

Traceability of Flow Information: Reconciling Compiler Optimizations and WCET Estimation

Hanbing LI, Inria/IRISA

Isabelle Puaut, University of Rennes 1/IRISA

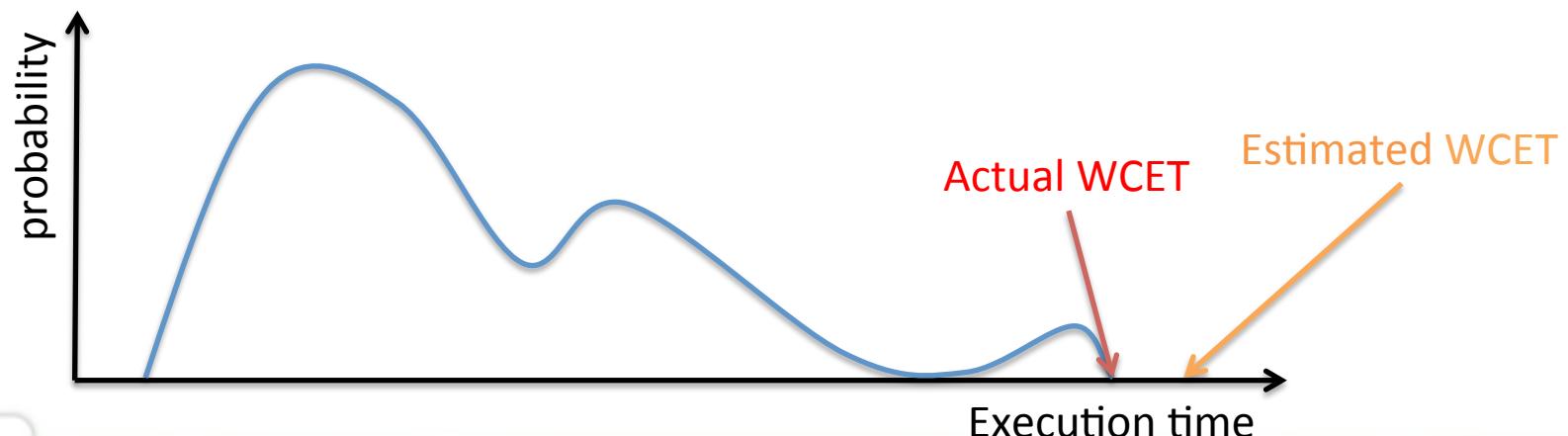
Erven Rohou, Inria/IRISA

RTNS 2014 , Versailles, France

Funded by W-SEPT, L'Agence nationale de la recherche

Context

- WCET: Worst-Case Execution Time
 - demonstrate that the system meets its timing constraints
 - computed at machine code level
 - Need the timing of processor operations
 - safe and as tight as possible



Context

- Flow information: Information on possible flows of control
 - Automatically
 - Manually at source code level
 - help tighten WCETs
 - Example:
 - loop bound information
 - infeasible path

Compiler optimizations

- Located in the middle of source code and machine code
- Deliver more performance
- Modify the structure of code

Example of optimizations

Loop unrolling: replicate the body of the loop in one iteration according to the unrolling factor

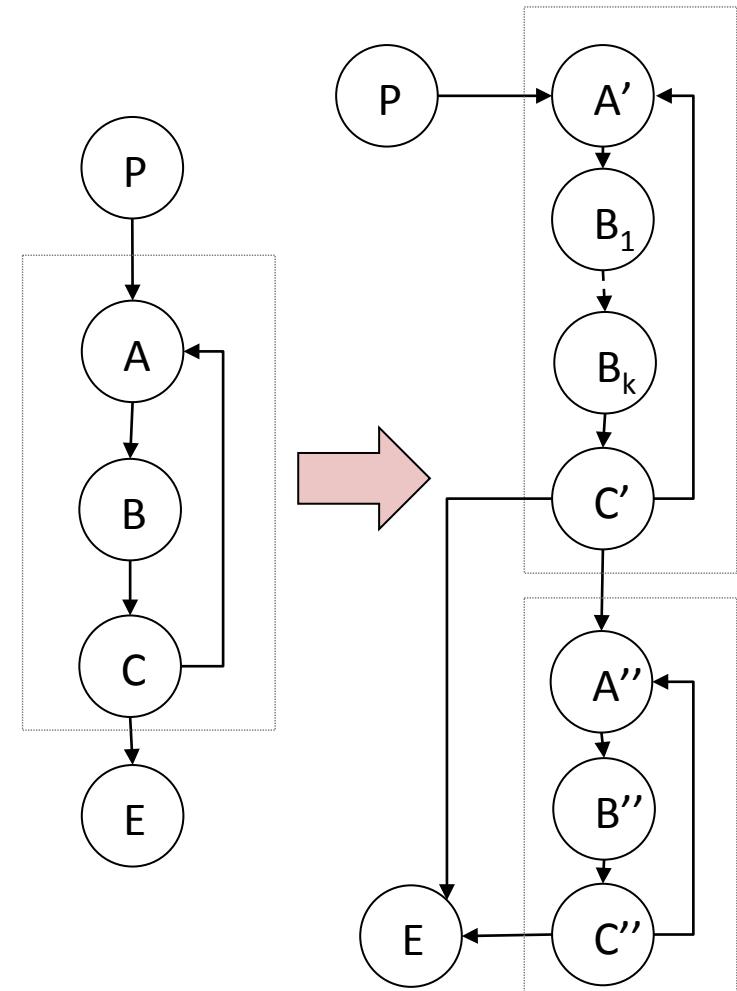
Advantages: reduce loop overhead and increase instruction parallelism

Original code

```
for(i=0; i<X; i++)  
{  
    body(i);  
}
```

Optimized code

```
for(i=0; i<X; i+=k) {  
    body(i);  
    body(i+1);  
    ...  
    body(i+k-1);  
}  
for(; i<X; i++) {  
    body(i);  
}
```

