Precise and Efficient Parametric Path Analysis

Ernst Althaus, Sebastian Altmeyer, Rouven Naujoks

Johannes Gutenberg-Universität Mainz
Saarland University
Max-Planck-Institut für Informatik

LCTES 2011, Chicago

SAARLAND UNIVERSITY
COMPUTER SCIENCE

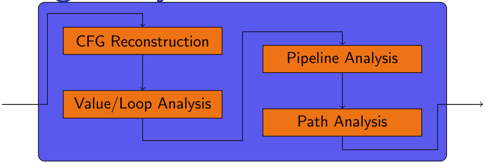
Parametric Timing Analysis – Why?

- timing analysis essential for hard real-time systems
- many systems depend on input parameters
  (operating system schedulers, etc.)

- only two possible solutions:
  1. assume upper bounds on the unknown parameters
     $\Rightarrow$ highly overapproximated execution-time bound
  2. restart the analysis for all parameter assignments
     $\Rightarrow$ very high analysis time

- parametric timing analysis delivers timing formula instead of a numeric value

Parametric Timing Analysis – How?



CFG Reconstruction extracts the control flow graph from the executable.

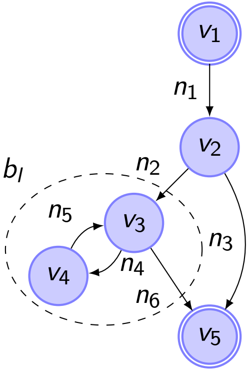
Value/Loop Analysis determines values for registers and memory accesses determines loop bounds and parametric loop bound expressions.

Pipeline Analysis derives bounds on the execution times $T(v_i)$ of all basic blocks.

Path Analysis combines execution times of basic blocks and loop bounds to determine longest execution path.

Framework according to [1].

Path Analysis; Longest Paths via ILP

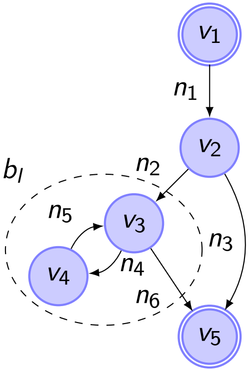


$$\max \sum_i \sum_{n_j \in inc(v_i)} T(v_i) n_j$$

$$\begin{aligned} n_1 &= 1 \\ n_1 &= n_2 + n_3 \\ n_2 + n_5 &= n_4 + n_6 \\ n_4 &= n_5 \\ n_3 + n_6 &= 1 \\ n_4 &<= b_l \cdot n_2 \end{aligned}$$

- *Implicit path enumeration* (IPET [4])
- Control flow graph and the loop bounds are transformed into *flow constraints*.
- Upper bounds for the execution times used as weights.

Parametric Path Analysis; Longest Paths via ILP

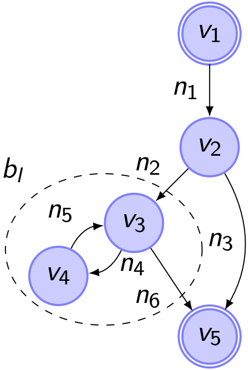


$$\max \sum_i \sum_{n_j \in inc(v_i)} T(v_i) n_j$$

$$\begin{array}{rcl}
n_1 & = & 1 \\
n_1 & = & n_2 + n_3 \\
n_2 + n_5 & = & n_4 + n_6 \\
n_4 & = & n_5 \\
n_3 + n_6 & = & 1 \\
n_4 & <= & \cancel{b_l \cdot n_2} \quad b_l \cdot c
\end{array}$$

- Non-linear inequalities $\Rightarrow$ need for approximation

Parametric Path Analysis; Longest Paths via ILP



$$\max \sum_i \sum_{n_j \in inc(v_i)} T(v_i) n_j$$

$$\begin{aligned}
n_1 &= 1 \\
n_1 &= n_2 + n_3 \\
n_2 + n_5 &= n_4 + n_6 \\
n_4 &= n_5 \\
n_3 + n_6 &= 1 \\
n_4 &<= \cancel{b_l \cdot n_2} \quad b_l \cdot c
\end{aligned}$$

- Non-linear inequalities $\Rightarrow$ need for approximation
- Need to solve an ILP/ parametric PIP [3]
- slow and imprecise in case of parametric ILP

Determing Longest Paths in Control Flow Graphs

Problem is NP-hard in general

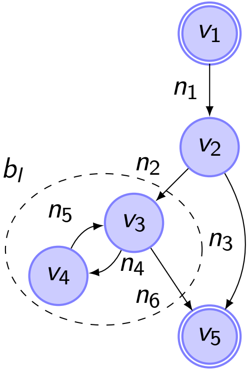
Determing Longest Paths in Control Flow Graphs

Problem is NP-hard in general

But: may be solved efficently for restricted graphs
$\Rightarrow$ Singleton-Loop Model

What is a Singleton Loop?

