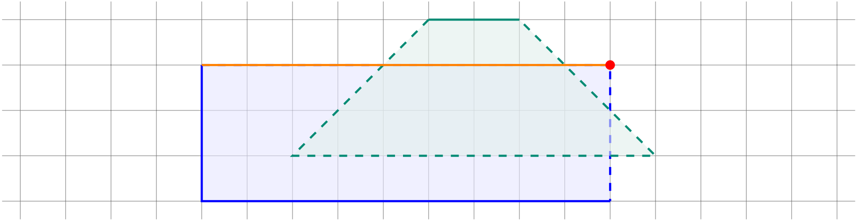
isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union

   $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

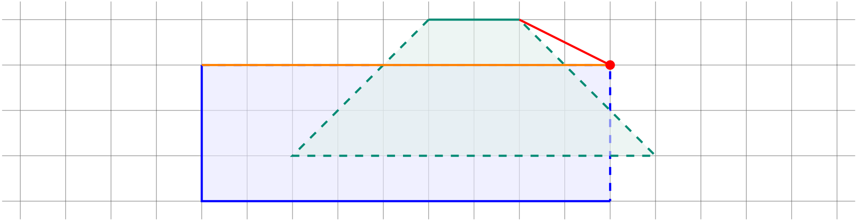
isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union

   $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

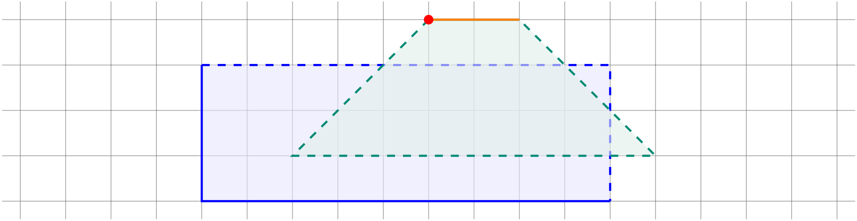
isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union

   ⇒ replace S_1 and S_2 by valid and wrapping constraints

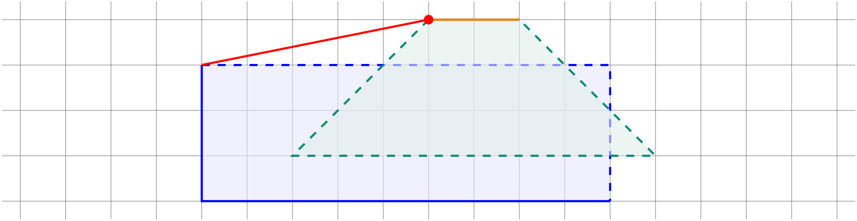
`isl` Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union

   $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

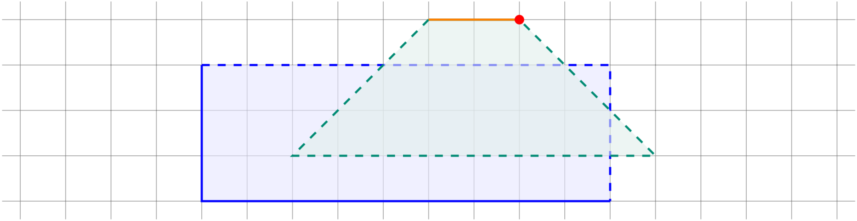
isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union

   $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

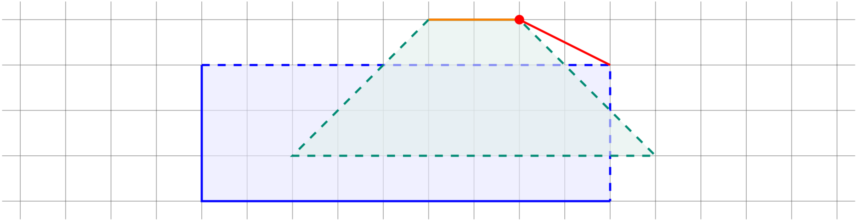
isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union

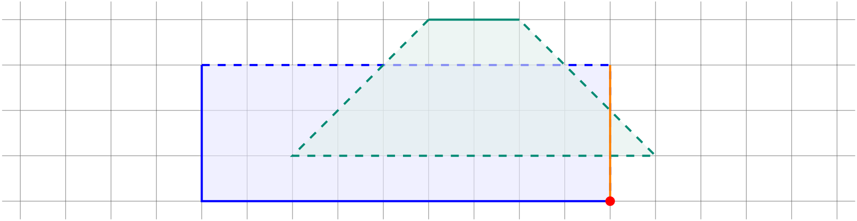
   $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union
      $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

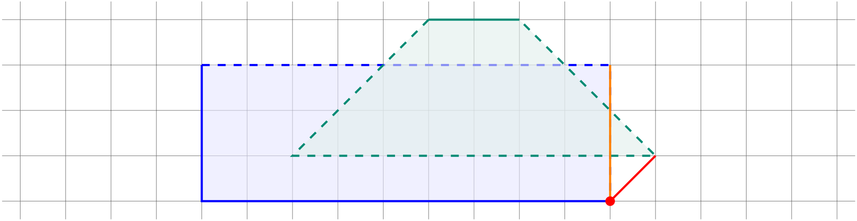
isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union

   $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

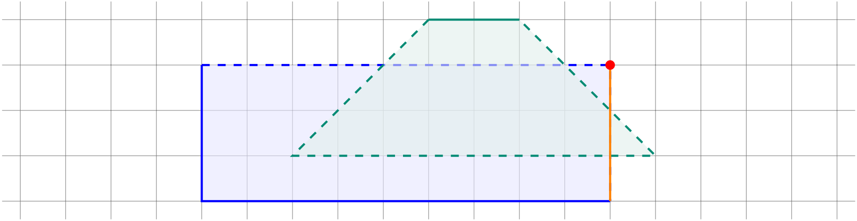
isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union

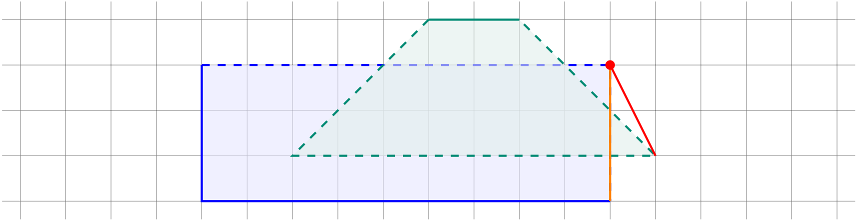
   $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



