Towards an Automatic Parametric WCET Analysis

Stefan Bygde and Björn Lisper

July 1, 2008

School of Innovation, Design and Engineering, Mälardalen University
Box 883, S-721 23 Västerås, Sweden
{stefan.bygde, bjorn.lisper}@mdh.se

1This work is supported by the Swedish Foundation for Strategic Research via the strategic research centre PROGRESS
Table of contents

1 Introduction
2 Method
   • Workflow
3 Implementation
   • Assumptions and Definitions
   • Abstract Interpretation
   • Symbolic Counting
   • Parametric Integer Programming
   • Combination
4 Conclusions
5 Future Work
Method

- We present a practical implementation of a method to calculate a parametric WCET
- The method was first presented by Björn Lisper, WCET’03
- It obtains a formula representing the WCET in terms of input variables of the program
- Works for possibly unstructured, deterministic and terminating programs
Method

- We present a practical implementation of a method to calculate a parametric WCET.
- The method was first presented by Björn Lisper, WCET’03.
  - It obtains a formula representing the WCET in terms of input variables of the program.
  - Works for possibly unstructured, deterministic and terminating programs.
Method

- We present a practical implementation of a method to calculate a parametric WCET
- The method was first presented by Björn Lisper, WCET’03
- It obtains a formula representing the WCET in terms of input variables of the program
- Works for possibly unstructured, deterministic and terminating programs
Method

- We present a practical implementation of a method to calculate a parametric WCET
- The method was first presented by Björn Lisper, WCET’03
- It obtains a formula representing the WCET in terms of input variables of the program
- Works for possibly unstructured, deterministic and terminating programs
Contribution

- Showing that the analysis can be implemented for simple programs
- Reducing complexity of the analysis by reducing the number of variables
- Simplifying Pugh’s method for counting solutions of presburger formulae to count points inside convex polyhedra
- Identifying the need for simplification of the resulting parametric WCET formula
Contribution

- Showing that the analysis can be implemented for simple programs
- Reducing complexity of the analysis by reducing the number of variables
  - Simplifying Pugh’s method for counting solutions of Presburger formulae to count points inside convex polyhedra
  - Identifying the need for simplification of the resulting parametric WCET formula
Contribution

- Showing that the analysis can be implemented for simple programs
- Reducing complexity of the analysis by reducing the number of variables
- Simplifying Pugh’s method for counting solutions of presburger formulae to count points inside convex polyhedra
- Identifying the need for simplification of the resulting parametric WCET formula
Contribution

• Showing that the analysis can be implemented for simple programs
• Reducing complexity of the analysis by reducing the number of variables
• Simplifying Pugh’s method for counting solutions of presburger formulae to count points inside convex polyhedra
• Identifying the need for simplification of the resulting parametric WCET formula
Contribution

- Showing that the analysis can be implemented for simple programs
- Reducing complexity of the analysis by reducing the number of variables
- Simplifying Pugh’s method for counting solutions of presburger formulae to count points inside convex polyhedra
- Identifying the need for simplification of the resulting parametric WCET formula
Outline of the method presented at WCET'03

Workflow
Structural Analysis

- Derives constraints on execution counts
- Imposed by the CFG structure only.
Low-Level Analysis

Obtains WCETs for atomic parts of a program
Parametric IPET

Maximises $\vec{c}^T \vec{x}$ subject to the structural constraints

The execution counts are assumed to be bounded from above by execution count parameters

The result is a formula in terms of these parameters
Abstract Interpretation

Derives constraints on the values of the program variables
Symbolic Counting

Counts the maximum number of possible environments (run-time states) at each program point

The result of the abstract interpretation constrains which environments are possible

The result is symbolic in the input parameters
The WCET formula obtained from the parametric IPET is symbolic in execution count parameters.

The execution counts are bounded by the possible number of environments.

