Tighter Timing Predictions by Automatic Detection and Exploitation of Value-Dependent Constraints

Christopher A. Healy
David B. Whalley

Florida State University

The Problem

- We need to accurately bound the WCET of a real-time program.
- Even with perfect architectural modeling, data dependencies influence branch outcomes and which paths are taken.
- Entering path constraint information manually is tedious and error prone.
- Our solution is to automatically detect constraints, and exploit this information in timing analysis.

Outline

- Detecting constraints on branches
- Creating path constraints
- Using path constraints in loop analysis
- Results
- Conclusions

Detecting Constraints on Branches

- Detect registers and variables on which each branch depends.
- Detect effects of each block on each branch.
- Two types of constraints:
  - — effect-based constraints
  - — iteration-based constraints

Expanding a Comparison

Instructions in a Basic Block

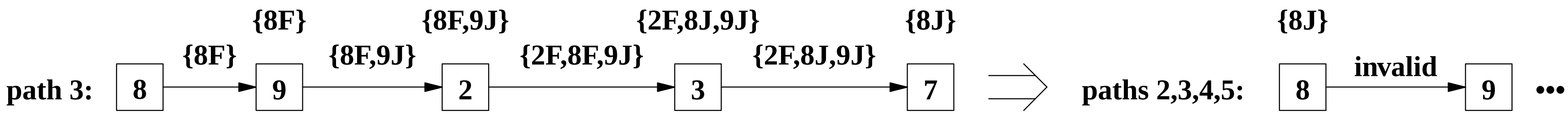
```
r[1]=HI[_g];              /* sethi %hi(_g),%g1          */
r[8]=R[r[1]+LO[_g]];      /* ld    [%g1+%lo(_g)],%o0    */
IC=r[8]?5;                /* cmp   %o0,5                */
PC=IC<0,L20;              /* bl    L20                  */
```

Expanded Comparison

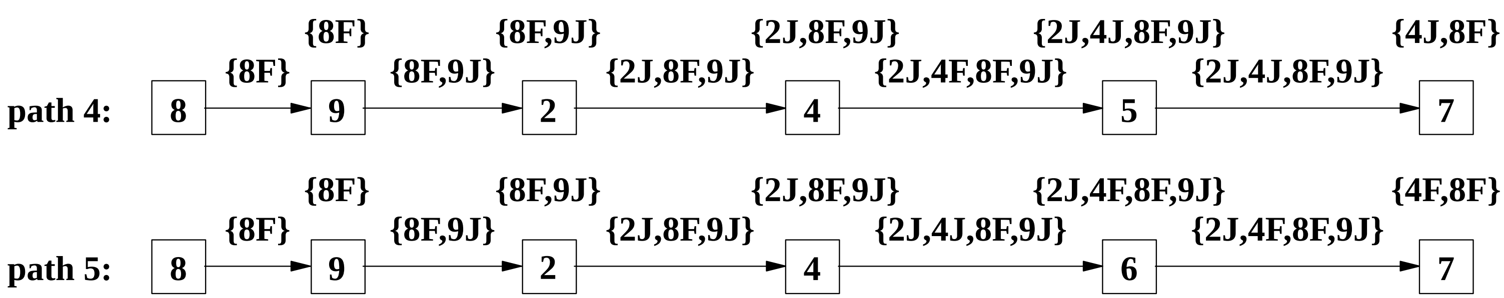
```
IC=R[HI[_g]+LO[_g]]?5;
```

- Compiler determines how register r[8] gets its value prior to the comparison.
- This comparison ultimately depends on **`g`**.

