Potential and Challenges of Two-Variable-Per-Inequality Sub-Polyhedral Compilation

Ramakrishna Upadrasta\textsuperscript{1,2} \hspace{1em} Albert Cohen\textsuperscript{1}

\textsuperscript{1}PARKAS group, INRIA, École Normale Supérieure

\textsuperscript{2}Paris-Sud 11 Université

April the 3rd, 2011
Outline

- **Introduction**
  - Main goals
  - An algorithmic estimate of complexity of scheduling
  - Causes of unscalability and need for approximations
- **Sub-Polyhedra**
  - Use in static analysis
  - Existing abstract domains
  - Two-Variable-Per-Inequality Sub-Polyhedra
- **Sub-Polyhedral Compilation**
  - Dependence approximation
  - Constraint approximation
- **Conclusions and Future Work**
Polyhedral Compilation

**Overall goal:** Compile SCoPs using Polyhedral techniques
- Lot of recent advancements
  - ... Bondhugula et al. 2008, ..., Pouchet et al. 2011 ...
- Academia, Industry, Open Source
  - PLUTO
  - IBM, RStream
  - GRAPHITE, LLVM
- Good optimizing compiler $\iff$ Has good polyhedral optimizer

**Our goal:** Compile larger and larger SCoPs using Polyhedral model
Possible Scenarios for Unscalability

Unscalability from **larger and more complex** SCoPs:

- longer source codes  
  (inlining, unrolling)
- difficult source languages
- source in medium/low level intermediate form
- preceded by more powerful static analysis  
  (alias/pointer analysis)

**Result:** Large \# of loops & large \# of statements
Algorithmic View: Assumptions & Estimates

Assumptions:

- **Statements**: shallow depth
- **Overall system**: #variables $\approx$ #constraints
- **Solvers**: LP feasibility (with simplex) takes cubic time

**Complexity estimate of scheduling**: $\approx \mathcal{O}(|E|^3)$

($|E|$: #dependences)
Scalability: Need for Approximation

Good engineering (it works now!) but, if not backed by algorithmics may hit the scalability problem sooner or later.
Scalability: Need for Approximation

Good engineering (it works now!) but, if not backed by algorithmics may hit the scalability problem sooner or later.

Motivational assumptions:

- Polyhedral compilation will hit the scalability problem
- We need to approximate:
  - ad hoc basis
  - by understanding the complexity of algorithms

Possible solution: Algorithms using approximations of Polyhedra
Sub-polyhedra as Abstract Domain in Static Analysis

**Goal:** Solve abstract interpretation and run-time verification problems

**Requirements**
- Polyhedra used as elements of lattices ($\langle D, \subseteq, \cup, \cap \rangle$)
- Iteratively solve recursive (“program-flow”) equations on lattices
- Regularly need generator representation ($\& \mathcal{H} \longleftrightarrow \mathcal{V}$ conversion)
- >1000’s of statements
Sub-polyhedra as Abstract Domain in Static Analysis

**Goal:** Solve abstract interpretation and run-time verification problems

**Requirements**
- Polyhedra used as elements of lattices ($\langle D, \subseteq, \cup, \cap \rangle$)
- Iteratively solve recursive (“program-flow”) equations on lattices
- Regularly need generator representation ($\& \mathcal{H} \iff \mathcal{V}$ conversion)
- > 1000’s of statements

**Conclusion:** Convex polyhedra does not scale

“... general polyhedron domain has a memory and time cost unbounded in theory and exponential in practice” [Miné-06]  
“... the price to pay for using it [convex polyhedra] is too high: the analysis not scale beyond dozens of variables, ... while mostly one wants to solve hundreds of constraints and variables.” [Laviron-Logoizzo-09]

**Result:** Various Sub-Polyhedra have been proposed.
Some Interesting Sub-polyhedral Abstract Domains

- **Interval**: \(a \leq x_i \leq b\)
- **Octagon (UTVPI)**: \(\pm x_i \pm x_j \leq c\)
- **TVPI**: \(ax_i + bx_j \leq c\)
- **Convex Polyhedra**: \(\sum a_i x_i \leq c\)

