A Short Zoology of Math Objects used in Automatic Parallelization

Christophe Alias

INRIA & LIP/ENS-Lyon
christophe.alias@ens-lyon.fr

WG Grenoble/Lyon, march 21, 2017


About Me

Researcher at Inria

- Working on compiler analysis for/of parallel architectures
- Interested by FPGA technology (for fun and profit)
- Co-founder of the XtremLogic start-up with Alexandru Plesco
  $\rightsquigarrow$ `http://www.xtremlogic.com`

This talk shows how advanced compilers leverage:

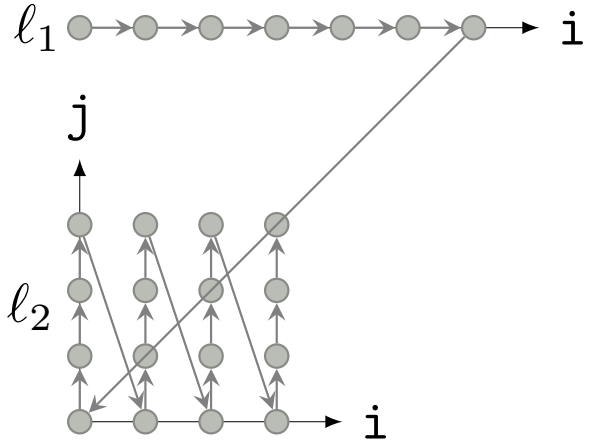
- Presburger sets
- Parametric integer programming
- Farkas lemma

Outline

1. Automatic Parallelization (polyhedral stuff)
2. Intermediate Representations (Presburger stuff)
3. Scheduling (Farkas stuff)

Polyhedral Compilation

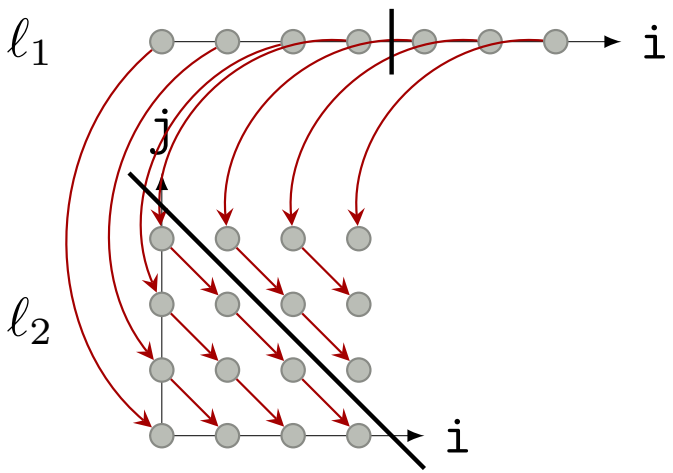
```
for i := 0 to 2N
  c[i] := 0;
for i := 0 to N
  for j := 0 to N
    c[i+j] := c[i+j] + a[i]*b[j];
```



- Execution trace computable statically
- Exact compiler analysis with polyhedral algorithms
  $\rightsquigarrow$ dependences, scheduling, resource allocation, etc

Typical Polyhedral Compiler

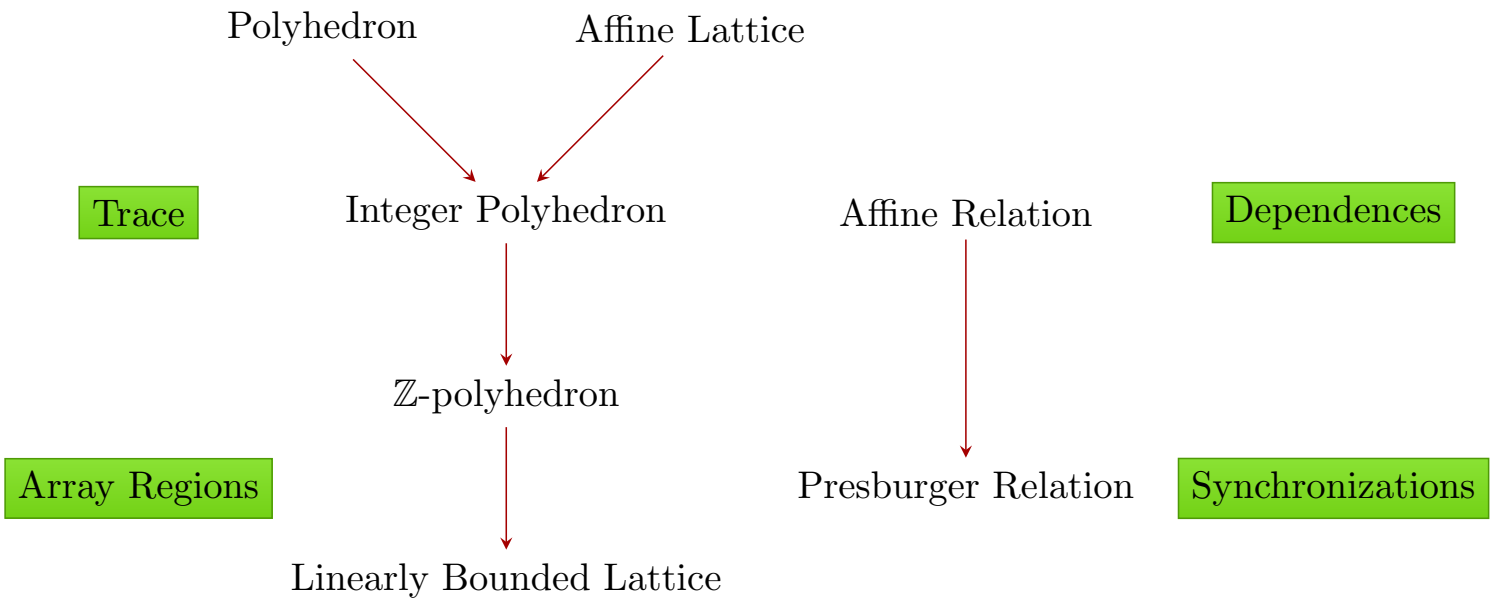
1. Polyhedral representation
2. Dependence analysis
3. Scheduling / allocation
4. Code generation