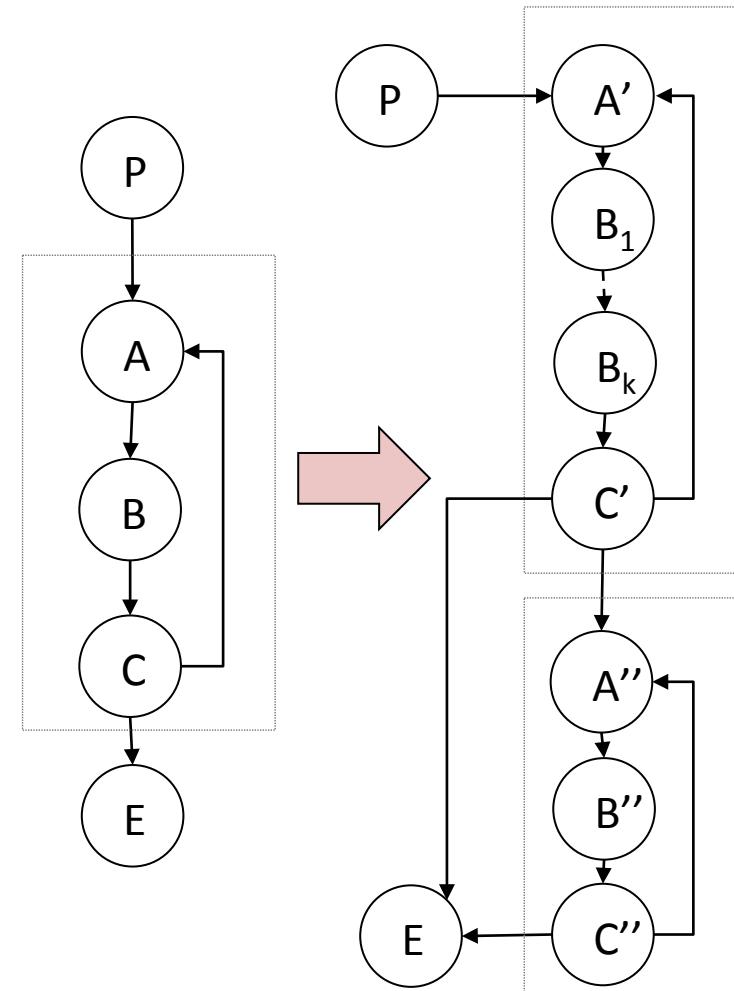


Example of optimizations

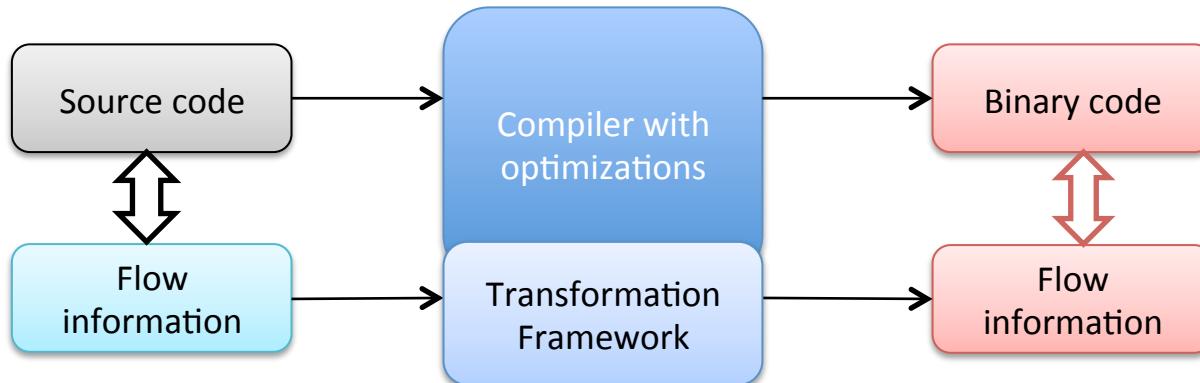
- Difficult to match the structure of source code and machine code
- Changes of flow information

```
for(i=0; i<X; i++)  
{  
    body(i);  
}
```

```
for(i=0; i<X; i+=k) {  
    body(i);  
    body(i+1);  
    ...  
    body(i+k-1);  
}  
For(; i<X; i++) {  
    body(i);  
}
```



Contribution



- Propose a framework to transform flow information from source code level to binary code level
- Implement the traceability of local loop bound within a modern optimizing compiler
- Show the impact of optimizations on WCET

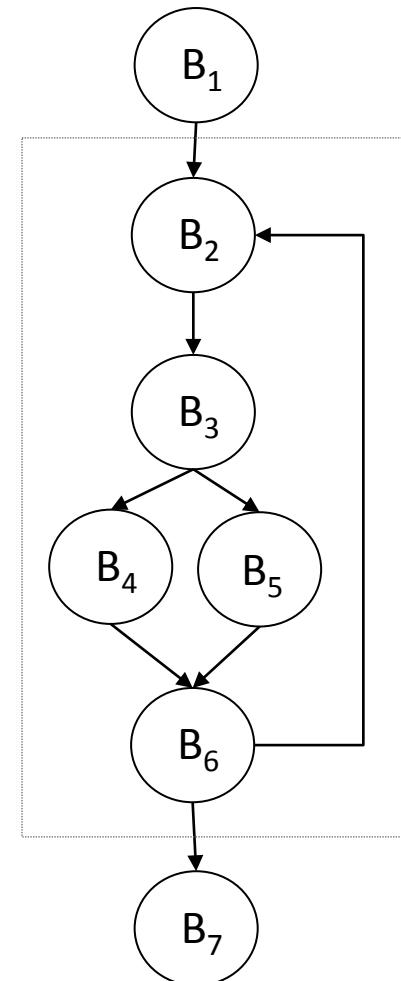
Outline

- WCET Calculation Method
- Transformation Framework
- Supported Optimizations
- Case Study: Loop Unrolling
- Implementation and Experimental Results
- Conclusion and Future Work

WCET calculation using IPET

- IPET: Implicit path enumeration technique
- This method operates on CFG, extracted from binary code

```
for(i=0; i<n; i++) {  
    a[i]=b[i]+c[i];  
    if(a[i]>m) {  
        branch1;  
    }  
    else {  
        branch2;  
    }  
}
```