Precision

\[
\frac{\text{Intervals } \subset \text{ Octagons (UTVPI)} \subset \text{ TVPI} \subset \text{ Poly}}{\text{Cost}}
\]
Sub-Polyhedra: TVPI \((ax_i + bx_j \leq c)\)

- LP optimization:
  - Polynomial time \(\Theta(m^3 n^2 \log A)\) [Wayne-99]
  - Reduction to graph theory network flow problem (Min-cost flow reduction, Max-flow specialization also possible)

- LP feasibility and Fourier-Motzkin:
  - **Strongly polynomial time** \(\Theta(mn^2 \log m)\) [Hochbaum-Naor-94] (and [Cohen-Megiddo-94])
  - Simple algorithm (uses clever pruning)

- Scalable algorithms of polygons easily extend to general \(n-d\)-TVPI
  \(\implies\) Low complexity geometric algorithms
Sub-Polyhedra: Octagons (UTVPI) \((\pm x_i; \pm x_j \leq c)\)

- UTVPI \(\subset\) TVPI \(\implies\) good complexity measures
  - LP optimization too is strongly polynomial time [Tardos-86]
  - Faster, simpler LP-feasibility expected
    (entries just \(\{0, \pm 1\}\)!)  
- Cubic time geometric operations
- Low cost implementation using simple data-structures
- Tools/support: advanced and well tested
  - Miné: Astree project, Apron Library
  - Bagnara et al: PPL support
Sub-Polyhedral Compilation
Polyhedral Compilation using Sub-polyhedra

Where to use Sub-Polyhedra?

- Dependence analysis
- **Scheduling**
  - Dependence over-approximation
  - Constraint under-approximation
- Code generation
Polyhedral Compilation using Sub-polyhedra

Where to use Sub-Polyhedra?

- Dependence analysis
- **Scheduling**
  - Dependence over-approximation
  - Constraint under-approximation
- Code generation

Conversion of Convex Polyhedra to Sub-Polyhedra:

- **Legality**: over/under approximate accordingly
- **Cost**: small cost of conversion
- **Closest**: closest \((\mathbb{Q}/\mathbb{I})\) Sub-Polyhedron to original
- **Expressiveness**: preserve interesting solution points
Dependence Approximation using Sub-Polyhedra
Dependence Approximation: Different Levels of Precisions

Over-approximations of dependences
- are legal (some loss of precision)
- long history

- Dependence Level (Allen-Kennedy)
- Dependence Direction Vector (Wolf-Lam)
- Dependence Cone (Irignon-Triplet)
- Polyhedral Distance Vector (Darte-Vivien)
- Dependence Polyhedron (Feautrier)
- \( \mathcal{Z} \)-Polyhedra (Le Verge-Wilde)
Dependence Approximation: Motivation for Sub-Polyhedra

\[ DL \subset DDV \subset DC \subset D \subset DP \subset DI \subset \cdots \]

- sub-polyhedral
- polyhedral
- beyond polyhedral

- dependence abstractions \( \equiv \) schedule expressiveness
  [Yang-Ancourt-Irigoin-94, Darte-Vivien-97]

- dependence abstractions \( \equiv \) scalability
Dependence Approximation: Motivation for Sub-Polyhedra

\[
\begin{align*}
&\text{sub-polyhedral} \subset \text{DDV} \subset \text{DC} \subset D \subset \text{DP} \subset \text{DI} \subset \cdots \\
&\text{dependence abstractions} \iff \text{schedule expressiveness} \\
&\text{dependence abstractions} \iff \text{scalability}
\end{align*}
\]

Use Sub-Polyhedral dependence abstractions:

- retain expressiveness of polyhedra \(\Rightarrow\) affine transformations
- exploit the scalable algorithms offered by Sub-Polyhedra
Dependence Approximation using TVPI?

\[
\text{Intervals} \subseteq \text{Octagons} \subseteq \text{TVPI} \subseteq \text{Poly} + \mathcal{N}
\]

**Parametric TVPI:**

- Exact dependence analysis [of Feautrier-88] **needs** parametrized-ILP.
  But, ILP on $\mathbb{I}$-TVPI is NP-Complete [Lagarias-85]!