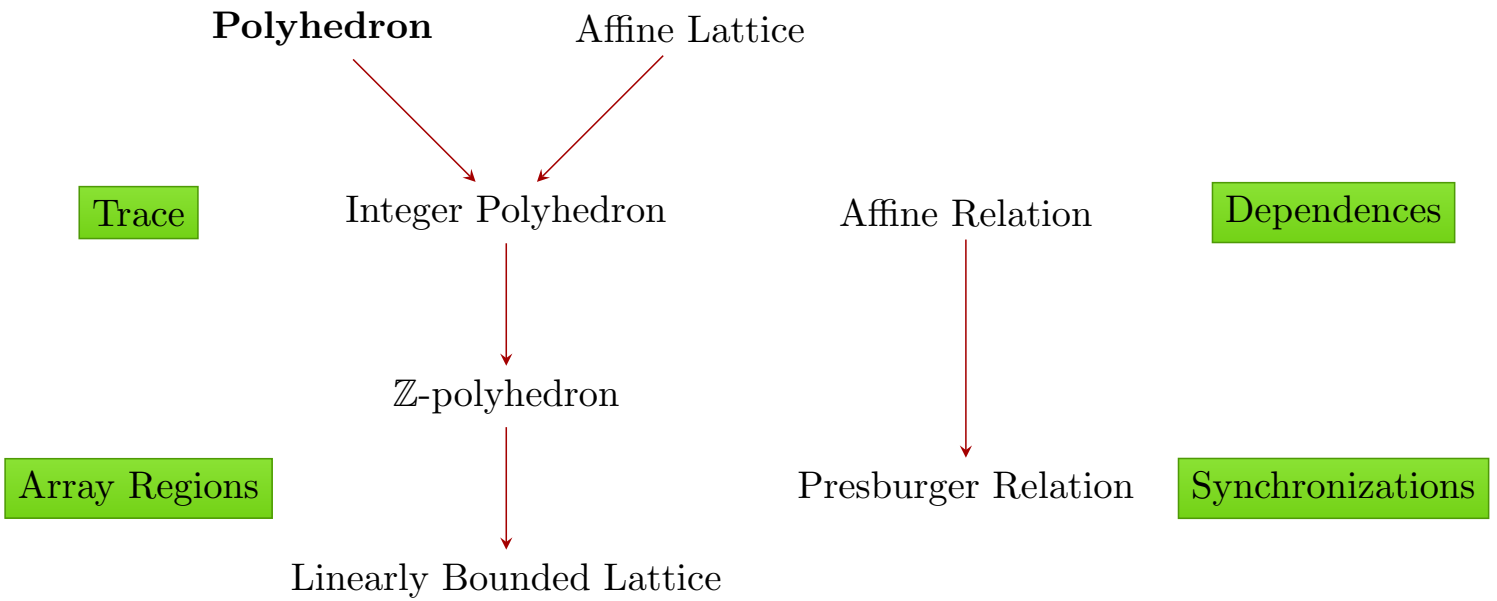
Outline

1. Automatic Parallelization (polyhedral stuff)
2. Intermediate Representations (Presburger stuff)
3. Scheduling (Farkas stuff)

Zoology


Polyhedron
Affine Lattice
Trace
Integer Polyhedron
Affine Relation
Dependences
ℤ-polyhedron
Array Regions
Presburger Relation
Synchronizations
Linearly Bounded Lattice