IPET Method

Objective function

$$\sum_{i \in CFG} f_i \times T_i$$

Structural constraints

$$f_1 = 1$$

$$f_{12} + f_{62} = f_{23} = f_2$$

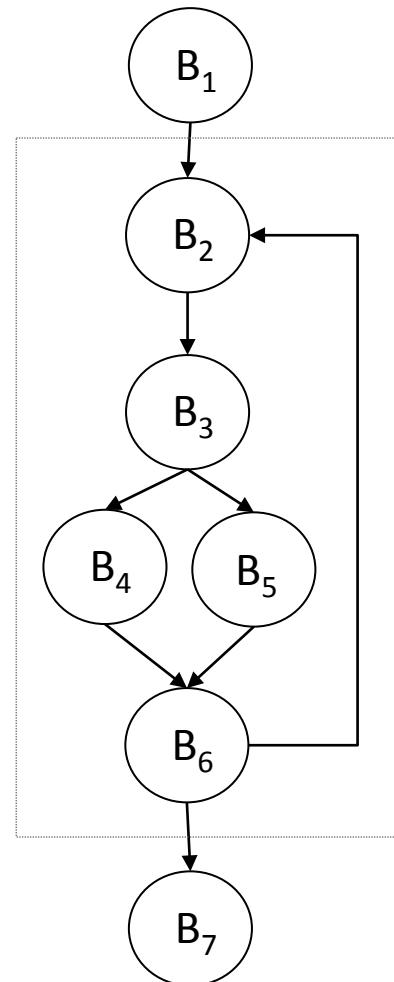
$$f_{23} = f_{34} + f_{35} = f_3$$

$$f_{34} = f_{46} = f_4$$

$$f_{35} = f_{56} = f_5$$

$$f_{46} + f_{56} = f_{62} + f_{67} = f_6$$

- Extracted from the structure of the CFG automatically



IPET Method

Objective function

$$\sum_{i \in CFG} f_i \times T_i$$

Structural constraints

$$f_1 = 1$$

$$f_{12} + f_{62} = f_{23} = f_2$$

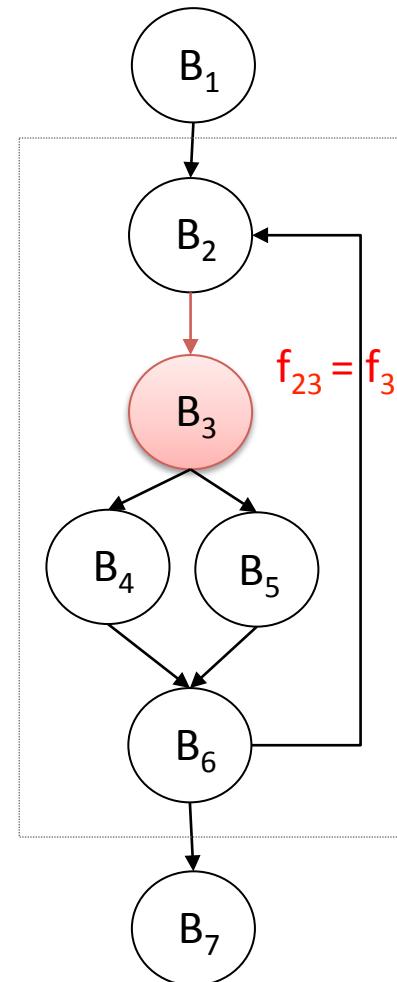
$$f_{23} = f_{34} + f_{35} = f_3$$

$$f_{34} = f_{46} = f_4$$

$$f_{35} = f_{56} = f_5$$

$$f_{46} + f_{56} = f_{62} + f_{67} = f_6$$

- Extracted from the structure of the CFG automatically



IPET Method

Objective function

$$\sum_{i \in CFG} f_i \times T_i$$

