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

Ramakrishna Upadrasta[1,2] Albert Cohen[1]

[1]PARKAS group, INRIA, École Normale Supérieure

[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 $\Longleftrightarrow$ 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:
  - ~~adhoc basis~~
  - by understanding the complexity of algorithms

Possible solution: Algorithms using approximations of Polyhedra

Sub-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 ($\langle D, \sqsubseteq, \sqcup, \sqcap \rangle$)
- 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

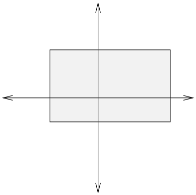
- Polyhedra used as elements of lattices ($\langle D, \sqsubseteq, \sqcup, \sqcap \rangle$)
- Iteratively solve recursive ("program-flow") equations on lattices
- Regularly need generator representation (& $\mathcal{H} \longleftrightarrow \mathcal{V}$ conversion)
- $>$ 1000's of statements

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-Logozzo-09]

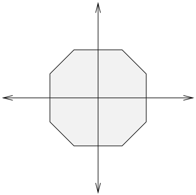
Result: Various Sub-Polyhedra have been proposed.

Some Interesting Sub-polyhedral Abstract Domains



Interval

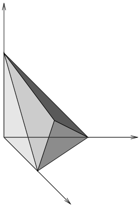
$a \leq x_i \leq b$



Octagon (UTVPI)

Unit Two Variable Per Inequality

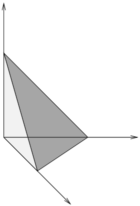
$\pm x_i \pm x_j \leq c$



TVPI

Two Variable Per Inequality

$ax_i + bx_j \leq c$



Convex Polyhedra

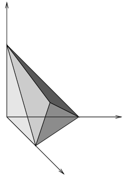
$\sum a_i x_i \leq c$

$$\text{Intervals} \subset \text{Octagons (UTVPI)} \subset \text{TVPI} \subset \text{Poly}$$

$$\xrightarrow{\text{Precision}} \qquad \xrightarrow[\text{Cost}]{}$$

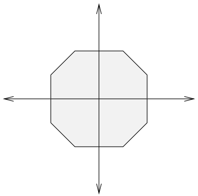
Sub-Polyhedra: TVPI ($ax_i + bx_j \leq c$)

- 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 $\Longrightarrow$ Low complexity geometric algorithms



Sub-Polyhedra: Octagons (UTVPI) ($\pm x_i \pm x_j \leq c$)

- $\mathrm{UTVPI} \subset \mathrm{TVPI} \implies$ 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:

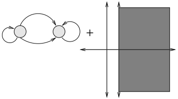
- Legality: over/under approximate accordingly
- Cost: small cost of conversion
- Closest: closest ($\mathbb{Q}/\mathbb{I}$) Sub-Polyhedron to original
- Expressiveness: preserve interesting solution points

Dependence Approximation using Sub-Polyhedra

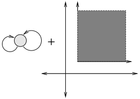
Dependence Approximation: Different Levels of Precisions

Over-approximations of dependences

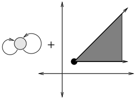
- are legal (some loss of precision)
- long history



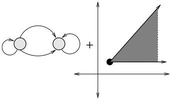
Dependence Level
(Allen-Kennedy)



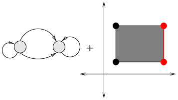
Dependence Direction Vector
(Wolf-Lam)



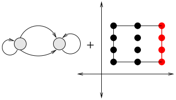
Dependence Cone
(Irigion-Triolet)



Polyhedral DistanceVector
(Darte-Vivien)



Dependence Polyhedron
(Feautrier)



$\mathcal{Z}$-Polyhedra
(Le Verge-Wilde)

Dependence Approximation: Motivation for Sub-Polyhedra

$$\underbrace{\mathrm{DL} \subset \mathrm{DDV} \subset \mathrm{DC}}_{\text{sub-polyhedral}} \subset \underbrace{\mathrm{D} \subset \mathrm{DP}}_{\text{polyhedral}} \subset \underbrace{\mathrm{DI} \subset \cdots}_{\text{beyond polyhedral}}$$

- dependence abstractions $\rightleftharpoons$ schedule expressiveness [Yang-Ancourt-Irigoin-94, Darte-Vivien-97]
- dependence abstractions $\overset{?}{\rightleftharpoons}$ scalability

Dependence Approximation: Motivation for Sub-Polyhedra

$$\underbrace{\mathrm{DL} \subset \mathrm{DDV} \subset \mathrm{DC}}_{\text{sub-polyhedral}} \subset \underbrace{\mathrm{D} \subset \mathrm{DP}}_{\text{polyhedral}} \subset \underbrace{\mathrm{DI} \subset \cdots}_{\text{beyond polyhedral}}$$

- dependence abstractions $\rightleftharpoons$ schedule expressiveness [Yang-Ancourt-Irigoin-94, Darte-Vivien-97]
- dependence abstractions $\overset{?}{\rightleftharpoons}$ scalability

Use Sub-Polyhedral dependence abstractions:

- retain expressiveness of polyhedra $\Rightarrow$ affine transformations
- exploit the scalable algorithms offered by Sub-Polyhedra

Dependence Approximation using TVPI?

$$\underbrace{\text{Intervals} \subset \text{Octagons} \subset \text{TVPI} \subset \text{Poly}}_{\mathcal{I}} + \mathcal{N}$$

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 $\Longrightarrow$ EXP# of contexts in $|\mathcal{N}|$ [Feautrier-88]!