Propagating Dependency Information

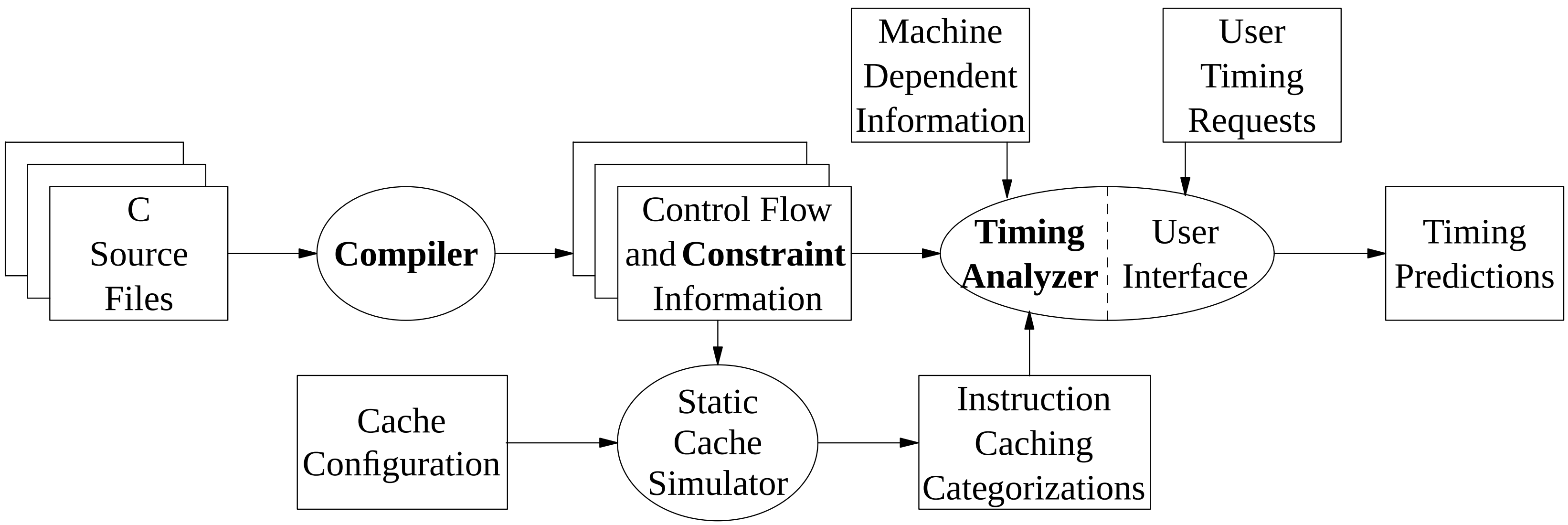


Only Path 1 can Follow Path 3



Paths 4 and 5 cannot follow themselves

Overview of the Timing Analysis Process



Path Information for First Example

Total Iters	Path ID	Exit Path	Possible Iterations	Maximum Iterations
1,001	1	Y	[1001..1001]	1
	2	Y	[1001..1001]	1
	3	N	[1000..1000]	1
	4	N	[2..1000]	500
	5	N	[1..1000]	500

Can Follow Matrix

- A *Can Follow* matrix is constructed to indicate for each path the set of paths that can legally follow it on the next iteration.
- This is used to constrain the number of iterations in which a path is allowed to execute.
- Can Follow Matrix for Previous Example

Current Path in Loop	Paths That Can Immediately Follow				
	1	2	3	4	5
1	N	N	N	N	N
2	N	N	N	N	N
3	Y	N	N	N	N
4	N	Y	Y	N	Y
5	N	Y	Y	Y	N

Worst-Case Loop Example

Iteration	P 1	P 2	P 3	P 4	P 5	Longest	Time
1	16	28	44	**56**	54	4	56
2	7	10	17	**20**	18	4	72
3-500	*7*	*10*	*17*	**20**	*18*	4	8040
501	7	10	17		**18**	5	8054
502-1000	*7*	*10*	*17*		**18**	5	15040
1001	7	**10**				2	15046

Test Programs

Name	Description
Expint	Computes exponential integral
Frenel	Computes noncomplex Fresnel integral
Gaujac	Gauss-Jacobi abscissas, weights
Sprsin	Convert matrix to sparse storage
Summidall	Sum middle half and all elements of array
Summinmax	Sum min and max of corresponding elements
Sumnegpos	Sum neg, pos and all elements in array
Sumoddeven	Sum odd and even elements in array

Results of Timing Predictions

Name	WCET Timing Prediction Results				
	Observed	Value Independent		Value Dependent	
	Cycles	Estimated Cycles	Estim. Ratio	Estim. Cycles	Estim. Ratio
Expint	58,397	1,292,086	22.126	58,471	1.001
Frenel	47,749	48,887	1.029	47,783	1.001
Gaujac	786,386	797,116	1.014	794,334	1.010
Sprsin	28,339	28,608	1.009	28,404	1.002
Summidall	15,340	18,090	1.179	15,341	1.000
Summinmax	16,080	17,080	1.062	16,080	1.000
Sumnegpos	11,067	13,068	1.181	11,068	1.000
Sumoddeven	15,093	16,102	1.067	15,099	1.000
Average	122,306	278,880	3.708	123,323	1.002

Response Time of Timing Analyzer

Name	Seconds Required for Analysis		
	Previous Analysis Time	Current Analysis Time	Time Ratio
Expint	0.382	0.300	0.785
Frenel	0.322	0.272	0.845
Gaujac	2.737	1.845	0.674
Sprsin	0.107	0.113	1.056
Summidall	0.060	0.052	0.867
Summinmax	0.067	0.050	1.034
Sumnegpos	0.050	0.037	0.746
Sumoddeven	0.038	0.038	1.000
Average	0.470	0.338	0.876

Future Work

- Predict best-case execution time.
- Perform interprocedural analysis to detect more constraints.

Conclusions

- Compiler detects constraints on branches.
- Path constraints are generated by propagating constraints through paths.
- Timing analyzer bounds number of iterations associated with each path for loop analysis.
- The result is significantly tighter WCET predictions.
- The approach is fully automated and efficient.