Structural constraints

$$f_1 = 1$$

$$f_{12} + f_{62} = f_{23} = f_2$$

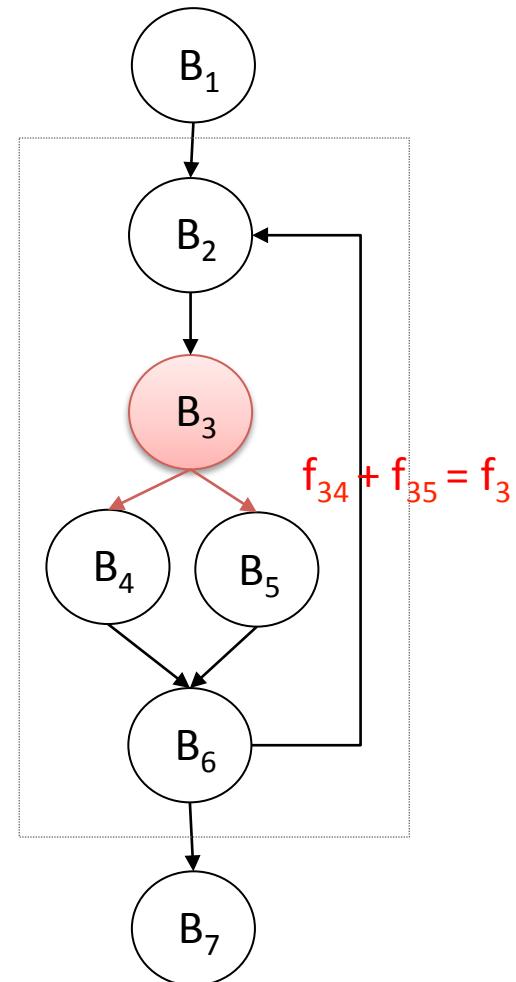
$$f_{23} = f_{34} + f_{35} = f_3$$

$$f_{34} = f_{46} = f_4$$

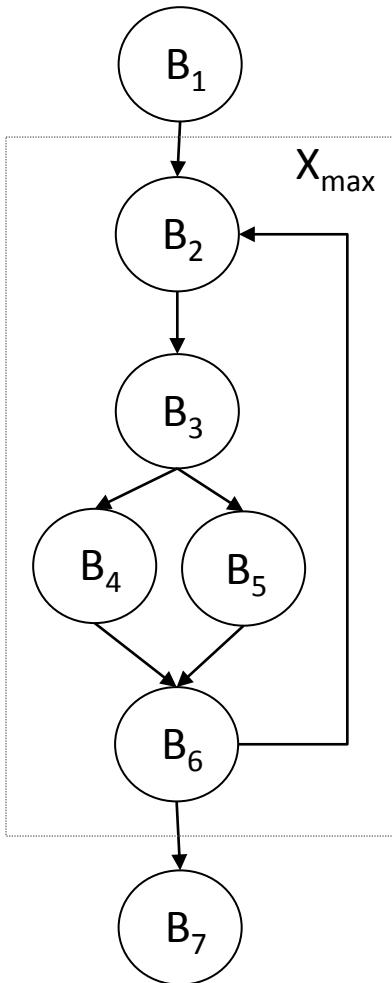
$$f_{35} = f_{56} = f_5$$

$$f_{46} + f_{56} = f_{62} + f_{67} = f_6$$

- Extracted from the structure of the CFG automatically



Flow information for IPET



Flow Information

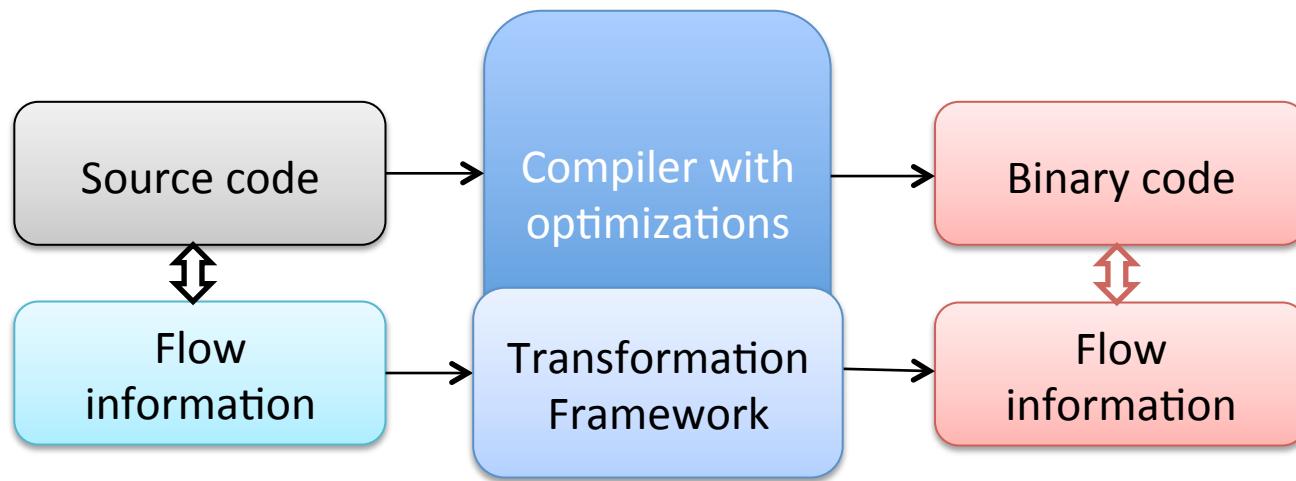
$$f_6 \leq X_{max}$$

$$f_4 \leq 2 \times f_5$$

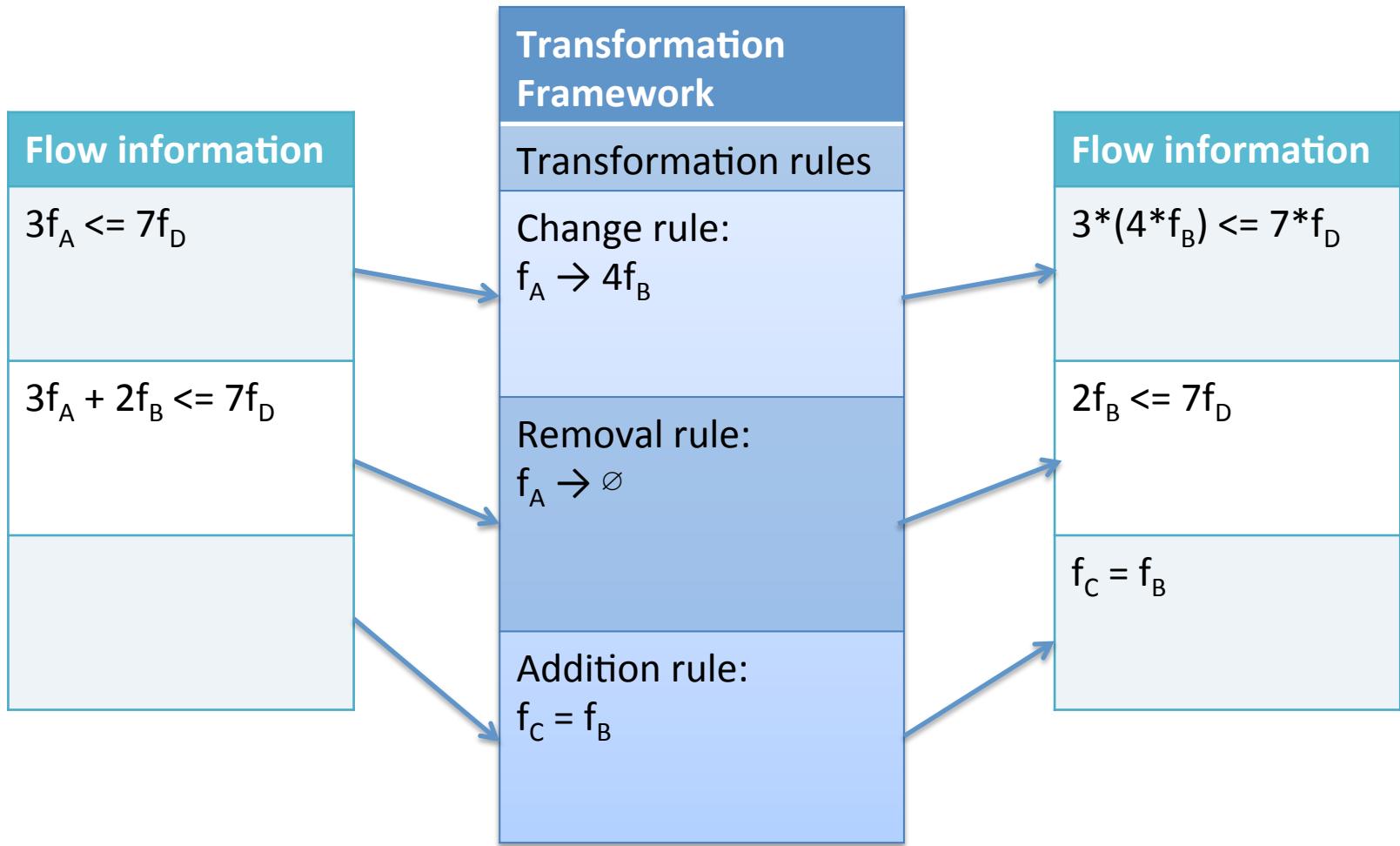
- specify flow information that cannot be obtained directly from the control flow graph
- inserted manually by the programmer
- obtained automatically using static analysis methods

Transformation Framework

- Transformation framework conveys flow information from source code level to machine code level