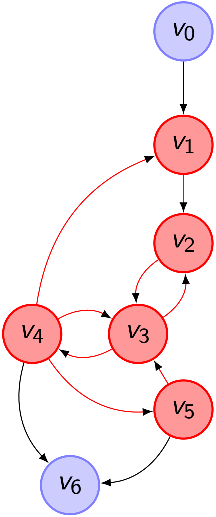
Idea: Code for hard real-time systems often well structured.



- A loop in a CFG is a *strongly connected component* (SCC).
- Structured loops (no Gotos etc.) have a *single entry node*.

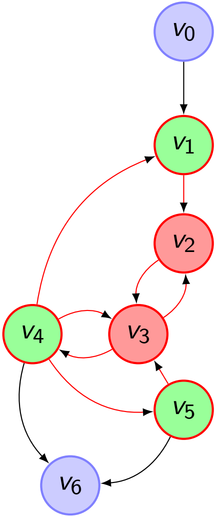
A *singleton loop* is a SCC with exactly one entry node.
A *singleton loop graph* is a CFG that contaisn only singleton loops.

Detailed Explanation



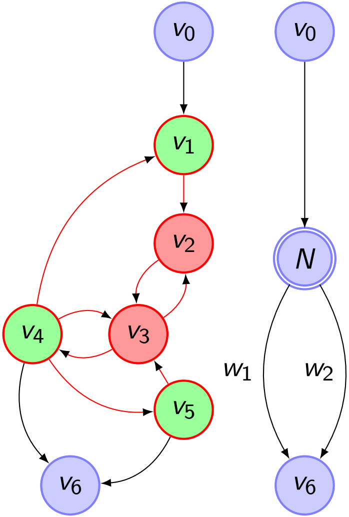
- Assume we know for each loop (by recursion):

Detailed Explanation



- Assume we know for each loop (by recursion):
  - Longest paths from its entry node to its portal nodes.

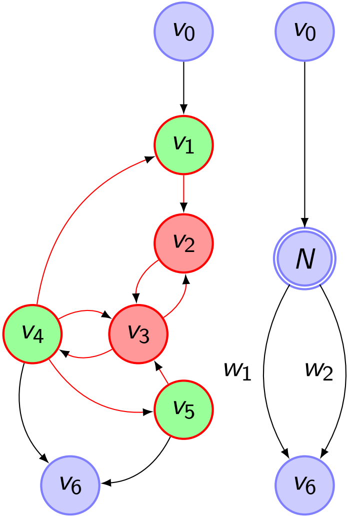
Detailed Explanation



- Assume we know for each loop (by recursion):
  - Longest paths from its entry node to its portal nodes.
- Contract loop to artifical node N.
  - set weight of incident edges appropriately
    $w_1 := lps(v_1, v_4) + w(v_4, v_6)$,
    $w_2 := lps(v_1, v_5) + w(v_5, v_6)$

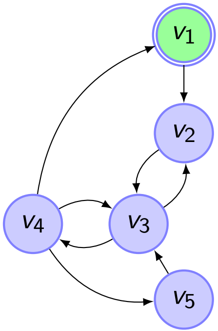
SAARLAND UNIVERSITY COMPUTER SCIENCE

Detailed Explanation



- Assume we know for each loop (by recursion):
  - Longest paths from its entry node to its portal nodes.
- Contract loop to artifical node N.
  - set weight of incident edges appropriately
    $w_1 := lps(v_1, v_4) + w(v_4, v_6)$,
    $w_2 := lps(v_1, v_5) + w(v_5, v_6)$
- Left with a directed acyclic graph.
  - Longest Path Computation in polynomial time.