- Distance Vectors $\approx$ TVPI + parametrization.
  But, parametrization $\implies$ EXP$\#$ of contexts in $|\mathcal{N}|$ [Feautrier-88]!
Dependence Approximation using TVPI?

Intervals $\subset$ Octagons $\subset$ TVPI $\subset$ Poly + $\mathcal{N}$

Parametric TVPI:
- Exact dependence analysis [of Feautrier-88] needs parametrized-ILP. But, ILP on $\mathbb{I}$-TVPI is NP-Complete [Lagarias-85]!
- Distance Vectors $\approx$ TVPI + parametrization. But, parametrization $\Rightarrow$ EXP$\#$ of contexts in $|\mathcal{N}|$ [Feautrier-88]!

Non-Parametric TVPI:
- TVPI over-approximation of non-parametric dependences
- Use non-parametric TVPI for JIT compilation?
Constraint Approximation using Sub-Polyhedra
Constraint Approximation: Motivation

Reminder: Feautrier’s scheduler:

Input: Dependence Polyhedron per e: $D_e(\mu, I, N)$

- Apply Farkas Lemma:

$$ (\mu, I, N) \equiv (\lambda, I, N) \implies (\mu, \lambda) $$

$P_e(\mu, \lambda)$ is constraint polyhedron on edge e.

- Construct $P = \bigcap_{e \in E} P_e$
- Solve $P$ for feasible solution points

New method: Under-approximate each Farkas system: $P_e \implies UA(P_e)$. 
Constraint Approximation: Algorithm

**New constraint system:**

\[
UA(\mathcal{P}) = \bigcap_{e \in E} UA(\mathcal{P}_e)
\]
Constraint Approximation: Algorithm

New constraint system:

\[ UA(\mathcal{P}) = \bigcap_{e \in E} UA(\mathcal{P}_e) \]

Scheduling Algorithm using TVPI Approximation of Constraints

1. Under-approximate each \( \mathcal{P}_e \mapsto UA_{TVPI}(\mathcal{P}_e) \)
2. Construct under-approximated system:

\[ UA_{TVPI}(\mathcal{P}) = \bigcap_{e \in E} UA_{TVPI}(\mathcal{P}_e) \]

3. Use FM of Hochbaum-Naor to find feasible points in \( UA_{TVPI}(\mathcal{P}) \)
Constraint Approximation: Details

Under-approximation:
- can be done per dependence edge
- may afford high cost
  (each $P_e$ is relatively small)

TVPI under-approximation:
- A framework with hierarchy of approximations
  \[
  \frac{\text{Precision}}{\text{Cost}} \quad \begin{align*}
  UA_{\text{Interval}}(P) &\subset UA_{\text{UTVPI}}(P) \\
  &\subset UA_{\text{TVPI}}(P)
  \end{align*}
  \]
- **Complexity of Scheduling:** Strongly polynomial time
Constraint Approximation: Questions and Problems

Questions on under-approximation:
- cost
- method followed
  - direct: $\mathcal{P} \rightarrow \text{UA}(\mathcal{P})$
  - indirect: $\mathcal{P} \rightarrow \mathcal{P}^* \dashv \circ \text{OA}(\mathcal{P}^*) \rightarrow \text{UA}(\mathcal{P})$
- how close?
- losing the redundant/useless points of $\mathcal{P}_e$
Wrapping up
Questions raised (Beaucoup!)

**Summary:**
- Scalability question: algorithmic view
- Use of Sub-Polyhedra in static analysis
- Precision vs. cost tradeoff using Sub-Polyhedra
  - $\Rightarrow$ achieve scalability
- Use Sub-Polyhedral approximation in scheduling
  - Questions on dependence over-approximation
  - Method based on constraint under-approximation
Challenges (Beaucoup!)

- Better algorithms for TVPI/UTVPI under/over approximation
- Empirical work (lots!)
- Are we going to face a scalability problem? (How? Where?)