Transformation Rules

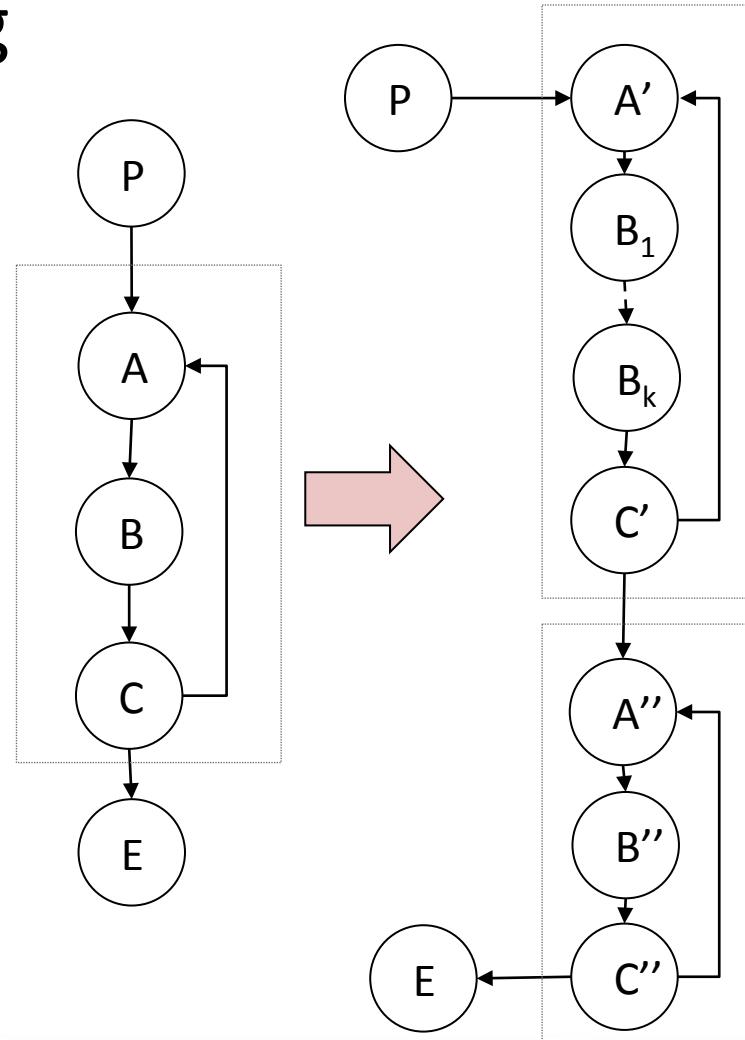


Supported compiler optimizations

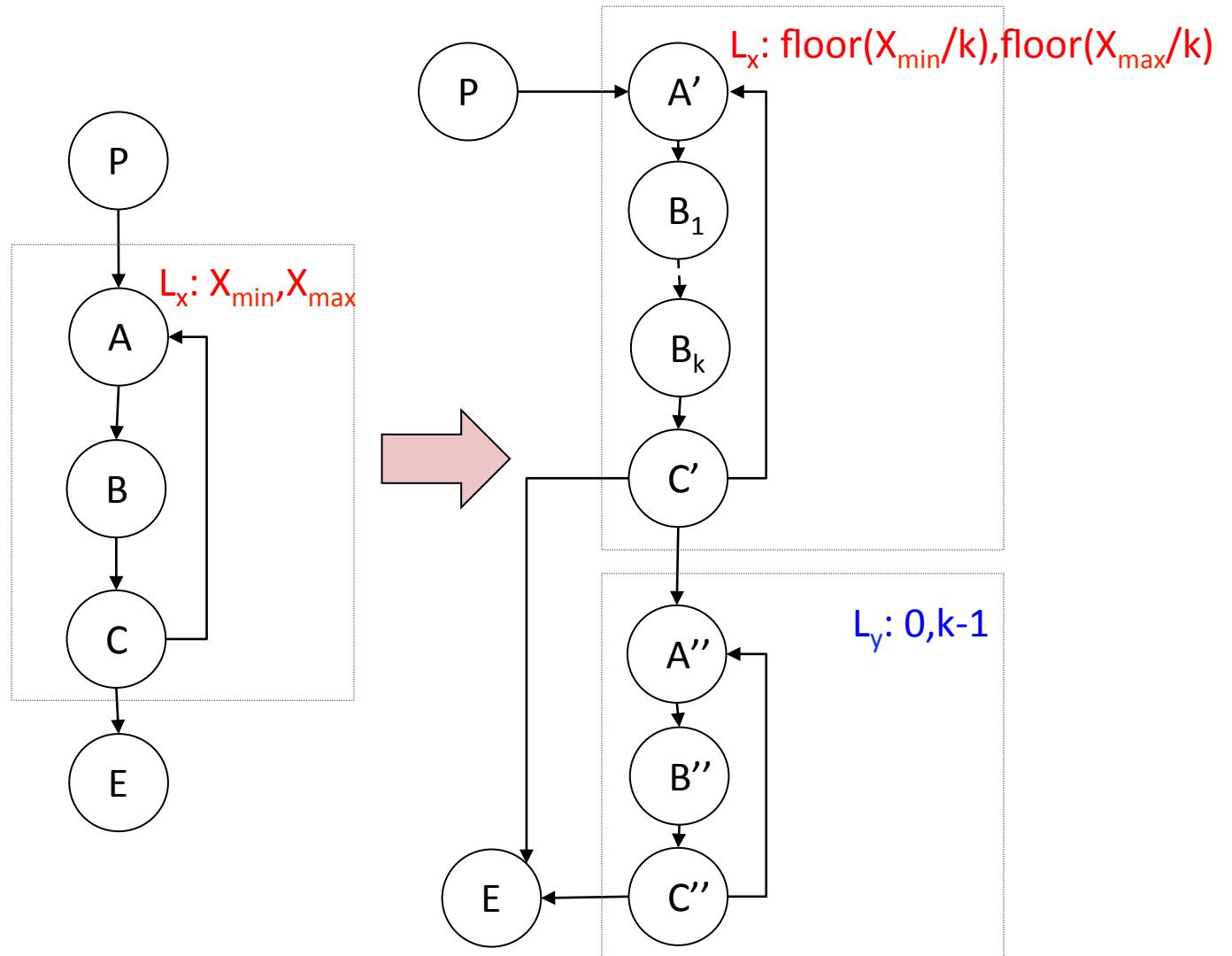
- Support most general optimizations in compilers
 - Redundancy elimination, control-flow and low-level optimizations
 - adce, correlated propagation, deadargelim, dse, early-cse, functionattrs, globalopt, ipsccp, jump-threading, mem2reg, sroa ...
 - Loop optimizations
 - loop-simplify, lcssa, licm, loop-unswitch, indvars, loop-idiom, loop-deletion, loop rotation, loop-unroll, , loop interchange, loop fission, loop fusion