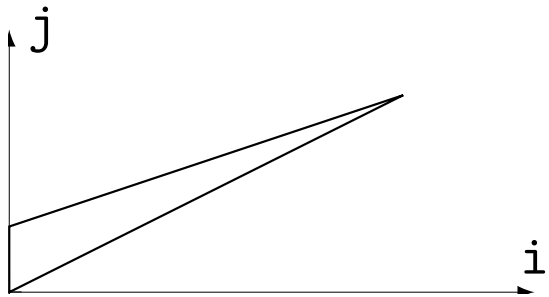
Zoology



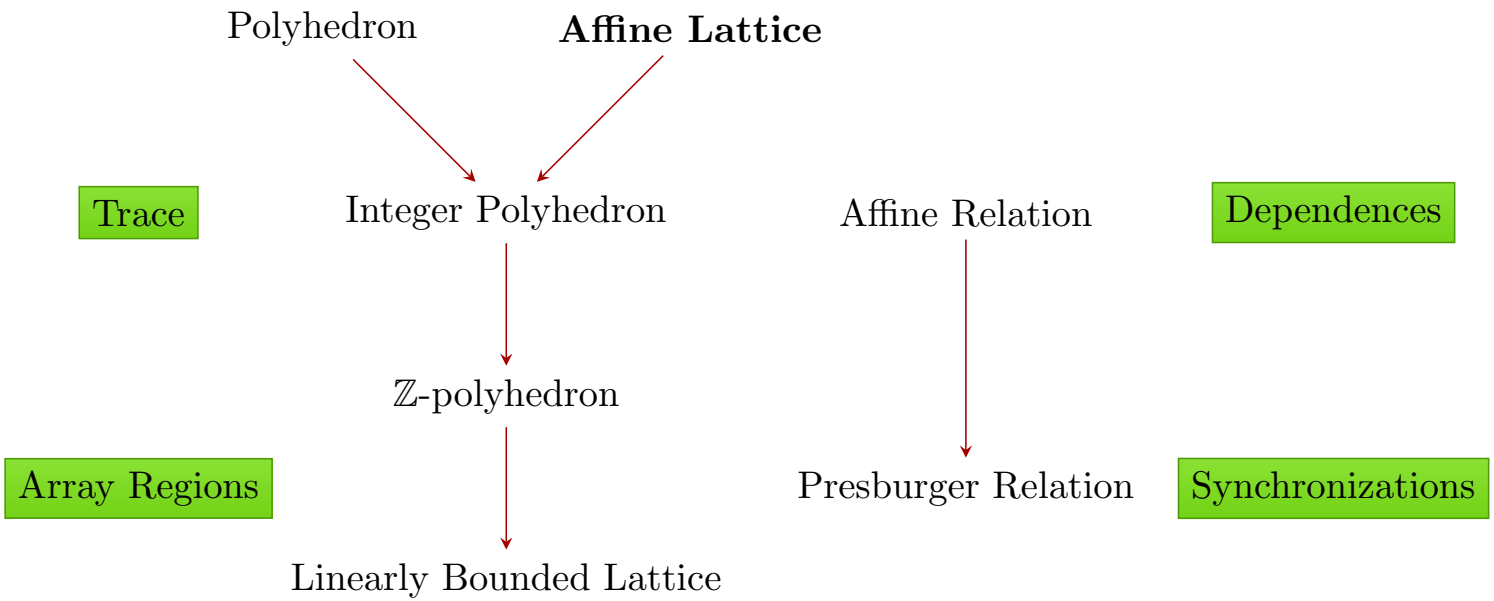
(Parametrized) Polyhedron

- $\vec{n}$ vector of parameters
- A, B integer matrices
- $\vec{c}$ integer vector

$$\mathcal{P}(\vec{n}) = \{\vec{x} \in \mathbf{R}^d,\ A\vec{x} + B\vec{n} + \vec{c} \geq 0\}$$