The execution count parameters are substituted by the functions from the symbolic counting.
Assumptions and Definitions

- Modular implementation of the different phases using C++ and Haskell
- Programs are modelled as CFGs
  - Variables are integer valued or Booleans
  - The CFG has entry, exit, assignment, conditional and merge nodes
  - No pointers, arrays or function calls
  - Assignments and conditionals are considered to be either linear or unknown
Assumptions and Definitions

- Modular implementation of the different phases using C++ and Haskell
- Programs are modelled as CFGs
  - Variables are integer valued or Booleans
  - The CFG has entry, exit, assignment, conditional and merge nodes
  - No pointers, arrays or function calls
  - Assignments and conditionals are considered to be either linear or unknown
Assumptions and Definitions cont’d

- Each program $P$ has a set of variables $\mathcal{V}_P$
- Each program has a set of input variables $\mathcal{I}_P \subseteq \mathcal{V}_P$ and each input variable
  - Affects program flow
  - Remains unchanged in the program part under analysis
- The atomic WCETs for each program point is assumed to be a single constant
- Structural analysis is trivial since the program is represented as a CFG
Assumptions and Definitions cont’d

- Each program $P$ has a set of variables $\mathcal{V}_P$
- Each program has a set of input variables $\mathcal{I}_P \subseteq \mathcal{V}_P$ and each input variable
  - Affects program flow
  - Remains unchanged in the program part under analysis

- The atomic WCETs for each program point is assumed to be a single constant
- Structural analysis is trivial since the program is represented as a CFG
Each program $P$ has a set of variables $V_P$

- Each program has a set of input variables $I_P \subseteq V_P$ and each input variable
  - Affects program flow
  - Remains unchanged in the program part under analysis

- The atomic WCETs for each program point is assumed to be a single constant

- Structural analysis is trivial since the program is represented as a CFG
Assumptions and Definitions cont’d

- Each program $P$ has a set of variables $\mathcal{V}_P$
- Each program has a set of input variables $\mathcal{I}_P \subseteq \mathcal{V}_P$ and each input variable
  - Affects program flow
  - Remains unchanged in the program part under analysis
- The atomic WCETs for each program point is assumed to be a single constant
- Structural analysis is trivial since the program is represented as a CFG
Abstract Interpretation

- Abstract Interpretation obtains a super set of the possible environments
- We have implemented the polyhedral domain with a existing library called ”New Polka”
- Encloses the possible environments inside a convex polyhedron
Abstract Interpretation obtains a super set of the possible environments.

We have implemented the polyhedral domain with a existing library called ”New Polka”.

Encloses the possible environments inside a convex polyhedron.
Abstract Interpretation

- Abstract Interpretation obtains a super set of the possible environments.
- We have implemented the *polyhedral domain* with an existing library called ”New Polka”.
- Encloses the possible environments inside a convex polyhedron.
Example

\[
\begin{align*}
    a_0 &= \top \\
    a_1 &= \{ i = 0 \} \\
    a_2 &= \{ 0 \leq i \leq n + 1 \} \\
    a_3 &= \{ i \geq 0, i \geq n + 1 \} \\
    a_4 &= \{ 0 \leq i \leq n \} \\
    a_5 &= \{ 1 \leq i \leq n + 1 \}
\end{align*}
\]
W. Pugh proposed a method for counting solutions to presburger formulae (SIGPLAN’94)

- The method computes generalised sums which are sums over several variables with complex constraints
Pugh’s Method

- W. Pugh proposed a method for counting solutions to Presburger formulae (SIGPLAN’94)
- The method computes *generalised sums* which are sums over several variables with complex constraints
Contribution I

Problem  The method requires complex data structures and is not formulated as an algorithm

Solution  By restricting the constraints to be convex polyhedra and to only summarise over polynomials, we are not exposed to the full complexity of the method and can make a straightforward implementation