Case Study

- Loop Unrolling



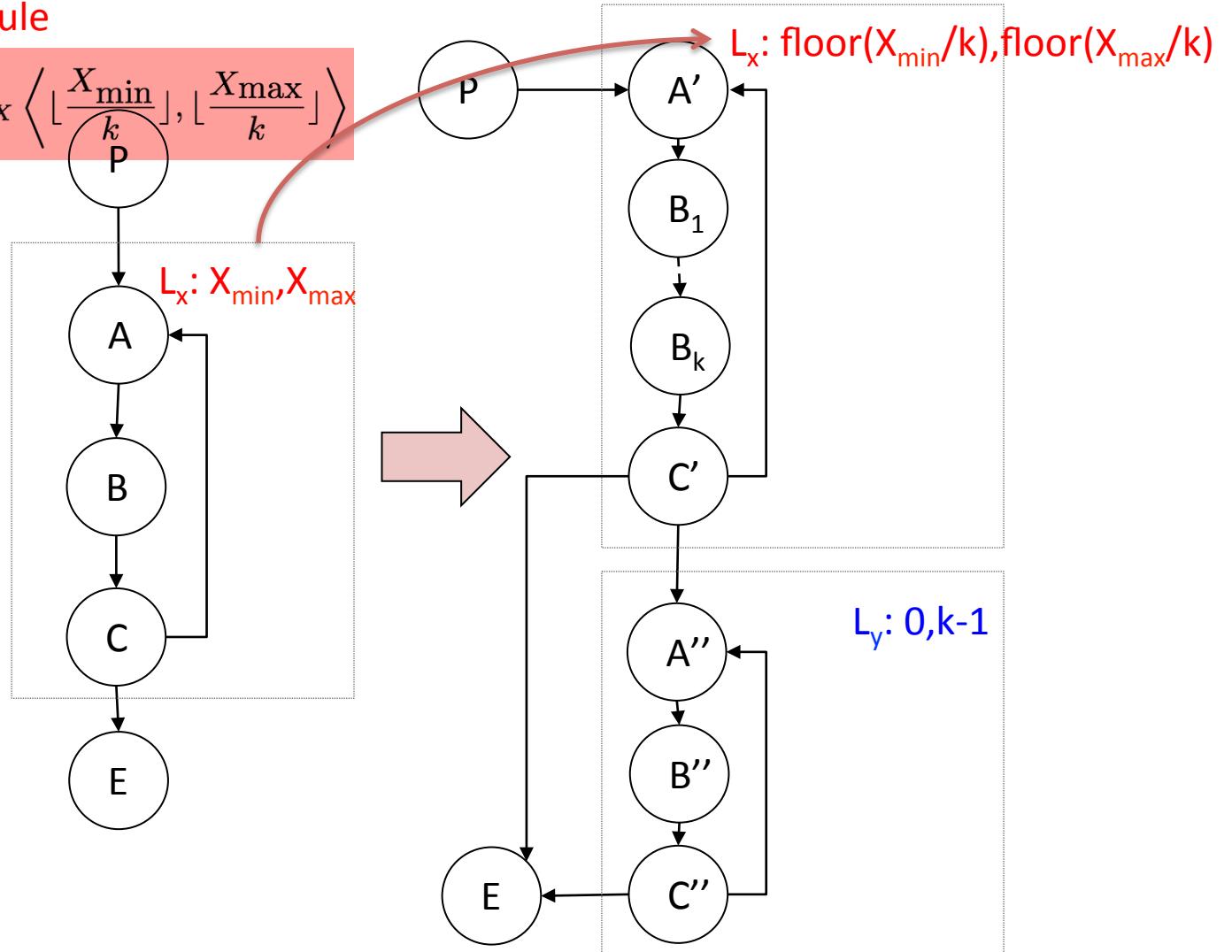
Loop Unrolling



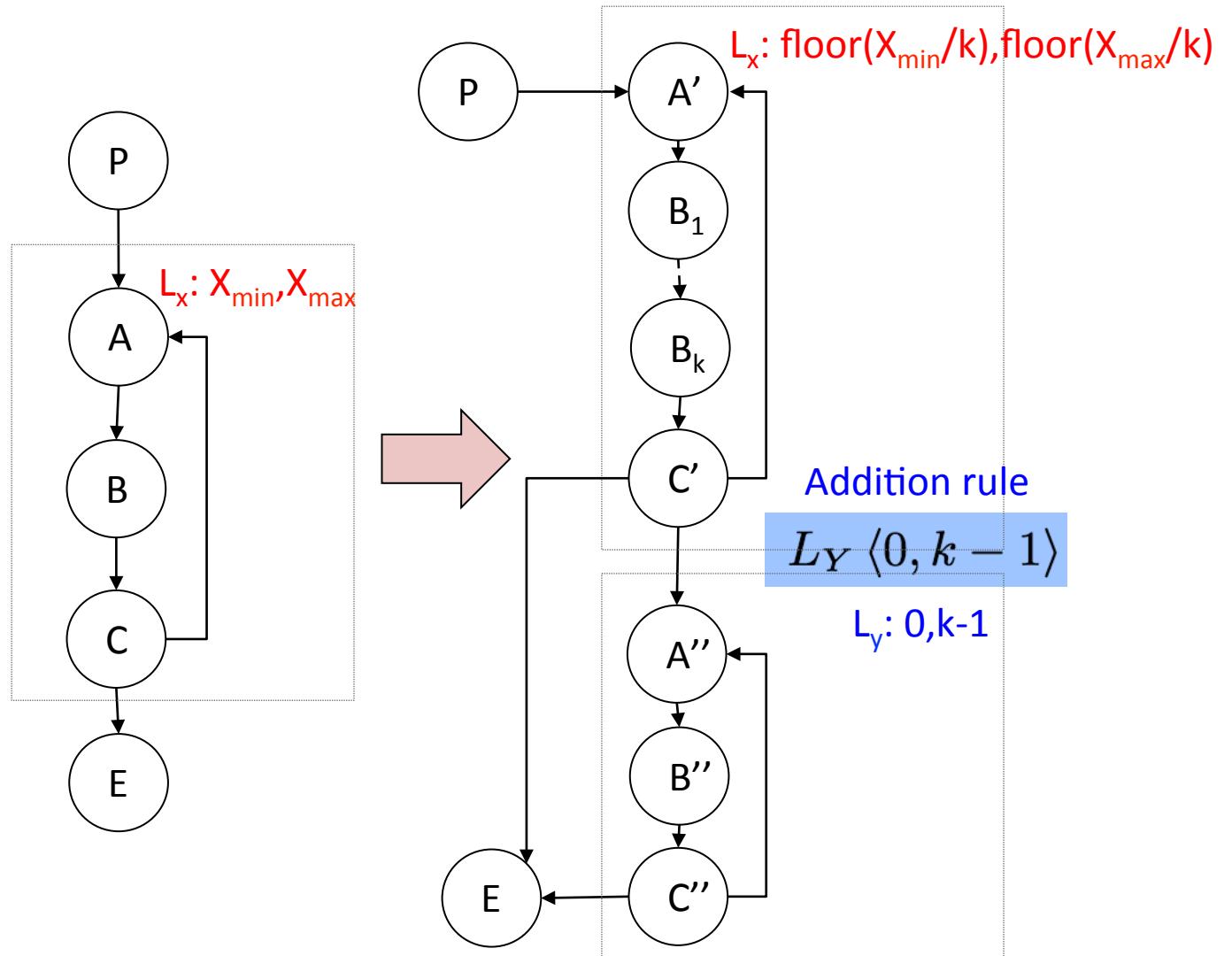
Loop Unrolling

Change rule

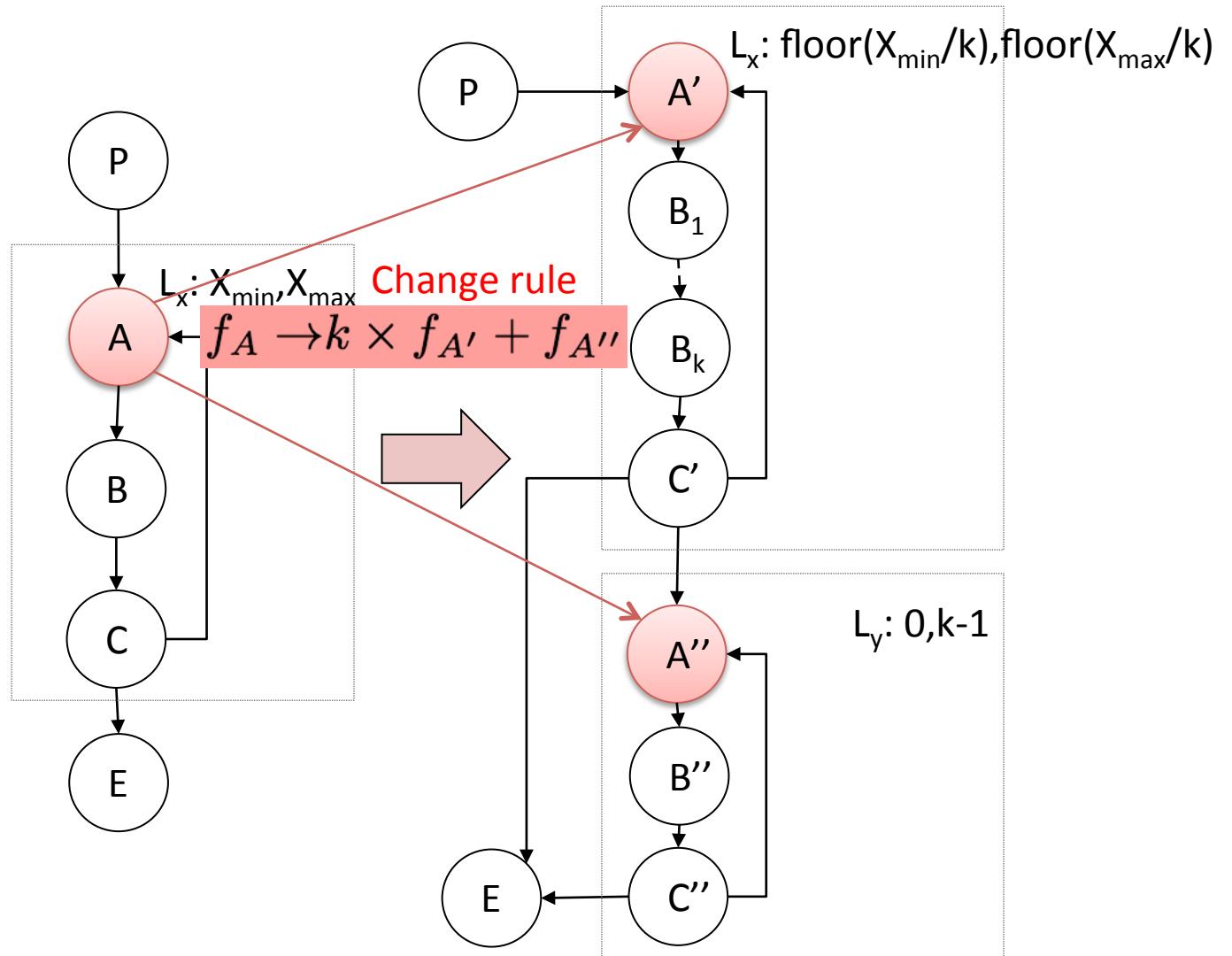
$$L_X \langle X_{\min}, X_{\max} \rangle \rightarrow L_X \left\langle \lfloor \frac{X_{\min}}{k} \rfloor, \lfloor \frac{X_{\max}}{k} \rfloor \right\rangle$$