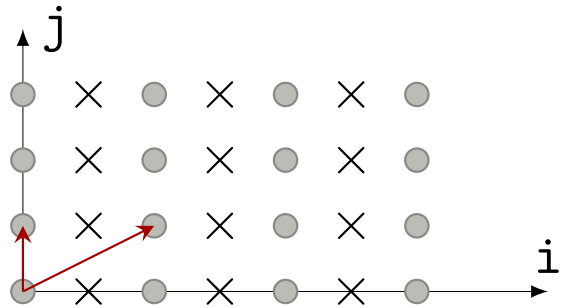
Zoology



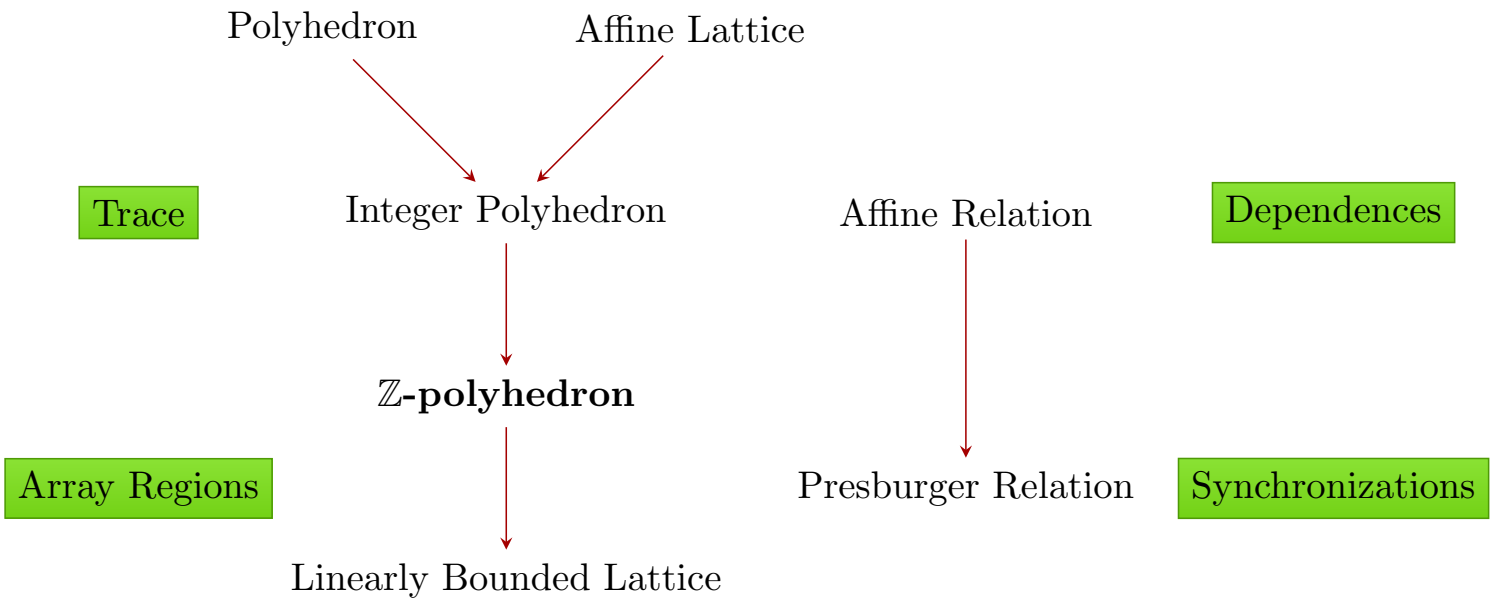
Affine Lattice

- $\vec{u}_0 \in \mathbb{Z}^d$
- $\vec{u}_1, \ldots, \vec{u}_d \in \mathbb{Z}^d$ linearly independant

$$\mathcal{L} = \{\vec{u}_0 + \sum_{i=1^d} \lambda_i \vec{u}_i,\ \lambda_i \in \mathbb{Z} \quad \forall i\}$$