Drawback  In some cases some precision is lost
Example

Since $n$ affects program flow and is not changed during execution, it is considered an input variable to the program

\[
\begin{align*}
    a_0 &= \top \\
    a_1 &= \{ i = 0 \} \\
    a_2 &= \{ 0 \leq i \leq n + 1 \} \\
    a_3 &= \{ i \geq 0, i \geq n + 1 \} \\
    a_4 &= \{ 0 \leq i \leq n \} \\
    a_5 &= \{ 1 \leq i \leq n + 1 \}
\end{align*}
\]

\[
\begin{align*}
    \Rightarrow \sum_{i \in \mathbb{Z}^2} 1 &= \infty \\
    \Rightarrow \sum_{i \in \{0\}} 1 &= 1 \\
    \Rightarrow n + 2 & \text{ if } n \geq -1 \\
    \Rightarrow \sum_{i \geq 0, n+1} 1 &= \infty \\
    \Rightarrow n + 1 & \text{ if } n \geq 0 \\
    \Rightarrow n + 1 & \text{ if } n \geq 0
\end{align*}
\]
Parametric Integer Programming

- Invented by P. Feautrier 1988
- Essentially a parameterised version of the (dual) simplex algorithm
- There is a tool called PipLib that solves parametric integer problems
- Can be used as parametric IPET calculation
Parametric Integer Programming

- Invented by P. Feautrier 1988
- Essentially a parameterised version of the (dual) simplex algorithm
  - There is a tool called PipLib that solves parametric integer problems
  - Can be used as parametric IPET calculation
Parametric Integer Programming

- Invented by P. Feautrier 1988
- Essentially a parameterised version of the (dual) simplex algorithm
- There is a tool called PipLib that solves parametric integer problems
- Can be used as parametric IPET calculation
Invented by P. Feautrier 1988

Essentially a parameterised version of the (dual) simplex algorithm

There is a tool called PipLib that solves parametric integer problems

Can be used as parametric IPET calculation
This program with assumed WCET of ten clock cycles for each program point gives the following IPET problem:

Maximise $10 \sum_{i=0}^{5} x_i$ subject to

\begin{align*}
x_0 &= 1 \\
x_1 &= x_0 \\
x_2 &= x_1 + x_5 \\
x_3 &= x_3 + x_4 \\
x_3 &= 1 \\
x_5 &= x_4 \\
x_i &\leq p_i
\end{align*}
Example

PipLib Produces (when converted to human readable)

\[
\text{Max} = \lambda(p_0, p_1, p_2, p_3, p_4, p_5).
\]

if \((p_0, p_1, p_2, p_3 \geq 1 \land p_2 < p_4, p_5)\) then

\[
30p_2 + 10 \text{ else if }
\]

\((p_0, p_1, p_2, p_3 \geq 1 \land p_2 < p_4 \land p_5 \geq p_2)\) then \((30p_5 + 40)\) else if ...
Contribution II

**Problem** Parametric ILP has exponential worst-case complexity and produces very large output files.

**Solution** The structural constraint matrix imposes a basis of the unknown variables which can be used in the PILP-problem instead of the variables themselves.
Reducing variables

Most unknowns in the system of linear equations obtained from structural analysis can be expressed as linear combinations of a basis of variables.

By Jordan-Gauss elimination we can obtain this basis.

Solving the IPET problem in terms of this basis reduces the number of unknowns and the general complexity of the problem.

As an example, doing the above on the given program example, we have reduced six unknowns to one unknown.
Reducing variables

- Most unknowns in the system of linear equations obtained from structural analysis can be expressed as linear combinations of a basis of variables.
- By Jordan-Gauss elimination we can obtain this basis.
- Solving the IPET problem in terms of this basis reduces the number of unknowns and the general complexity of the problem.
- As an example, doing the above on the given program example, we have reduced six unknowns to one unknown.
Reducing variables

- Most unknowns in the system of linear equations obtained from structural analysis can be expressed as linear combinations of a basis of variables.
- By Jordan-Gauss elimination we can obtain this basis.
- Solving the IPET problem in terms of this basis reduces the number of unknowns and the general complexity of the problem.
- As an example, doing the above on the given program example, we have reduced six unknowns to one unknown.
Reducing variables

- Most unknowns in the system of linear equations obtained from structural analysis can be expressed as linear combinations of a basis of variables.
- By Jordan-Gauss elimination we can obtain this basis.
- Solving the IPET problem in terms of this basis reduces the number of unknowns and the general complexity of the problem.
- As an example, doing the above on the given program example, we have reduced six unknowns to one unknown.
Combination