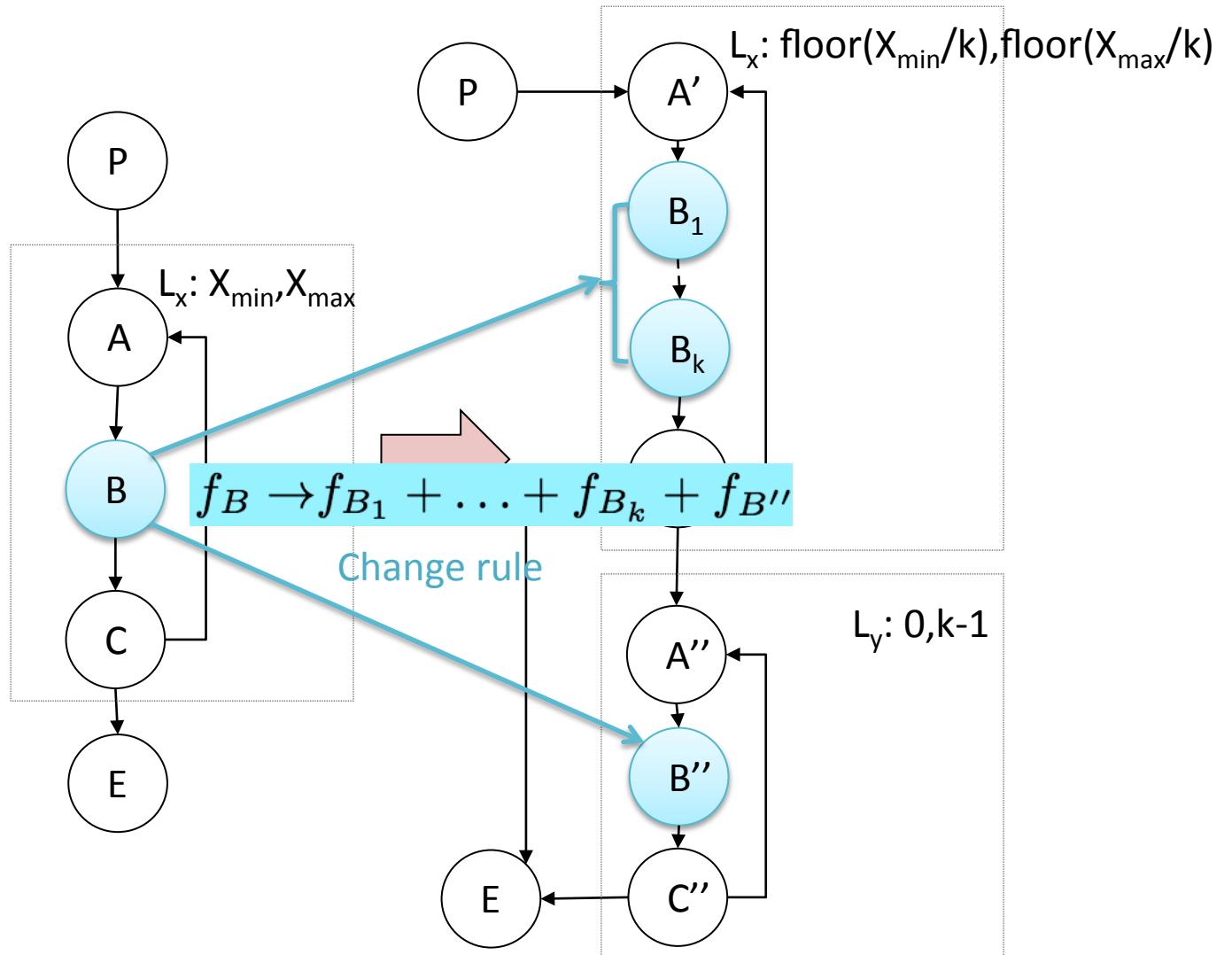
Loop Unrolling



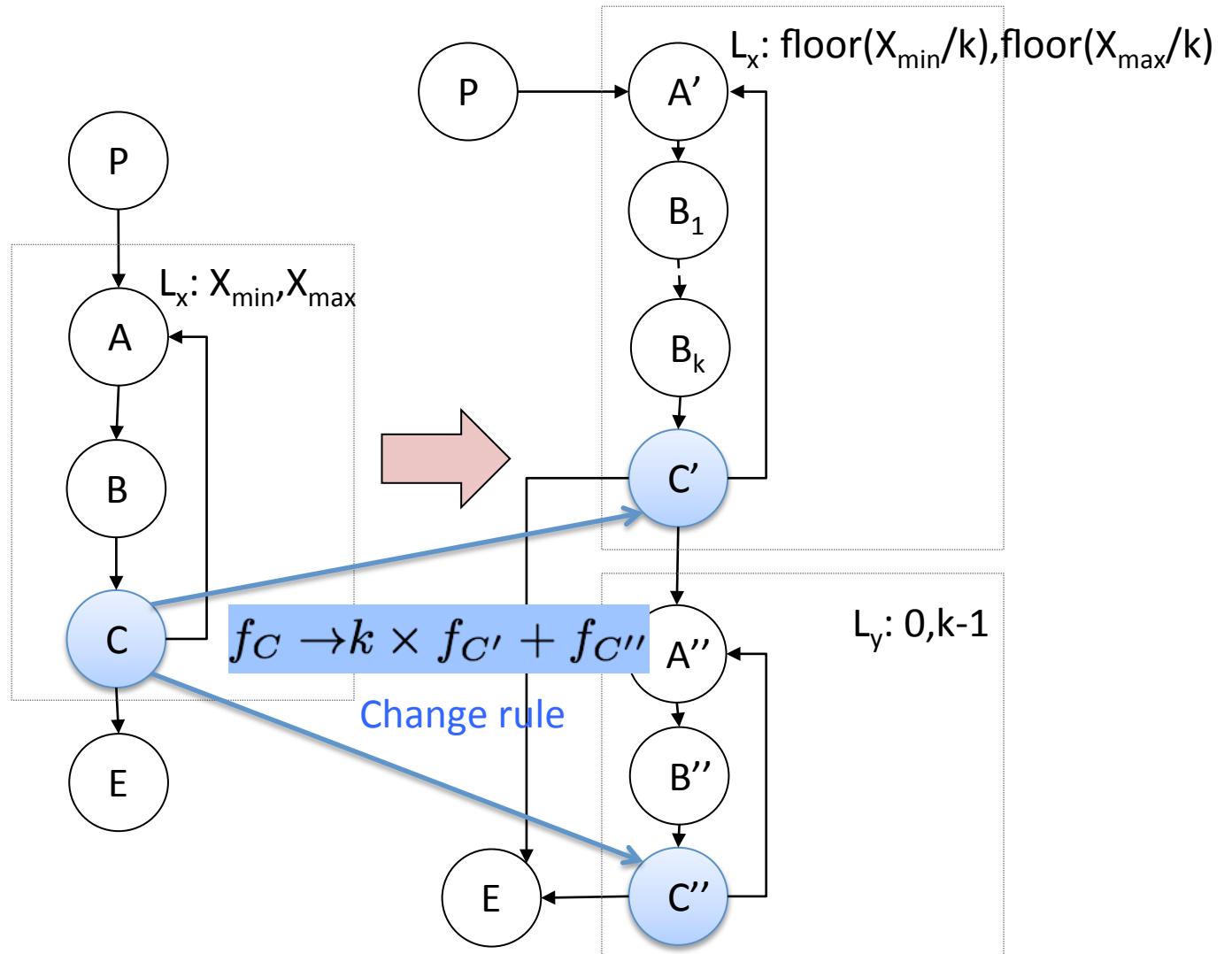
Loop Unrolling



Loop Unrolling