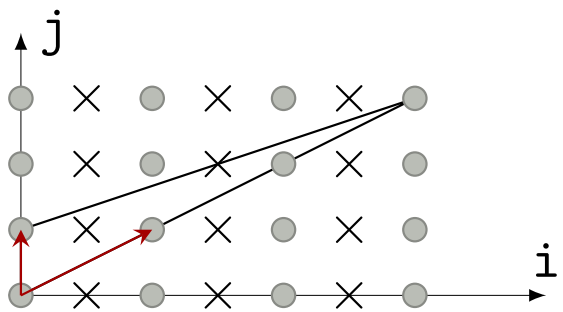


Zoology

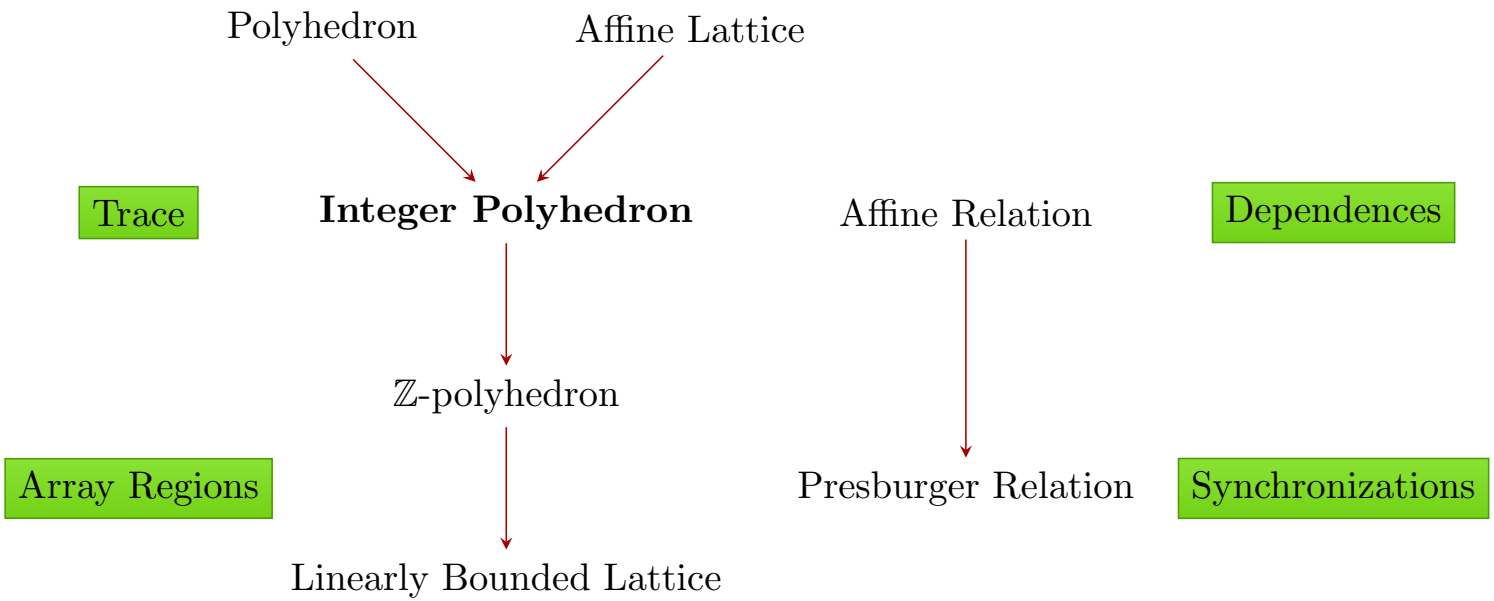


ℤ-polyhedron

$\mathcal{P}(\vec{n}) \cap \mathcal{L}$

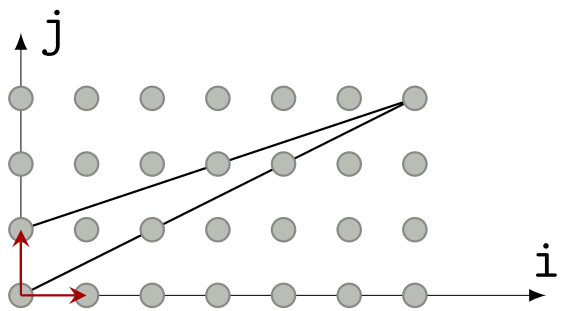


Zoology

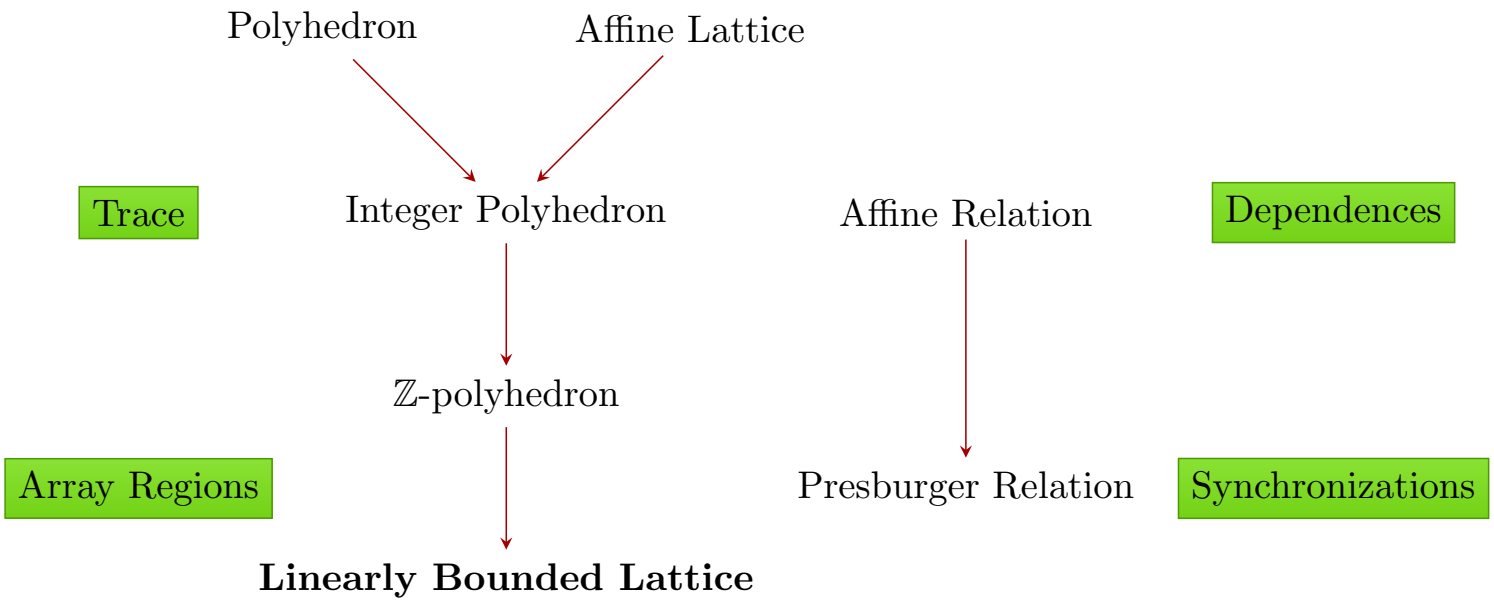


Integer Polyhedron

$\mathcal{P}(\vec{n}) \cap \mathbb{Z}^d$



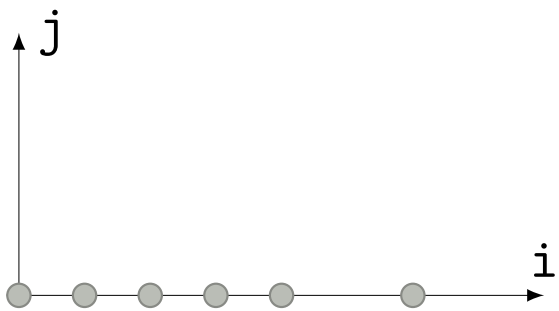
Zoology



Linearly Bounded Lattice (LBL)

Finite union of $\mathbb{Z}$-polyhedra

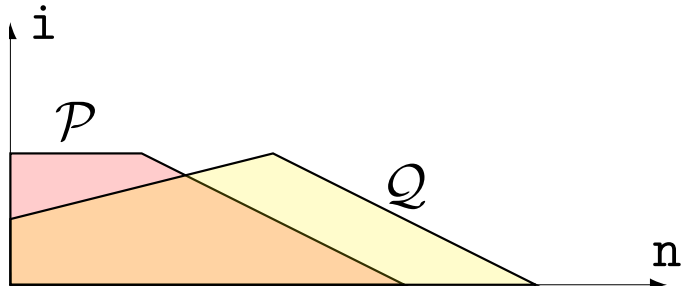
Presburger set — expensive!