Dependence Approximation using TVPI?

$$\underbrace{\text{Intervals} \subset \text{Octagons} \subset \text{TVPI} \subset \text{Poly}}_{\mathcal{I}} + \mathcal{N}$$

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 $\Longrightarrow$ 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: $\mathcal{D}_e(\mu, \mathcal{I}, \mathcal{N})$

- Apply Farkas Lemma:

$$(\mu, \mathcal{I}, \mathcal{N}) \equiv (\lambda, \mathcal{I}, \mathcal{N}) \Longrightarrow (\mu, \lambda)$$

  $\mathcal{P}_e(\mu, \lambda)$ is constraint polyhedron on edge e.

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

New method: Under-approximate each Farkas system: $\mathcal{P}_e \Longrightarrow \mathrm{UA}(\mathcal{P}_e)$.

Constraint Approximation: Algorithm

New constraint system:

$$\mathrm{UA}(\mathcal{P}) = \bigcap_{e \in E} \mathrm{UA}(\mathcal{P}_e)$$

Constraint Approximation: Algorithm

New constraint system:

$$\mathrm{UA}(\mathcal{P}) = \bigcap_{e \in E} \mathrm{UA}(\mathcal{P}_e)$$

Scheduling Algorithm using TVPI Approximation of Constraints

1. Under-approximate each $\mathcal{P}_e \implies \mathrm{UA}_{\mathrm{TVPI}}(\mathcal{P}_e)$
2. Construct under-approximated system:

$$\mathrm{UA}_{\mathrm{TVPI}}(\mathcal{P}) = \bigcap_{e \in E} \mathrm{UA}_{\mathrm{TVPI}}(\mathcal{P}_e)$$

3. Use FM of Hochbaum-Naor to find feasible points in $\mathrm{UA}_{\mathrm{TVPI}}(\mathcal{P})$

Constraint Approximation: Details

Under-approximation:

- can be done per dependence edge
- may afford high cost
  (each $\mathcal{P}_e$ is relatively small)

TVPI under-approximation:

- A framework with hierarchy of approximations

$$\xrightarrow{\text{Precision}} \\ \mathrm{UA}_{\mathrm{Interval}}(\mathcal{P}) \subset \mathrm{UA}_{\mathrm{UTVPI}}(\mathcal{P}) \subset \mathrm{UA}_{\mathrm{TVPI}}(\mathcal{P}) \\ \xrightarrow[\text{Cost}]{}$$

- Complexity of Scheduling: Strongly polynomial time

Constraint Approximation: Questions and Problems

Questions on under-approximation:

- cost
- method followed
  - direct: $\mathcal{P} \rightarrow \mathrm{UA}(\mathcal{P})$
  - indirect: $\mathcal{P} \rightarrow \mathcal{P}^* \rightsquigarrow \mathrm{OA}(\mathcal{P}^*) \rightarrow \mathrm{UA}(\mathcal{P})$
- 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 $\overset{?}{\Longrightarrow}$ 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

Sub-polyhedra	Approximation (nature of constraint) $(a, b, c \in \mathbb{Q})$	Proposed by
Intervals ("Boxes")	$a \leq x_i \leq b$	Cousot-Cousot
DBM	$x_i - x_j \leq c; x_i, x_j \geq 0$	Bagnara et al, Miné
Octagons (UTVPI)	$\pm x_i \pm x_j \leq c$	Miné
TVPI	$ax_i + bx_j \leq c$	Simon, Howe, King
Sub-Polyhedra (SubPoly)	$\mathrm{LinEq} \otimes \mathrm{Interval}$	Laviron, Logozzo
Pentagons	$a \leq x_i \leq b \wedge x_i < x_j$	Logozzo Fahndrich
Octahedra	$\pm x_i \pm \cdots \pm x_j \leq c$	Clariso Cortadella
Convex Polyhedra	$\sum a_i x_i \leq c$	–

General and Sub-polyhedra: Comparison (Worstcase Complexity)

Problem	Convex Polyhedra	TVPI	UTVPI/DBM
GEN ($\mathcal{V} \leftrightarrow \mathcal{H}$)	$\mathrm{EXP}(n)$, $\mathrm{EXP}(d)$	$\mathrm{EXP}(n)$, $\mathrm{EXP}(d)$	$\mathrm{EXP}(n)$, $\mathrm{EXP}(d)$
LP-OPT	WNPC	$\Theta(m^3 n^2 \log B)$	$\Theta(m^3 n^2 \log B)$
LP-OPT (2 var in Objfun)	WNPC	$\Theta_c(\log m)$	$\Theta_c(1)$
LP-FEAS	WNPC	$\Theta(mn^2 \log m)$	$\Theta(mn^2 \log m)$
FM	$\omega(\mathrm{EXP}(n))$, $\omega(\mathrm{EXP}(d))$	$\Theta(mn^2 \log m)$	$\Theta(mn^2 \log m)$
CLOSURE	–	$\Theta(n^3)$	$\Theta(n^3)$
ILP	NPC	NPC	NPC?

More Future Work

- Improving approximation algorithm
  - Over-approximation: Currently $\mathcal{O}(n^7)$ 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?