From symbolic counting:

\[\begin{align*}
c_0 &= \infty \\
c_1 &= 1 \\
c_2 &= n + 2 \text{ if } n \geq -1 \\
c_3 &= \infty \\
c_4 &= n + 1 \text{ if } n \geq 0 \\
c_5 &= n + 1 \text{ if } n \geq 0
\end{align*}\]

From PipLib:

\[
\text{PWCET} = \lambda n. \\
\text{if}(c_0, c_1, c_2, c_3 \geq 1 \land c_2 < c_4, c_5) \text{ then} \\
30c_2 + 10 \text{ else if} \\
(c_0, c_1, c_2, c_3 \geq 1 \land c_2 < c_4 \land c_5 \geq c_2) \text{ then} (30c_5 + 40) \text{ else if} ...
\]

There is a need for simplification…
Combination

From symbolic counting:

\[
\begin{align*}
    c_0 &= \infty \\
    c_1 &= 1 \\
    c_2 &= n + 2 \text{ if } n \geq -1 \\
    c_3 &= \infty \\
    c_4 &= n + 1 \text{ if } n \geq 0 \\
    c_5 &= n + 1 \text{ if } n \geq 0
\end{align*}
\]

From PipLib:

\[
\begin{align*}
    \text{PWCET} &= \lambda n. \\
    \text{if} (c_0, c_1, c_2, c_3 \geq 1 \land c_2 < c_4, c_5) \text{ then} \\
    30c_2 + 10 \text{ else if} \\
    (c_0, c_1, c_2, c_3 \geq 1 \land c_2 < c_4 \land c_5 \geq c_2) \text{ then} \\
    (30c_5 + 40) \text{ else if} ...
\end{align*}
\]

There is a need for simplification...
Conclusions

- An implementation of a parametric WCET analysis has been made
- We have presented a simplified Pugh’s method to count solutions to presburger formulae to count integer points inside polyhedra
- We have presented a general method to reduce the number of variables in a (parametric) IPET calculation
- There are indications of that computational and representational complexity will be a challenge to deal with
Conclusions

- An implementation of a parametric WCET analysis has been made.
- We have presented a simplified Pugh’s method to count solutions to Presburger formulae to count integer points inside polyhedra.
- We have presented a general method to reduce the number of variables in a (parametric) IPET calculation.
- There are indications of that computational and representational complexity will be a challenge to deal with.
Conclusions

- An implementation of a parametric WCET analysis has been made.
- We have presented a simplified Pugh’s method to count solutions to Presburger formulae to count integer points inside polyhedra.
- We have presented a general method to reduce the number of variables in a (parametric) IPET calculation.
- There are indications of that computational and representational complexity will be a challenge to deal with.
Conclusions

- An implementation of a parametric WCET analysis has been made.
- We have presented a simplified Pugh’s method to count solutions to Presburger formulae to count integer points inside polyhedra.
- We have presented a general method to reduce the number of variables in a (parametric) IPET calculation.
- There are indications of that computational and representational complexity will be a challenge to deal with.
Future Work

- Enhance the input language to be more realistic (include pointers, function calls and arrays)
- Do a full evaluation of the method with benchmarks
- Investigate how simplifications can be made on the final formula and the intermediate results
- Investigate complexity/precision trade-offs in different abstract domains and different approaches of the phases
Future Work

- Enhance the input language to be more realistic (include pointers, function calls and arrays)
- Do a full evaluation of the method with benchmarks
  - Investigate how simplifications can be made on the final formula and the intermediate results
  - Investigate complexity/precision trade-offs in different abstract domains and different approaches of the phases
Future Work

- Enhance the input language to be more realistic (include pointers, function calls and arrays)
- Do a full evaluation of the method with benchmarks
- Investigate how simplifications can be made on the final formula and the intermediate results
- Investigate complexity/precision trade-offs in different abstract domains and different approaches of the phases
Future Work

- Enhance the input language to be more realistic (include pointers, function calls and arrays)
- Do a full evaluation of the method with benchmarks
- Investigate how simplifications can be made on the final formula and the intermediate results
- Investigate complexity/precision trade-offs in different abstract domains and different approaches of the phases
The end

- Thank you!