Operations on Parametrized Sets

Basic set operations (union, subtract, project): free

Convex Hull: parametrized



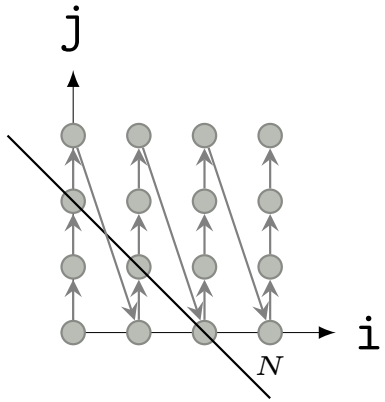
Integer linear programming: parametrized

$$\vec{x}^{\star}(\vec{n}) = \max_{\ll} \vec{x} \text{ s.t. } A\vec{x} + B\vec{n} + \vec{c} \geq 0$$

Toolbox: Polylib, Integer Set Library (ISL)

A Simple Analysis

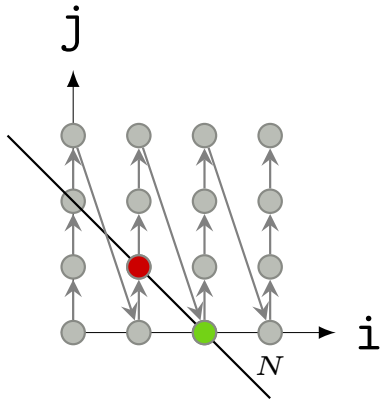
```
for i := 0 to N
  for j := 0 to N
    c[i+j] := c[i+j] + a[i]*b[j];
```



Q: Find the last loop iteration writing c[2]

A Simple Analysis

```
for i := 0 to N
  for j := 0 to N
    c[i+j] := c[i+j] + a[i]*b[j];
```



Q: Find the last loop iteration writing c[2]

A:

$$\max_{\ll}(i,j) \text{ s.t. } (i,j) \in [\![0,N]\!]^2 \wedge i+j=2$$

$$= \begin{cases} N \geq 2 : (2,0) \\ N = 1 : (1,1) \\ N \leq 0 : \bot \end{cases}$$

$\max_{\ll} \mathcal{P}(\vec{n})$ is almost piecewise affine

$$\max_{\ll} i \text{ s.t. } 2i \leq n = \lfloor n/2 \rfloor$$

$\max_{\ll} \mathcal{P}(\vec{n})$ is piecewise quasi-affine: integer division by a constant in pieces and functions

Expensive operations:

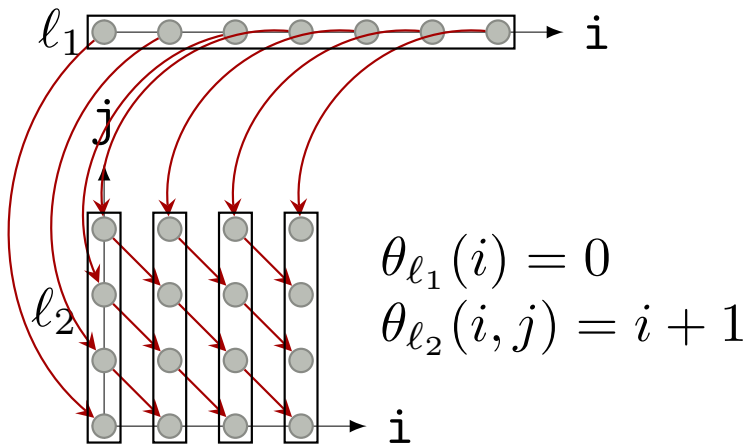
- Computing the max of several piecewise quasi-affine functions
- Simplifying the result (empty clauses, redondant clauses)

Outline

1. Automatic Parallelization (polyhedral stuff)
2. Intermediate Representations (Presburger stuff)
3. Scheduling (Farkas stuff)

(Affine) Scheduling

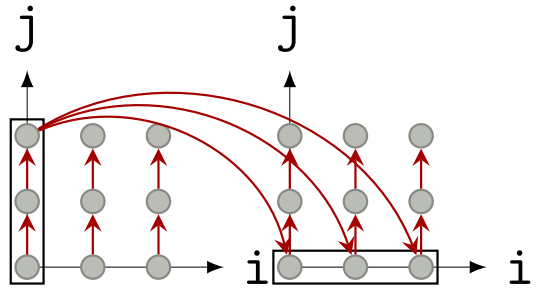
```
for i := 0 to 2N
  c[i] := 0;
for i := 0 to N
  for j := 0 to N
    c[i+j] := c[i+j] + a[i]*b[j];
```