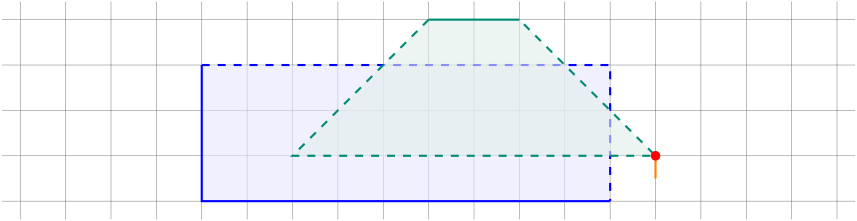
2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union
      $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union
      $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

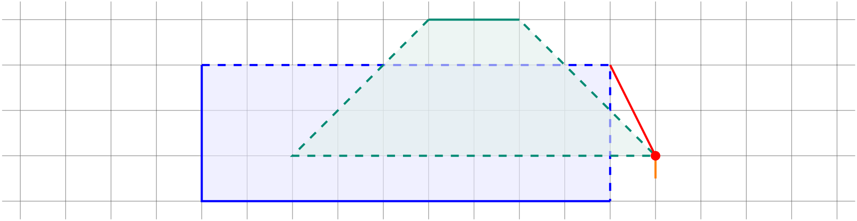
isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union

   $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

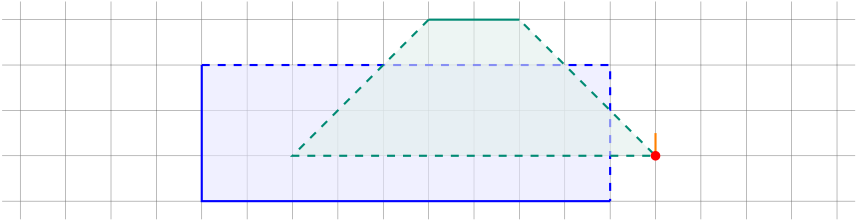
isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union

   $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

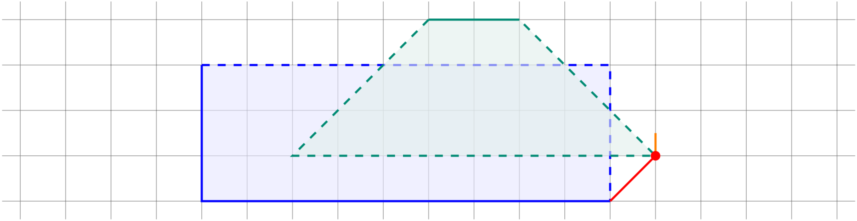
isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union

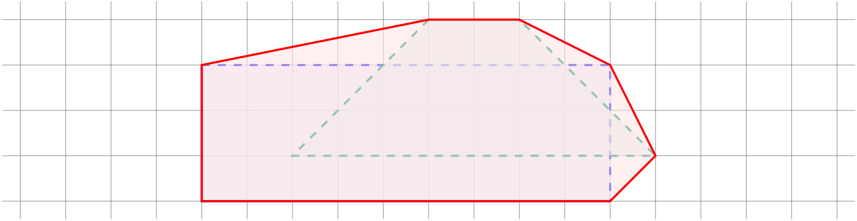
   $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

isl Operation: Set Coalescing



2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union
      $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

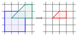
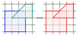
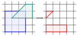
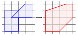
isl Operation: Set Coalescing

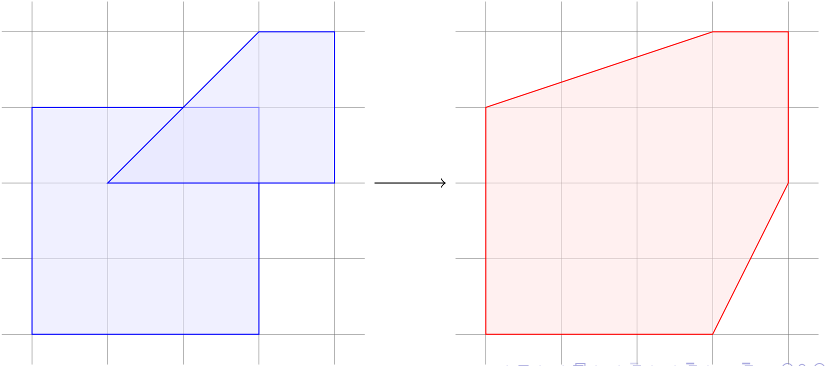


2. Case distinction
   1. non-redundant constraints of S_1 are valid for S_2, i.e., $S_2 \subseteq S_1$
   2. no separating constraints and cut constraints of S_2 are valid for cut facets of S_1 (similar to BFT2001)
   3. single pair of adjacent inequalities (other constraints valid)
   4. single adjacent pair of an inequality (S_1) and an equality (S_2) + constraints of S_2 valid for facet of relaxed inequality
   5. single adjacent pair of an inequality (S_1) and an equality (S_2) + inequality and equality can be wrapped to include union
   6. S_2 extends beyond S_1 by at most one and all cut constraints of S_1 and parallel slices of S_2 can be wrapped to include union

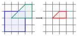
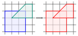
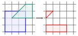
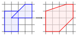
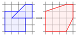
   $\Rightarrow$ replace S_1 and S_2 by valid and wrapping constraints

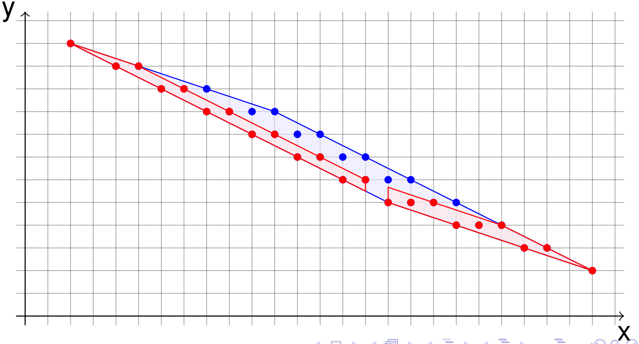
Supported Operations

- Intersection 
- Union 
- Set difference 
- Closed convex hull (“wrapping”, FLL2000) 
- Coalescing