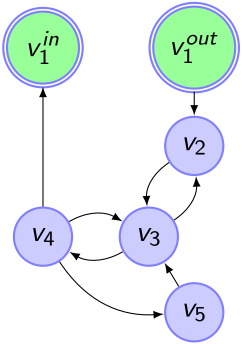
The Singleton-Loop Model - How to recurse



- Given a loop L, with loop bound b_L.
- Recall:
  want to determine LPs from entry node to portal nodes

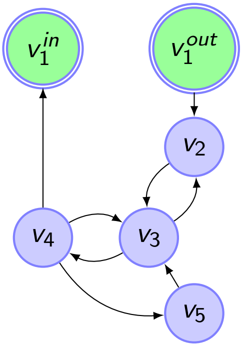
SAARLAND UNIVERSITY COMPUTER SCIENCE

The Singleton-Loop Model - How to recurse



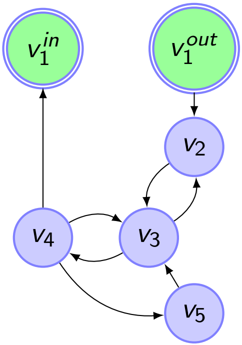
- Given a loop L, with loop bound b_L.
- Recall:
  want to determine LPs from entry node to portal nodes
- Replace entry node v_1 by
  - two nodes v_1^{in}, v_1^{out} with
  - in- and outgoing edges of v_1 assigned accordingly

The Singleton-Loop Model - How to recurse



- Given a loop L, with loop bound b_L.
- Recall:
  want to determine LPs from entry node to portal nodes
- Replace entry node v_1 by
  - two nodes v_1^{in}, v_1^{out} with
  - in- and outgoing edges of v_1 assigned accordingly
- Recurse algorithm on this new graph
  - we know $LP(v_1^{out}, v_i)$ and $LP(v_1^{out}, v_1^{in})$

The Singleton-Loop Model - How to recurse



- Given a loop L, with loop bound b_L.
- Recall:
  want to determine LPs from entry node to portal nodes
- Replace entry node v_1 by
  - two nodes v_1^{in}, v_1^{out} with
  - in- and outgoing edges of v_1 assigned accordingly
- Recurse algorithm on this new graph
  - we know $LP(v_1^{out}, v_i)$ and $LP(v_1^{out}, v_1^{in})$
- $lps(v_1, p_i) := (b_L - 1) \cdot lps(v_1^{out}, v_1^{in}) + lps(v_1^{out}, p_i))$

Runtime Properties

Worst Case Running Times

- numeric bounds: $O(|V||E|)$
- symbolic bounds: $O(|V||E| + |V|^2 \cdot x \cdot s(x))$ where
  - x is the $\#$ of symbolic bounds
  - $s(x)$ is the output size

Output Size

In the worst case:

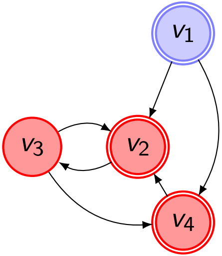
$$2^{2^{x-1}} \leq s(x) \leq 2^{2^x}$$

Output Sensitivity

The algorithm is polynomial output sensitive, i.e. its running time is polynomial in the input size and in the output size.

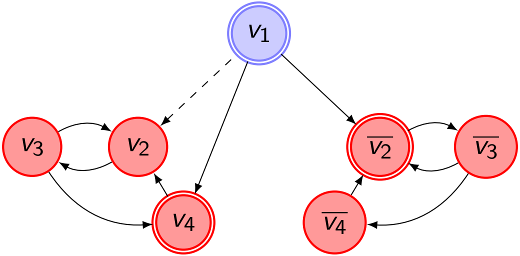
Beyond the Singleton-Loop Model

What happens, if CFG has non-singleton loops?



Beyond the Singleton-Loop Model

What happens, if CFG has non-singleton loops?



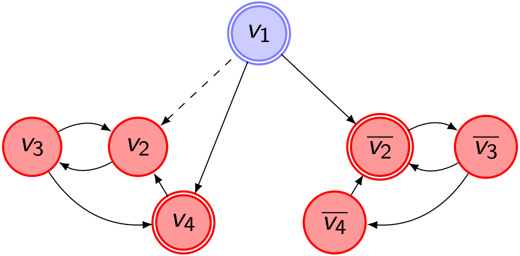
Convert the CFG!

Each CFG can be transformed into an Singleton Loop Graph

SAARLAND UNIVERSITY COMPUTER SCIENCE

Beyond the Singleton-Loop Model

What happens, if CFG has non-singleton loops?