- Affine-by-statement schedule: $\theta_\ell(\vec{i}) = A_\ell \vec{i} + \vec{b}_\ell$, dates ordered by the lexicographic order
- Correctness: $\langle \ell, \vec{i} \rangle \rightarrow \langle \ell', \vec{j} \rangle \Rightarrow \theta_\ell(\vec{i}) \ll \theta_{\ell'}(\vec{j})$
- Latency: $\min_{\ll}(u_S, v_S)$ s.t. $\theta_S(i)[d] \leq \vec{u}_S \cdot \vec{n} + \vec{v}_S \qquad \forall S \forall d$

Matrix Vector Composition

```
//y := Ax
for i := 1 to N
  for j := 1 to N
    y[i] += A[i,j]*x[j]; //S
//z := By
for i := 1 to N
  for j := 1 to N
    z[i] += B[i,j]*y[j]; //T
```



$$\theta_S(i,j) = (i,j) \qquad \theta_T(i,j) = (j+1,i)$$

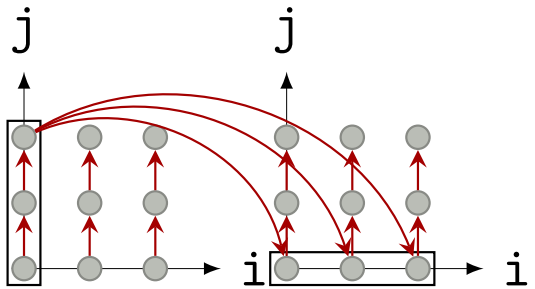
Checking the Schedule

For each dependence $s \rightarrow d$:

$$\Delta\theta = \theta(d) - \theta(s) \gg 0$$

$$\Delta\theta = (0, \ldots, 0, > 0, *)$$

	$S \rightarrow S$	$S \rightarrow T$	$T \rightarrow T$
$\Delta\theta[0]$	0	>0	>0
$\Delta\theta[1]$	>0	*	*



$$\theta_S(i,j) = (i,j) \qquad \theta_T(i,j) = (j+1, i)$$

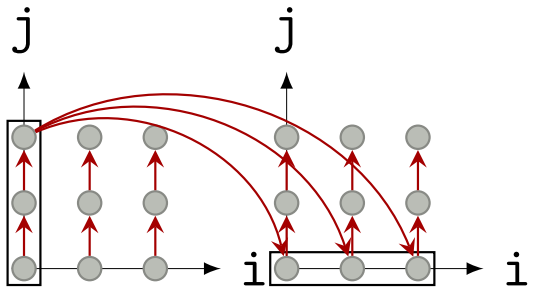
Checking the Schedule

For each dependence $s \rightarrow d$:

$$\Delta\theta = \theta(d) - \theta(s) \gg 0$$

$$\Delta\theta = (0, \ldots, 0, > 0, *)$$

	$S \rightarrow S$	$S \rightarrow T$	$T \rightarrow T$
$\Delta\theta[0]$	0	>0	>0
$\Delta\theta[1]$	>0	*	*



$$\theta_S(i,j) = (i,j) \qquad \theta_T(i,j) = (j+1,i)$$

Greedy Algorithm

d := 0

while there remains dependences to satisfy

Find θ[d] satisfying

$$\min_{\ll}(-\sum_{S,T}\delta_{S,T}, \sum_{S}\vec{u}_S, \sum_{S} v_S)$$

s.t.

$$\forall\langle S,\vec{x}\rangle \rightarrow \langle T,\vec{y}\rangle : \quad \theta_S(\vec{x})[d] + \delta_{S,T} \leq \theta_T(\vec{y})[d]$$

$$\forall S\ \forall\vec{x} \in \mathcal{D}_S : \quad 0 \leq \theta_S(\vec{x})[d] \leq \vec{u}_S \cdot \vec{n} + v_S$$

Remove dependences $\langle S,\vec{x}\rangle \rightarrow \langle T,\vec{y}\rangle$ with $\delta_{S,T} > 0$;

d := d + 1;

Farkas Lemma (Affine Form)

Lemma (Farkas, Affine Form)

Let:

- $\mathcal{P} = \{\vec{x},\ A\vec{x} + \vec{b} \geq 0\} \subset \mathbf{R}^d$ non empty
- $\phi : \mathbf{R}^d \to \mathbf{R}$ an affine form
- $\phi(\vec{x}) \geq 0 \quad \forall \vec{x} \in \mathcal{P}$

Then: $\exists \lambda_0 \vec{\lambda} \geq 0$ s.t.

$$\forall x \in \mathbf{R}^d : \phi(\vec{x}) = \lambda_0 + \vec{\lambda}^T(A\vec{x} + \vec{b}) = \mathbf{F}(\lambda_0, \vec{\lambda}, A, \vec{b})(\vec{x})$$

Main trick: Identification on $\vec{x}$

Proposition

The following assertions are equivalent:

- $\mathbf{F}(\lambda_0, \vec{\lambda}, A, \vec{b}) = 0$
- $\lambda_0 + \vec{\lambda}^T.\vec{b} = 0 \wedge A^T\vec{\lambda} = 0$

Main trick: Identification on $\vec{x}$

Proposition

The following assertions are equivalent:

- $\mathbf{F}(\lambda_0, \vec{\lambda}, A, \vec{b}) = 0$
- $\lambda_0 + \vec{\lambda}^T.\vec{b} = 0 \wedge A^T\vec{\lambda} = 0$

Proof

$$\forall \vec{x}: \qquad \mathbf{F}(\lambda_0, \vec{\lambda}, A, \vec{b})(\vec{x}) = \mathbf{F}(\lambda_0, \vec{\lambda}, A, \vec{b})(\vec{x})^T$$
$$= \lambda_0 + \vec{\lambda}^T.\vec{b} + \vec{x}^T A^T \vec{\lambda}$$

$\mathbf{F}(\lambda_0, \vec{\lambda}, A, \vec{b})(\vec{x}) = 0\ \forall \vec{x}$ iff the constant and the coefficients of $\vec{x} = (x_1, \ldots, x_d)$ are 0:

$$\lambda_0 + \vec{\lambda}^T.\vec{b} = 0 \wedge A^T\vec{\lambda} = 0$$

Farkas Toolbox

Farkas trick 1 (correctness)

The following assertions are equivalent:

- $\mathbf{F}(\lambda_0, \vec{\lambda}, A, \vec{b}) + \mathbf{F}(\mu_0, \vec{\mu}, C, \vec{d}) + \mathbf{F}(\nu_0, \vec{\nu}, E, \vec{f}) = 0$
- $\lambda_0 + \mu_0 + \nu_0 + \vec{\lambda}^T.\vec{b} + \vec{\mu}^T.\vec{d} + \vec{\nu}^T.\vec{f} = 0 \wedge$
  $A^T\vec{\lambda} + C^T\vec{\mu} + E^T\vec{\nu} = 0$

Farkas trick 2 (latency)

The following assertions are equivalent:

- $\mathbf{F}(\lambda_0, \vec{\lambda}, A, \vec{b})(\vec{x}) + \mathbf{F}(\mu_0, \vec{\mu}, C, \vec{d})(\vec{x}) + \vec{u}.\vec{x} + v = 0 \qquad \forall \vec{x}$
- $\lambda_0 + \mu_0 + \nu_0 + \vec{\lambda}^T.\vec{b} + \vec{\mu}^T.\vec{d} + \vec{\nu}^T.\vec{f} = 0 \wedge$
  $A^T\vec{\lambda} + C^T\vec{\mu} + \vec{u} = 0$

Back to Scheduling, correctness

Correctness constraint

$$\forall \langle S, \vec{x} \rangle \rightarrow \langle T, \vec{y} \rangle : \theta_T(\vec{y})[d] - \theta_S(\vec{x})[d] - \delta_{S,T} \geq 0$$

Translate to Farkas language

- $\theta_T(\vec{y})[d] - \theta_S(\vec{x})[d] - \delta_{S,T} = \mathbf{F}(\mu_0^{S,T}, \vec{\mu}^{S,T}, A_{S,T}, b_{S,T})(\vec{x}, \vec{y})$
- $\theta_S(\vec{x})[d] \geq 0 \Rightarrow \theta_S(\vec{x})[d] = \mathbf{F}(\lambda_0^S, \vec{\lambda}^S, A_S, b_S)(\vec{x}) \qquad \forall \vec{x}$

Apply Farkas Trick

$\mathbf{F}(\lambda_0, \vec{\lambda}^T, A_T, b_T) + \mathbf{F}(-\lambda_0^S - \delta_{S,T}, -\vec{\lambda}^S, A_S, b_S) +$
$\mathbf{F}(-\mu_0^{S,T}, -\vec{\mu}^{S,T}, A_{S,T}, b_{S,T}) = 0$

Back to Scheduling, latency

Latency constraint

$$\forall S\ \forall \vec{x} \in \mathcal{D}_S : \vec{u}_S \cdot \vec{n} + v_S - \theta_S(\vec{x})[d] \geq 0$$

Translate to Farkas language

- $\vec{u}_S \cdot \vec{n} + v_S - \theta_S(\vec{x})[d] = \mathbf{F}(\nu_0^S, \vec{\nu}^S, A_S, b_S)(\vec{x}, \vec{n})$
- $\theta_S(\vec{x})[d] \geq 0 \Rightarrow \theta_S(\vec{x})[d] = \mathbf{F}(\lambda_0^S, \vec{\lambda}^S, A_S, b_S)(\vec{x})$ $\forall \vec{x}$

Apply Farkas Trick

$$\mathbf{F}(-\lambda_0^S - \delta_{S,T}, -\vec{\lambda}^S, A_S, b_S) + \mathbf{F}(-\nu_0^S, -\vec{\nu}^S, A_S, b_S) + \vec{u}_S \cdot \vec{n} + v_S = 0$$

Open Issues

Top 3 Expensive operations:

1. $\min_{\ll}$ / $\max_{\ll}$ on piecewise quasi-affine functions
2. Projections on integer polyhedra
3. $\min_{\ll}$ / $\max_{\ll}$ on integer polyhedra

Farkasseries

- Assuming integer polyhedron vs Presburger set
- Big system (though sparse) $\rightsquigarrow$ projections required

variables: #dep(#constraints/dep) + #assign(#constraints/iteration domain)

constraints: #dep(#var source + #var target) + #assign (#var)