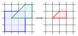
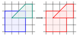
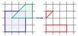
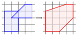
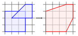
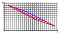


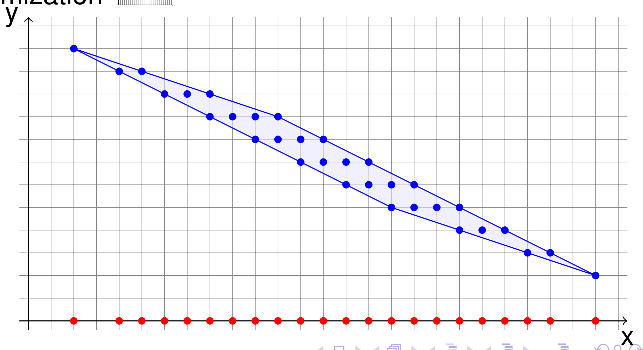
Supported Operations

- Intersection 
- Union 
- Set difference 
- Closed convex hull (“wrapping”, FLL2000) 
- Coalescing 
- Lexicographic minimization

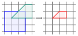
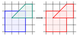
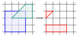
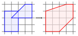
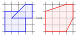
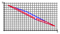


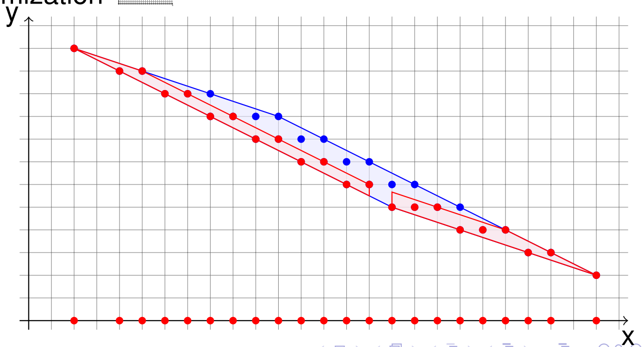
Supported Operations

- Intersection 
- Union 
- Set difference 
- Closed convex hull (“wrapping”, FLL2000) 
- Coalescing 
- Lexicographic minimization 
- Integer projection

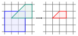
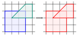
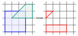
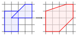
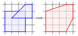
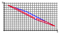
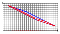


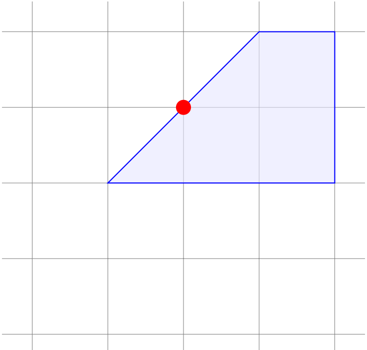
Supported Operations

- Intersection 
- Union 
- Set difference 
- Closed convex hull (“wrapping”, FLL2000) 
- Coalescing 
- Lexicographic minimization 
- Integer projection

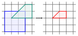
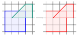
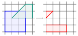
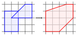
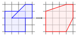
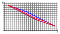
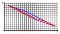


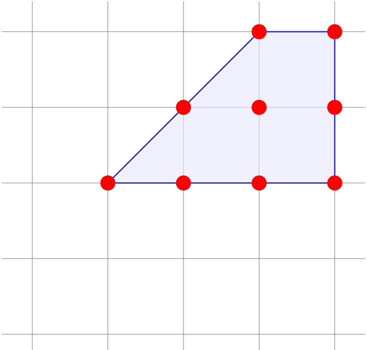
Supported Operations

- Intersection 
- Union 
- Set difference 
- Closed convex hull (“wrapping”, FLL2000) 
- Coalescing 
- Lexicographic minimization 
- Integer projection 
- Sampling (GBR)

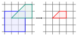
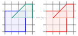
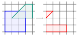
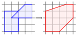
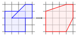
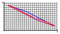
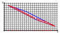


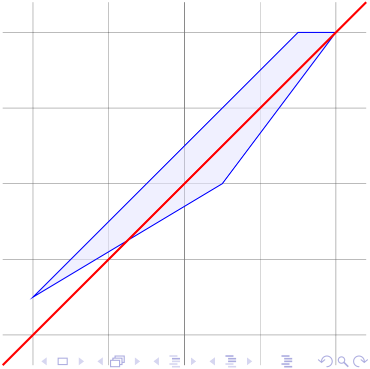
Supported Operations

- Intersection 
- Union 
- Set difference 
- Closed convex hull (“wrapping”, FLL2000) 
- Coalescing 
- Lexicographic minimization 
- Integer projection 
- Sampling (GBR) 
- Scanning (GBR)

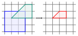
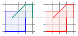
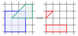
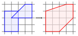
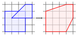
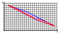
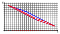


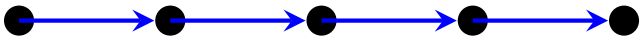
Supported Operations

- Intersection 
- Union 
- Set difference 
- Closed convex hull (“wrapping”, FLL2000) 
- Coalescing 
- Lexicographic minimization 
- Integer projection 
- Sampling (GBR) 
- Scanning (GBR) 
- Integer affine hull (GBR)

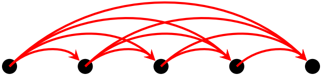


Supported Operations

- Intersection 
- Union 
- Set difference 
- Closed convex hull (“wrapping”, FLL2000) 
- Coalescing 
- Lexicographic minimization 
- Integer projection 
- Sampling (GBR) 
- Scanning (GBR) 
- Integer affine hull (GBR) 
- Transitive closure (approx.)