Convert the CFG!

Each CFG can be transformed into an Singleton Loop Graph

Disadvantage: Comes at the cost of increased running time!
(e.g. symbolic bounds can be doubled!)

How useful is the new approach in practise?

Depends on:

- How well does the Singleton-Loop Model fits real CFGs?
- How does the Singleton-Loop Approach perform?
- How much precision is gained?

Testsetting:

- Benchmarks from Mälardalen WCET benchmark suite.
- Compiled via gcc to the $\mathrm{ARM7}$ processor.
- Analyzed on an Intel Core2Duo, 2GHz, 2 GB Ram with Ubuntu 9.10.

Number of Singleton Loops

- Only 8 of 33 test-cases exhibit non-singleton loops (adpcm, cnt, compress, duff, matmult, ndes, ns, qsort-exam).
- Only in one case (*compress*) a higher number of loop-duplications (65) is needed (all others < 10).

Deeply nested loops, unstructured code segments, calls to external library functions causes non-singleton loops.

Compare perfomance to?

Numerica Path Analysis

- ILP Formulation with lp_solve (free lp-solver)
- ILP Formulation with CPLEX
  (commercial lp-solver)

Parametric Path Analysis

- Parametirc ILP Formulation with PIP [3]
  (free parametric lp-solver)
- Parametric timing analysis by Bygde and Lisper [5, 2]
  resorts to C-level, not to binary level, uses a polyhedran approach;
  direct comparison not possible

Performance Evaluation – Test-Cases are ver small

Testcases from Mälardalen WCET benchmark suite are very small
(all are solved in less than 1 second by all approaches)

Name	Size (in Byte) C-File	Size (in Byte) Exec	Singleton Graph	# duplicated loops
s-graph-1	208273	235222	yes	-
s-graph-2	468944	292305	yes	-
s-graph-3	702670	386961	yes	-
s-graph-4	936396	481609	yes	-
s-graph-5	670452	284593	yes	-
ns-graph-1	90274	215433	no	77
ns-graph-2	315562	247443	no	77
ns-graph-3	766144	426427	no	77
ns-graph-4	990502	520579	no	5
ns-graph-5	979908	518338	no	9
ns-graph-6	942084	502580	no	74

Larger benchmarks created by combining and duplicating original test-cases from the benchmark suite

(s-graph-X are singleton loop graphs, ns-graph-X non-singleton loop graphs)

Performance Evaluation, Numeric

Name	Runtime (s)		
	SingletonLoop	lp_solve	CPLEX
nsichneu	0.02	0.86	0.05
s-graph-1	0.03	3.46	0.08
s-graph-2	0.05	13.69	0.08
s-graph-3	0.08	30.85	0.11
s-graph-4	0.11	57.31	0.18
s-graph-5	0.11	108.8	0.13
adpcm	0.04	0.07	0.02
compress	0.3	0.03	0.03
statemate	0.05	0.3	0.04
ns-graph-1	0.97	4.5	0.04
ns-graph-2	0.95	14.58	0.05
ns-graph-3	1.01	48.13	0.12
ns-graph-4	0.14	92.1	0.11
ns-graph-5	0.16	113.3	0.12
ns-graph-6	0.64	65.9	0.17

Performance Evaluation, Parametric

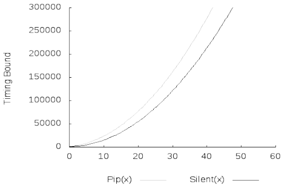
Name	Runtime # of parameters						
	0	1	2	3	4	6	8
s-graph-1	0.03	0.03	0.03	0.03	0.03	-	-
s-graph-2	0.05	0.05	0.05	0.05	0.05	0.05	0.06
s-graph-3	0.08	0.08	0.08	0.08	0.08	0.08	0.11
s-graph-4	0.11	0.11	0.11	0.11	0.12	0.12	0.12
s-graph-5	0.11	0.12	0.12	0.12	0.12	0.12	0.13
ns-graph-1	0.97	1	1.73	1.9	2.02	2.11	2.11
ns-graph-2	0.95	2.03	2.03	2.04	2.36	2.35	2.38
ns-graph-3	1.01	1.01	1.01	1.01	1.24	3.42	3.44
ns-graph-4	0.14	0.22	0.28	0.3	0.33	0.45	0.45
ns-graph-5	0.16	0.28	0.33	0.62	0.67	1.11	1.11
ns-graph-6	0.64	0.64	0.64	0.66	0.71	1.19	1.19

measurements only of singleton loop method
all other approaches fail to solve these test-cases
(PIP and Bygde's approach [2] handle at most two parameters)