**Summary:** We think that there is potential

- in Sub-Polyhedral compilation
- in using TVPI and UTVPI systems

We think there are lot of challenging problems to be solved in this area!
Merci et Questions

Merci Beaucoup!

- Paul Feautrier
- Axel Simon (ENS-Paris/TU-Munich)
- Reviewers

“Although this may seem a paradox, all exact science is dominated by the idea of approximation” – Bertrand Russell

Questions?
Backup
### Some Existing Sub-polyhedral Abstract Domains

<table>
<thead>
<tr>
<th>Sub-polyhedra</th>
<th>Approximation (nature of constraint) ((a, b, c \in \mathbb{Q}))</th>
<th>Proposed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals (&quot;Boxes&quot;)</td>
<td>(a \leq x_i \leq b)</td>
<td>Cousot-Cousot</td>
</tr>
<tr>
<td>DBM</td>
<td>(x_i - x_j \leq c; x_i, x_j \geq 0)</td>
<td>Bagnara et al, Miné</td>
</tr>
<tr>
<td>Octagons (UTVPI)</td>
<td>(\pm x_i \pm x_j \leq c)</td>
<td>Miné</td>
</tr>
<tr>
<td>TVPI</td>
<td>(ax_i + bx_j \leq c)</td>
<td>Simon, Howe, King</td>
</tr>
<tr>
<td>Sub-Polyhedra (SubPoly)</td>
<td>LinEq (\otimes) Interval</td>
<td>Laviron, Logozzo</td>
</tr>
<tr>
<td>Pentagons</td>
<td>(a \leq x_i \leq b \land x_i &lt; x_j)</td>
<td>Logozzo Fahndrich</td>
</tr>
<tr>
<td>Octahedra</td>
<td>(\pm x_i \pm \cdots \pm x_j \leq c)</td>
<td>Clariso Cortadella</td>
</tr>
<tr>
<td>Convex Polyhedra</td>
<td>(\sum a_i x_i \leq c)</td>
<td>–</td>
</tr>
</tbody>
</table>
## General and Sub-polyhedra: Comparison (Worstcase Complexity)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Convex Polyhedra</th>
<th>TVPI</th>
<th>UTVP/DBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEN ($\mathcal{V} \leftrightarrow \mathcal{H}$)</td>
<td>$\text{EXP}(n), \text{EXP}(d)$</td>
<td>$\text{EXP}(n), \text{EXP}(d)$</td>
<td>$\text{EXP}(n), \text{EXP}(d)$</td>
</tr>
<tr>
<td>LP-OPT</td>
<td>$\text{WNPC}$</td>
<td>$\Theta(m^3n^2 \log B)$</td>
<td>$\Theta(m^3n^2 \log B)$</td>
</tr>
<tr>
<td>LP-OPT (2 var in Objfun)</td>
<td>$\text{WNPC}$</td>
<td>$\Theta_c(\log m)$</td>
<td>$\Theta_c(1)$</td>
</tr>
<tr>
<td>LP-FEAS</td>
<td>$\text{WNPC}$</td>
<td>$\Theta(mn^2 \log m)$</td>
<td>$\Theta(mn^2 \log m)$</td>
</tr>
<tr>
<td>FM</td>
<td>$\omega(\text{EXP}(n))$, $\omega(\text{EXP}(d))$</td>
<td>$\Theta(mn^2 \log m)$</td>
<td>$\Theta(mn^2 \log m)$</td>
</tr>
<tr>
<td>CLOSURE</td>
<td>$-$</td>
<td>$\Theta(n^3)$</td>
<td>$\Theta(n^3)$</td>
</tr>
<tr>
<td>ILP</td>
<td>$\text{NPC}$</td>
<td>$\text{NPC}$</td>
<td>$\text{NPC}$?</td>
</tr>
</tbody>
</table>
More Future Work

- Improving approximation algorithm
  - Over-approximation: Currently $O(n^7)$ algorithm
  - Under-approximation: use direct method or over-approximation?

- Offline/Online method using approximation
  - Offline: Conversion and Approximation
  - Online: Finding schedule
  - Matching of paradigms for commonly occurring dependences?