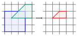
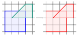
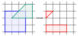
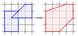
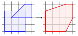
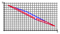
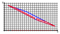


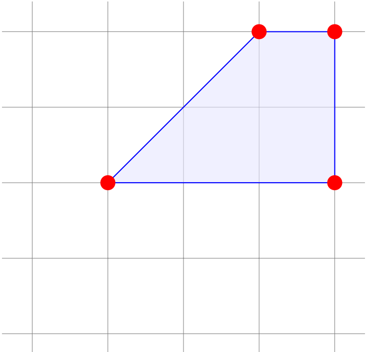
$\{ x \to x + 1 \mid 0 \leq x \leq 3 \}$



$\{ x \to y \mid 0 \leq x < y \leq 4 \}$

Supported Operations

- Intersection 
- Union 
- Set difference 
- Closed convex hull (“wrapping”, FLL2000) 
- Coalescing 
- Lexicographic minimization 
- Integer projection 
- Sampling (GBR) 
- Scanning (GBR) 
- Integer affine hull (GBR) 
- Transitive closure (approx.) 
- Parametric vertex enumeration



$\mathcal{H}$-Parametric Polytopes and their Vertices

Polytopes described by hyperplanes that depend linearly on parameters

$$P(\mathbf{s}) = \{ \mathbf{x} \in \mathbb{Q}^d \mid A\mathbf{x} + B\mathbf{s} \geq \mathbf{c} \}$$

Example:

$$P(N) = \{ (i, j) \mid i \geq 1 \wedge i \leq N \wedge j \geq 1 \wedge j \leq i \}$$

Parametric vertices:

$$P = \text{conv.hull} \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} N \\ 1 \end{bmatrix}, \begin{bmatrix} N \\ N \end{bmatrix} \right\}$$

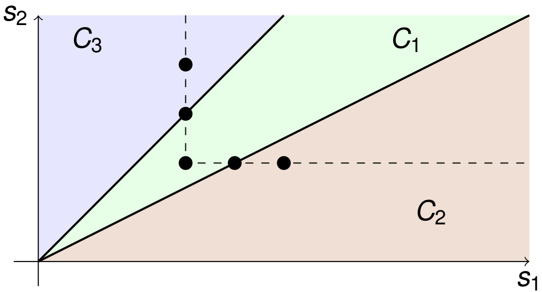
In general: different (active) vertices on different parts of the parameter space (chamber decomposition)

Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$

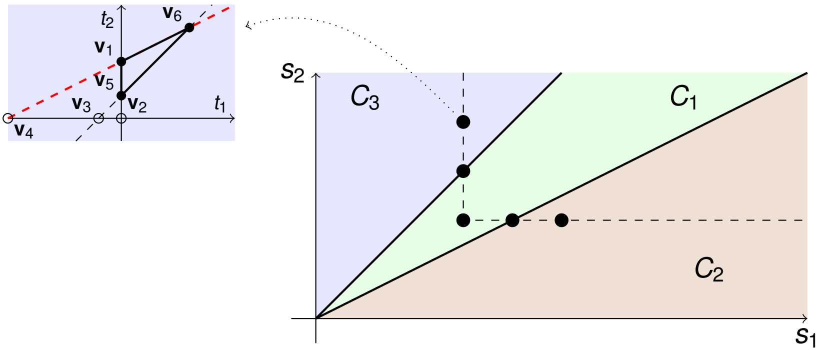
Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$



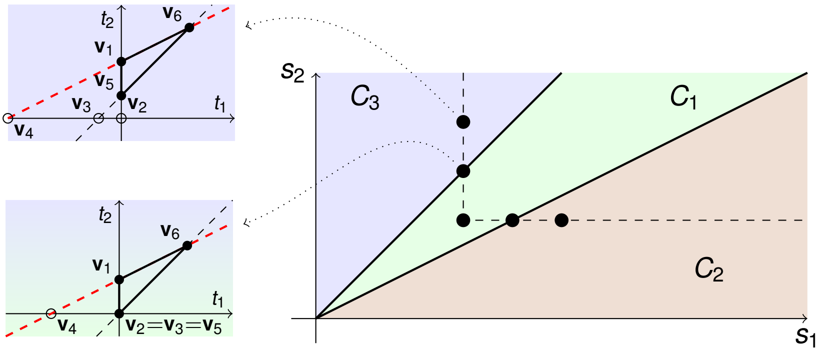
Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$



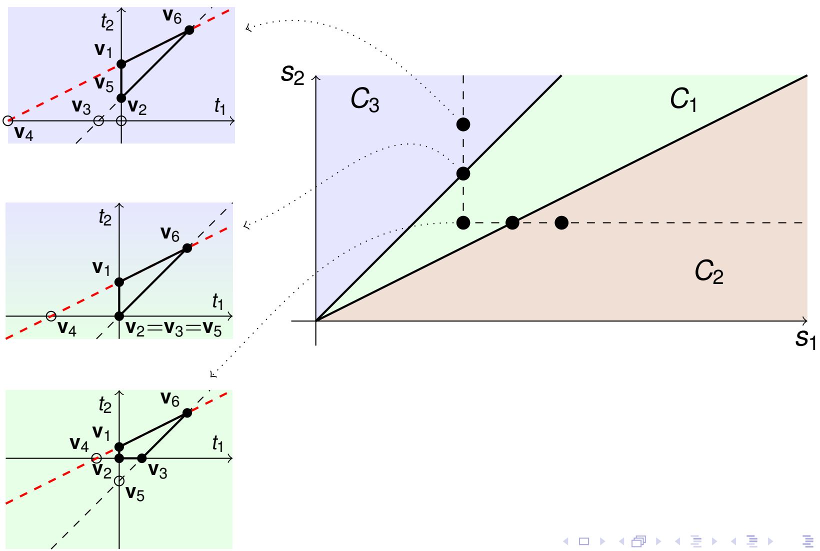
Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$



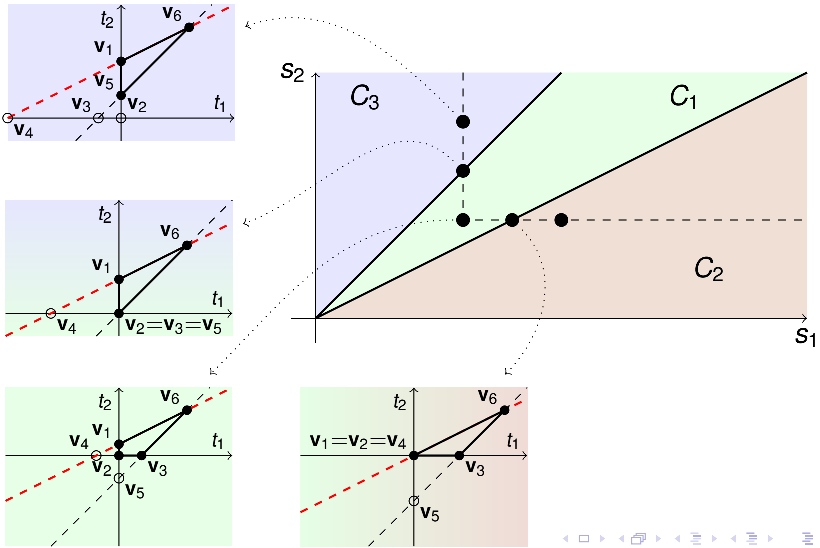
Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$