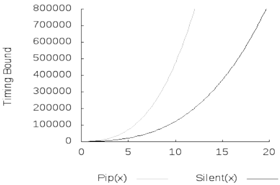
SAARLAND UNIVERSITY
COMPUTER SCIENCE

Evaluation: Precision of the Parametric Formulas

Insertsort



Matmult



$$Time_{PIP}(n) = 156n^2 + 674n + 1186$$

$$Time_{Singleton}(n) = 131n^2 + 71n + 1185$$

$$Time_{PIP}(n) = \begin{cases} 386n^3 + 782n^2 + 790n + 643 & \text{if } n > 1 \\ 2992 & \text{if } n \leq 1 \end{cases}$$

$$Time_{Singleton}(n) = 111n^3 + 164n^2 + 845n + 793$$

- Singleton Loop Method is precise
- PIP suffers fro imprecision due to loop bound transformation.
- Bygde's approach is precise in most, but not in all cases

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Conclusions

Singleton Loop Graphs are restricted CFG that enable computation of

- numeric timing bound in polynomial time,
- parametric timing bound in output-polynomial time (significant improvment over former methods), and
- precise parametric timing bounds.

All CFGs can be transformed to singleton loop graphs
(at the cost of performance loss).

Evaluation showed that

- most benchmarks fit the singleton loop model,
- singleton loop approach can compete with CPLEX,
- enable fast and precise computation of parametric timing bounds.

Thanks for your attention.

References

S. Altmeyer, C. Hümbert, B. Lisper, and R. Wilhelm.
Parametric timing analyis for complex architectures.
In *RTCSA'08*, 2008.

S. Bygde, A. Ermedahl, and B. Lisper.
An efficient algorithm for parametric wcet calculation.
In *RTCSA'09*, 2009.

P. Feautrier.
The parametric integer programming's home `http:\www.piplib.org`.

Y.-T. S. Li and S. Malik.
Performance analysis of embedded software using implicit path enumeration.
In *DAC '95*.

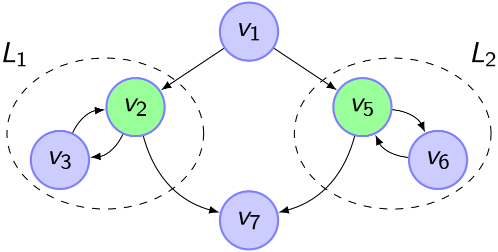
B. Lisper.
Fully automatic, parametric worst-case execution time analysis.
In *(WCET 03)*.

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Appendix: Mälardalen Benachmark Suite

Name	Size (in Byte) C-File	Exec	Singleton Graph	# duplicated loops
adpcm	26582	156759	no	5
bs	4248	144447	yes	-
bs100	2779	144629	yes	-
cnt	2880	149801	yes	-
compress	13411	149804	no	65
cover	5026	148301	yes	-
crc	5168	145615	yes	-
duff	2374	144739	no	6
edn	10563	150682	yes	-
expint	4288	145867	yes	-
fac	426	144148	yes	-
fdct	8863	147128	yes	-
fft1	6244	153303	yes	-
fibcall	3499	144152	yes	-
fir	11965	151589	yes	-
insertsort	3892	144305	yes	-
jannecomplex	1564	144242	yes	-
jfdctint	16028	146858	yes	-
lcdnum	1678	144509	yes	-
lms	7720	157868	yes	-
ludpcm	5160	151848	yes	-
matmult	3737	145083	no	4
minver	5805	152845	yes	-
ndes	7345	148689	no	2
ns	10436	149567	no	7
nsichneu	118351	176240	yes	-
prime	904	144538	yes	-
qsort-exam	4535	146468	no	3
qurt	4898	151214	yes	-
recursion	620	144341	yes	-
select	4494	146283	yes	-
sqrt	3567	154282	yes	-
statemate	52618	162870	no	7

Appendix: Imprecision of Parametric ILP Approach



- Only one loop taken in actual execution
- parametric ILP needs to upper bound entry node: both loops are part of the WCET Path