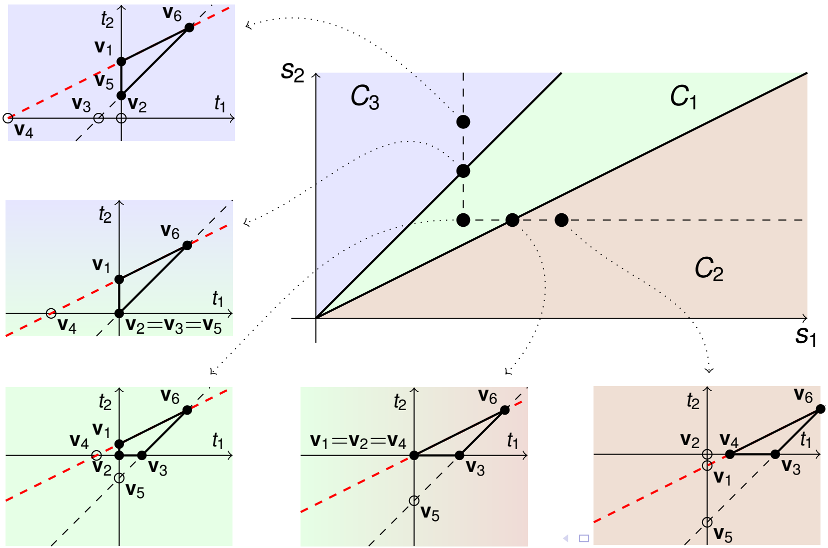
Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$



Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$



Parametric Vertex Enumeration

- Vertex computation
  - Consider all combinations of d inequalities
  - Turn them into equalities
  - Record vertex and activity domain if non-empty

Parametric Vertex Enumeration

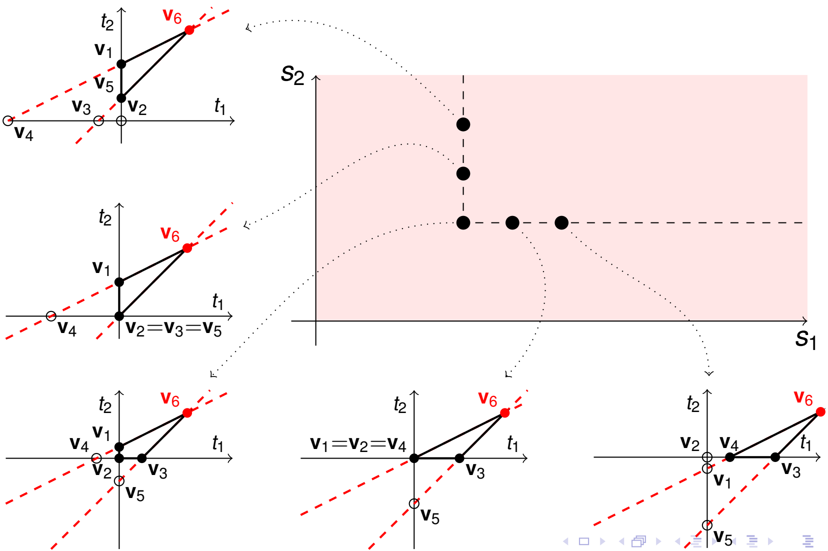
- Vertex computation
  - Consider all combinations of d inequalities
    - $\Rightarrow$ using backtracking and incremental LP solver
  - Turn them into equalities
  - Record vertex and activity domain if non-empty

Parametric Vertex Enumeration

- Vertex computation
  - Consider all combinations of d inequalities
    - ⇒ using backtracking and incremental LP solver
  - Turn them into equalities
  - Record vertex and activity domain if non-empty
    - ⇒ only record for lexmin inequalities

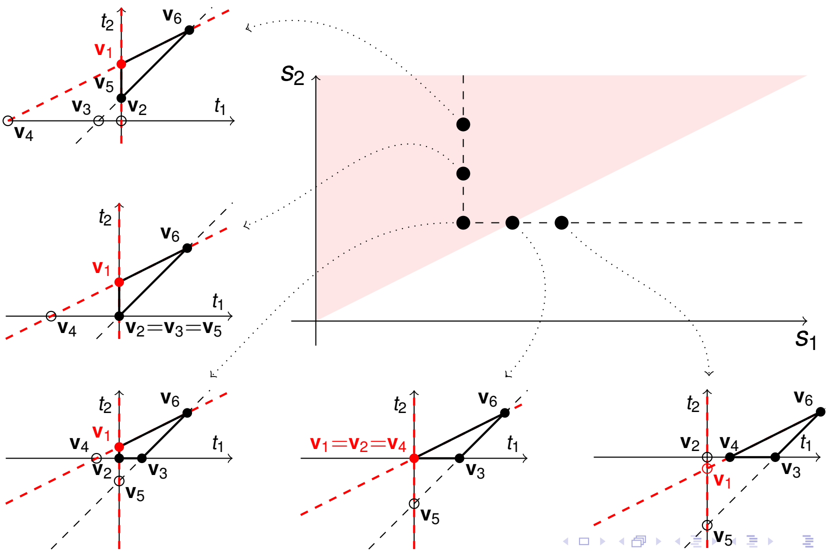
Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$



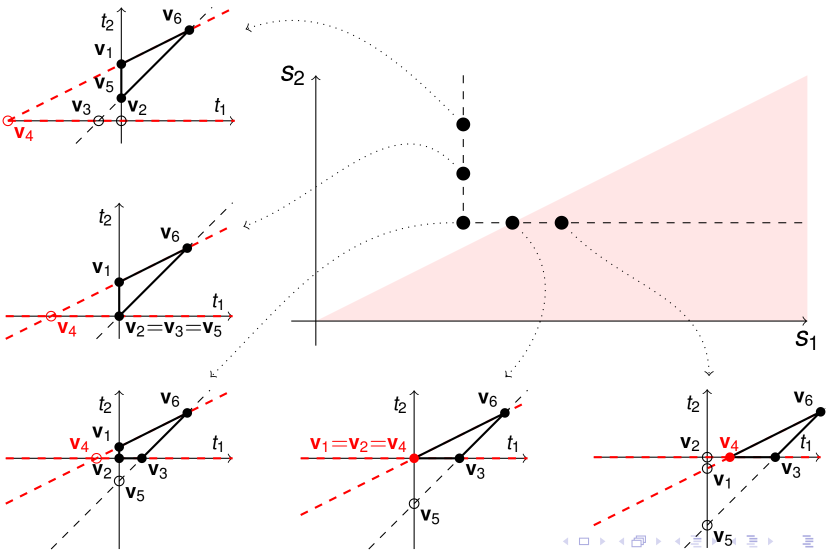
Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$



Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$



Parametric Vertex Enumeration

- Vertex computation
  - Consider all combinations of d inequalities
    - $\Rightarrow$ using backtracking and incremental LP solver
  - Turn them into equalities
  - Record vertex and activity domain if non-empty
    - $\Rightarrow$ only record for lexmin inequalities

Parametric Vertex Enumeration

- Vertex computation
  - Consider all combinations of d inequalities
    - $\Rightarrow$ using backtracking and incremental LP solver
  - Turn them into equalities
  - Record vertex and activity domain if non-empty
    - $\Rightarrow$ only record for lexmin inequalities
- Chamber decomposition (note: only full-dimensional chambers)
  PolyLib:
  - iterate over all activity domains
  - compute differences and intersections with previous activity domains

Parametric Vertex Enumeration

- Vertex computation
  - Consider all combinations of d inequalities

    $\Rightarrow$ using backtracking and incremental LP solver
  - Turn them into equalities
  - Record vertex and activity domain if non-empty

    $\Rightarrow$ only record for lexmin inequalities
- Chamber decomposition (note: only full-dimensional chambers)

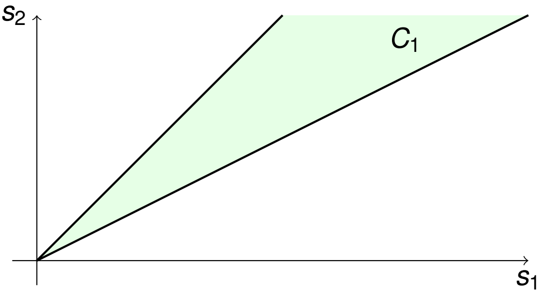
  PolyLib:
  - iterate over all activity domains
  - compute differences and intersections with previous activity domains

  isl:
  - compute initial chamber (intersection of activity domains)
  - pick unhandled internal facet
  - intersect activity domains that contain facet and other side

    $\Rightarrow$ new chamber
  - repeat while there are unhandled internal facets

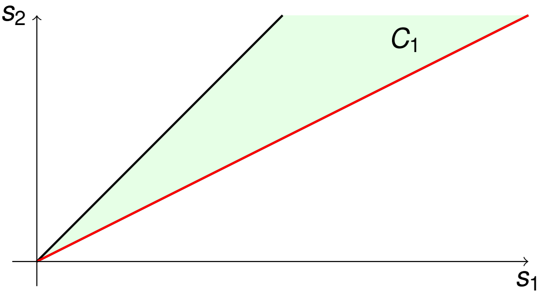
Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$



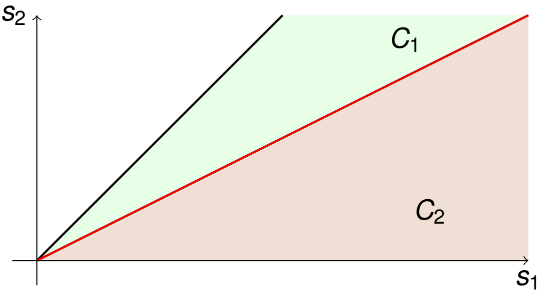
Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$



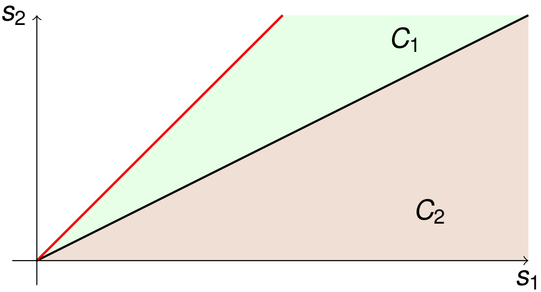
Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$



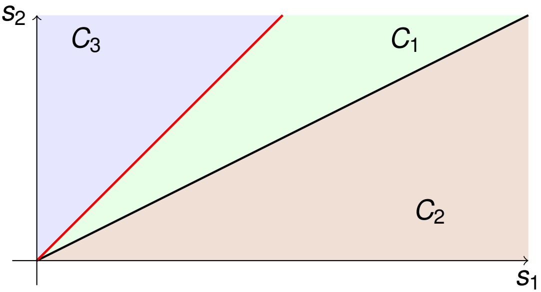
Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$



Chamber Decomposition

$$\left\{ \mathbf{t} \in \mathbb{Q}^2 \mid -s_1 + 2s_2 + t_1 - 2t_2 \geq 0 \wedge s_1 - s_2 - t_1 + t_2 \geq 0 \wedge t_1 \geq 0 \wedge t_2 \geq 0 \right\}$$



Parametric Vertex Enumeration

- Vertex computation
  - Consider all combinations of d inequalities
    $\Rightarrow$ using backtracking and incremental LP solver
  - Turn them into equalities
  - Record vertex and activity domain if non-empty
    $\Rightarrow$ only record for lexmin inequalities
- Chamber decomposition (note: only full-dimensional chambers)

  PolyLib:
  - iterate over all activity domains
  - compute differences and intersections with previous activity domains

  isl:
  - compute initial chamber (intersection of activity domains)
  - pick unhandled internal facet
  - intersect activity domains that contain facet and other side
    $\Rightarrow$ new chamber
  - repeat while there are unhandled internal facets

Parametric Vertex Enumeration

- Vertex computation
  - Consider all combinations of d inequalities
    $\Rightarrow$ using backtracking and incremental LP solver
  - Turn them into equalities
  - Record vertex and activity domain if non-empty
    $\Rightarrow$ only record for lexmin inequalities
- Chamber decomposition (note: only full-dimensional chambers)
  PolyLib:
  - iterate over all activity domains
  - compute differences and intersections with previous activity domains

  isl:
  - compute initial chamber (intersection of activity domains)
  - pick unhandled internal facet
  - intersect activity domains that contain facet and other side
    $\Rightarrow$ new chamber
  - repeat while there are unhandled internal facets

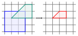
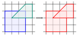
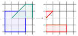
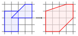
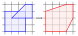
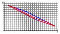
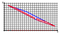
Parametric Vertex Enumeration

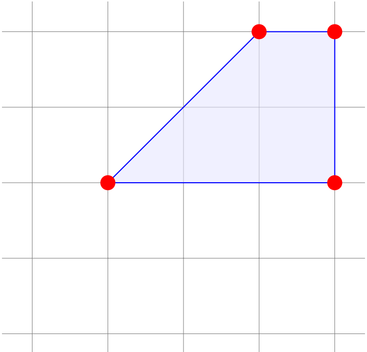
- Vertex computation
  - Consider all combinations of d inequalities
    $\Rightarrow$ using backtracking and incremental LP solver
  - Turn them into equalities
  - Record vertex and activity domain if non-empty
    $\Rightarrow$ only record for lexmin inequalities
- Chamber decomposition (note: only full-dimensional chambers)
  PolyLib:
  - iterate over all activity domains
  - compute differences and intersections with previous activity domains

  isl:
  - compute initial chamber (intersection of activity domains)
  - pick unhandled internal facet
  - intersect activity domains that contain facet and other side
    $\Rightarrow$ new chamber
  - repeat while there are unhandled internal facets

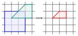
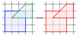
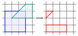
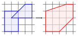
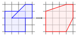
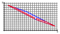
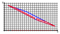
  $\Rightarrow$ much faster than PolyLib; similar to TOPCOM 0.16.2

Supported Operations

- Intersection 
- Union 
- Set difference 
- Closed convex hull (“wrapping”, FLL2000) 
- Coalescing 
- Lexicographic minimization 
- Integer projection 
- Sampling (GBR) 
- Scanning (GBR) 
- Integer affine hull (GBR) 
- Transitive closure (approx.) 
- Parametric vertex enumeration



Supported Operations

- Intersection 
- Union 
- Set difference 
- Closed convex hull (“wrapping”, FLL2000) 
- Coalescing 
- Lexicographic minimization 
- Integer projection 
- Sampling (GBR) 
- Scanning (GBR) 
- Integer affine hull (GBR) 
- Transitive closure (approx.) 
- Parametric vertex enumeration 
- Bounds on quasipolynomials (approx.)

$\mathcal{V}$-Parametric Polytopes and Bounds on Polynomials

- $\mathcal{V}$-parametric polytopes

$$P : D \to \mathbb{Q}^n :$$
$$\mathbf{q} \mapsto P(\mathbf{q}) = \{\, \mathbf{x} \mid \exists \alpha_i \in \mathbb{Q} : \mathbf{x} = \textstyle\sum_i \alpha_i \mathbf{v}_i(\mathbf{q}), \alpha_i \geq 0, \sum_i \alpha_i = 1 \,\}$$

$D \subset \mathbb{Q}^r$: parameter domain
$\mathbf{v}_i(\mathbf{q}) \in \mathbb{Q}[\mathbf{q}]$ arbitrary polynomials in parameters
$\mathbf{v}_i(\mathbf{q})$ are generators of the polytope

Note: $\mathcal{V}$-parametric polytope can be computed from $\mathcal{H}$-parametric polytope through parameter vertex enumeration + chamber decomposition

$\mathcal{V}$-Parametric Polytopes and Bounds on Polynomials

- $\mathcal{V}$-parametric polytopes

$$P : D \to \mathbb{Q}^n :$$
$$\mathbf{q} \mapsto P(\mathbf{q}) = \{\, \mathbf{x} \mid \exists \alpha_i \in \mathbb{Q} : \mathbf{x} = \sum_i \alpha_i \mathbf{v}_i(\mathbf{q}), \alpha_i \geq 0, \sum_i \alpha_i = 1 \,\}$$

  $D \subset \mathbb{Q}^r$: parameter domain
  $\mathbf{v}_i(\mathbf{q}) \in \mathbb{Q}[\mathbf{q}]$ arbitrary polynomials in parameters
  $\mathbf{v}_i(\mathbf{q})$ are generators of the polytope

- Bounds on quasipolynomials (CFGV2009)

Input: Parametric polytope P and quasipolynomial $p(\mathbf{q}, \mathbf{x})$
Output: Bound $B(\mathbf{q})$ on quasipolynomial over polytope

$$B(\mathbf{q}) \geq \max_{\mathbf{x} \in P(\mathbf{q})} p(\mathbf{q}, \mathbf{x})$$

Note: $\mathcal{V}$-parametric polytope can be computed from $\mathcal{H}$-parametric polytope through parameter vertex enumeration + chamber decomposition

Bounds on Quasipolynomials: Example

$$p(x_1, x_2) = \frac{1}{2}x_1^2 + \frac{1}{2}x_1 + x_2 \qquad P = \mathrm{conv.hull}\{(0,0),(N,0),(N,N)\}$$

To compute:

$$M(N) = \max_{(x_1,x_2)\in P} p(x_1, x_2)$$

Bounds on Quasipolynomials: Example

$$p(x_1, x_2) = \frac{1}{2}x_1^2 + \frac{1}{2}x_1 + x_2 \qquad P = \text{conv.hull}\{(0,0),(N,0),(N,N)\}$$

To compute:

$$B(N) \geq M(N) = \max_{(x_1,x_2)\in P} p(x_1, x_2)$$

Bounds on Quasipolynomials: Example

$$p(x_1, x_2) = \frac{1}{2}x_1^2 + \frac{1}{2}x_1 + x_2 \qquad P = \text{conv.hull}\{(0,0), (N,0), (N,N)\}$$

To compute:

$$B(N) \geq M(N) = \max_{(x_1,x_2) \in P} p(x_1, x_2)$$

How? $\Rightarrow$ Bernstein expansion

- Express $\mathbf{x} \in P$ as convex combination of vertices

$$(x_1, x_2) = \alpha_1(0,0) + \alpha_2(N,0) + \alpha_3(N,N), \quad \alpha_i \geq 0, \quad \sum_i \alpha_i = 1$$

$$p(\alpha_1, \alpha_2, \alpha_3) = \frac{1}{2}N^2\alpha_2^2 + N^2\alpha_2\alpha_3 + \frac{1}{2}N^2\alpha_3^2 + \frac{1}{2}N\alpha_2 + \frac{3}{2}N\alpha_3$$

- Express $p(\mathbf{x})$ as convex combination of polynomials in parameters

Bounds on Quasipolynomials: Example

$$p(\alpha) = \frac{N^2}{2}\alpha_2^2 + N^2\alpha_2\alpha_3 + \frac{N^2}{2}\alpha_3^2 + \frac{N}{2}\alpha_2 + \frac{3N}{2}\alpha_3 \quad \alpha_i \geq 0, \quad \sum_i \alpha_i = 1$$

- Express $p(\mathbf{x})$ as convex combination of polynomials in parameters

$$p(\mathbf{x}) = \sum B_j(\alpha) b_j(\mathbf{N})$$

Bounds on Quasipolynomials: Example

$$p(\alpha) = \frac{N^2}{2}\alpha_2^2 + N^2\alpha_2\alpha_3 + \frac{N^2}{2}\alpha_3^2 + \frac{N}{2}\alpha_2 + \frac{3N}{2}\alpha_3 \quad \alpha_i \geq 0, \quad \sum_i \alpha_i = 1$$

- Express $p(\mathbf{x})$ as convex combination of polynomials in parameters

$$\min_j b_j(\mathbf{N}) \leq p(\mathbf{x}) = \sum B_j(\alpha) b_j(\mathbf{N}) \leq \max_j b_j(\mathbf{N})$$

Bounds on Quasipolynomials: Example

$$p(\alpha) = \frac{N^2}{2}\alpha_2^2 + N^2\alpha_2\alpha_3 + \frac{N^2}{2}\alpha_3^2 + \frac{N}{2}\alpha_2 + \frac{3N}{2}\alpha_3 \quad \alpha_i \geq 0, \quad \sum_i \alpha_i = 1$$

- Express $p(\mathbf{x})$ as convex combination of polynomials in parameters

$$\min_j b_j(\mathbf{N}) \leq p(\mathbf{x}) = \sum B_j(\alpha) b_j(\mathbf{N}) \leq \max_j b_j(\mathbf{N})$$

$$1 = (\alpha_1 + \alpha_2 + \alpha_3)^2 = \alpha_1^2 + \alpha_2^2 + \alpha_3^2 + 2\alpha_1\alpha_2 + 2\alpha_3\alpha_3 + 2\alpha_3\alpha_1$$

Bounds on Quasipolynomials: Example

$$p(\alpha) = \frac{N^2}{2}\alpha_2^2 + N^2\alpha_2\alpha_3 + \frac{N^2}{2}\alpha_3^2 + \frac{N}{2}\alpha_2 + \frac{3N}{2}\alpha_3 \quad \alpha_i \geq 0, \quad \sum_i \alpha_i = 1$$

- Express $p(\mathbf{x})$ as convex combination of polynomials in parameters

$$\min_j b_j(\mathbf{N}) \leq p(\mathbf{x}) = \sum B_j(\alpha) b_j(\mathbf{N}) \leq \max_j b_j(\mathbf{N})$$

$$1 = (\alpha_1 + \alpha_2 + \alpha_3)^2 = \alpha_1^2 + \alpha_2^2 + \alpha_3^2 + 2\alpha_1\alpha_2 + 2\alpha_3\alpha_3 + 2\alpha_3\alpha_1$$

$$p(\alpha_1, \alpha_2, \alpha_3) = \alpha_1^2 0 + \alpha_2^2\left(\frac{N^2+N}{2}\right) + \alpha_3^2\left(\frac{N^2+3N}{2}\right) + (2\alpha_1\alpha_2)\,\frac{N}{4} + (2\alpha_1\alpha_3)\,\frac{3N}{2} + (2\alpha_2\alpha_3)\,\frac{N^2+2N}{2}$$

Bounds on Quasipolynomials: Example

$$p(\alpha) = \frac{N^2}{2}\alpha_2^2 + N^2\alpha_2\alpha_3 + \frac{N^2}{2}\alpha_3^2 + \frac{N}{2}\alpha_2 + \frac{3N}{2}\alpha_3 \quad \alpha_i \geq 0, \quad \sum_i \alpha_i = 1$$

- Express $p(\mathbf{x})$ as convex combination of polynomials in parameters

$$\min_j b_j(\mathbf{N}) \leq p(\mathbf{x}) = \sum B_j(\alpha) b_j(\mathbf{N}) \leq \max_j b_j(\mathbf{N})$$

$$1 = (\alpha_1 + \alpha_2 + \alpha_3)^2 = \alpha_1^2 + \alpha_2^2 + \alpha_3^2 + 2\alpha_1\alpha_2 + 2\alpha_3\alpha_3 + 2\alpha_3\alpha_1$$

$$p(\alpha_1, \alpha_2, \alpha_3) = \alpha_1^2 0 + \alpha_2^2\left(\frac{N^2+N}{2}\right) + \alpha_3^2\left(\frac{N^2+3N}{2}\right) + (2\alpha_1\alpha_2)\frac{N}{4} + (2\alpha_1\alpha_3)\frac{3N}{2} + (2\alpha_2\alpha_3)\frac{N^2+2N}{2}$$

Outline

Conclusion

- `isl`: a relatively new integer set library
- currently used in
  - equivalence checking tool
  - `barvinok`
  - `CLooG`
- explicit support for parameters and existentially quantified variables
- all computations in exact integer arithmetic using GMP
- built-in incremental LP solver
- built-in (P)ILP solver
- released under LGPL license
- available from `http://freshmeat.net/projects/isl/`

Conclusion

- `isl`: a relatively new integer set library
- currently used in
  - equivalence checking tool
  - `barvinok`
  - `CLooG`
- explicit support for parameters and existentially quantified variables
- all computations in exact integer arithmetic using GMP
- built-in incremental LP solver
- built-in (P)ILP solver
- released under LGPL license
- available from `http://freshmeat.net/projects/isl/`

Future work: port `barvinok` to `isl`; now uses

- `PolyLib`: GPL, …
  - ⇒ `isl` already supports operations provided by PolyLib, but a lot of code still needs to be ported
- `NTL`: not thread-safe, C++
  - ⇒ `isl` needs LLL