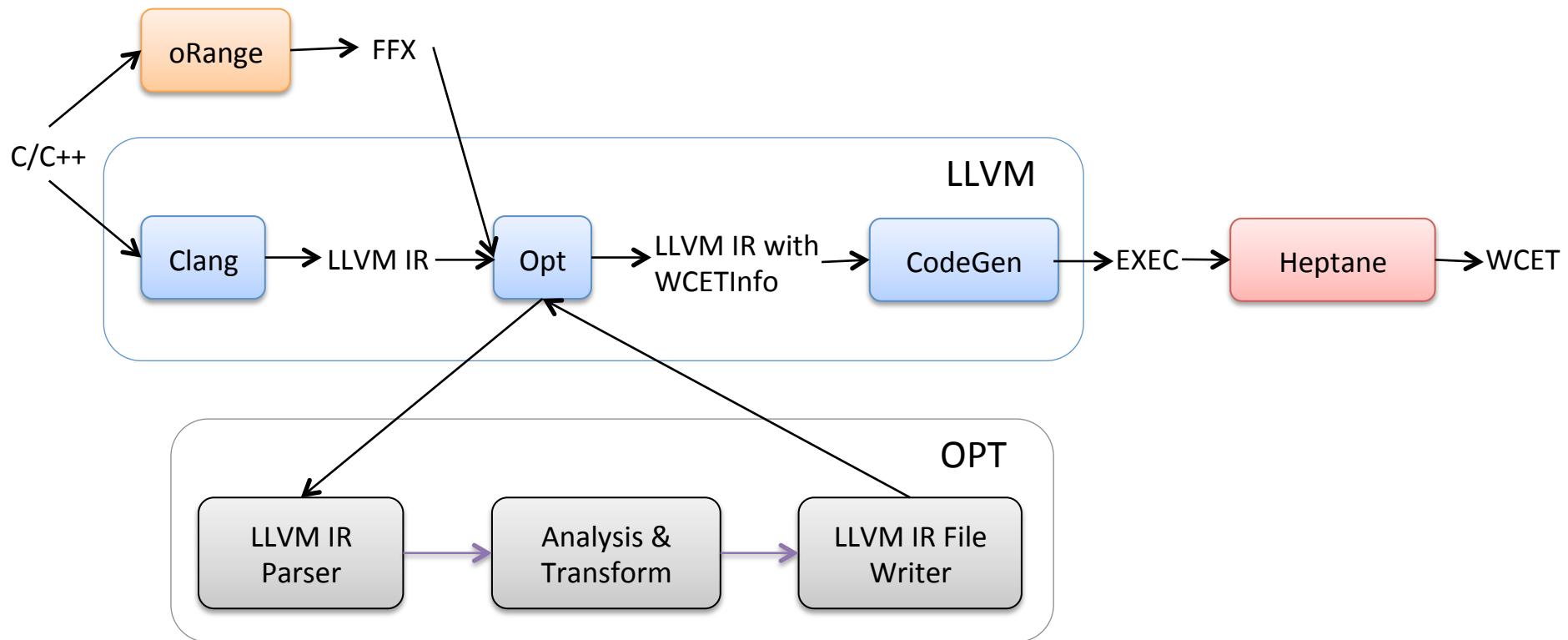


Loop Unrolling



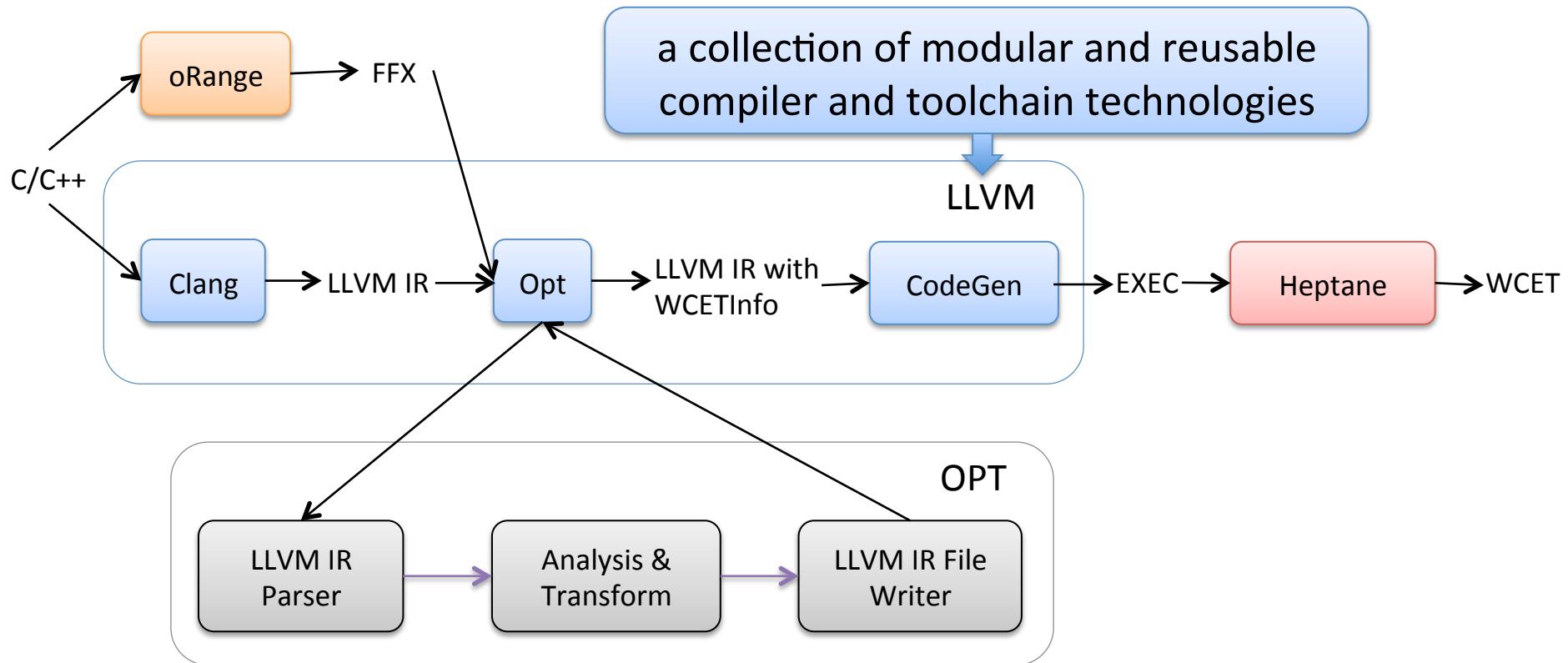
Implementation

- LLVM compiler infrastructure



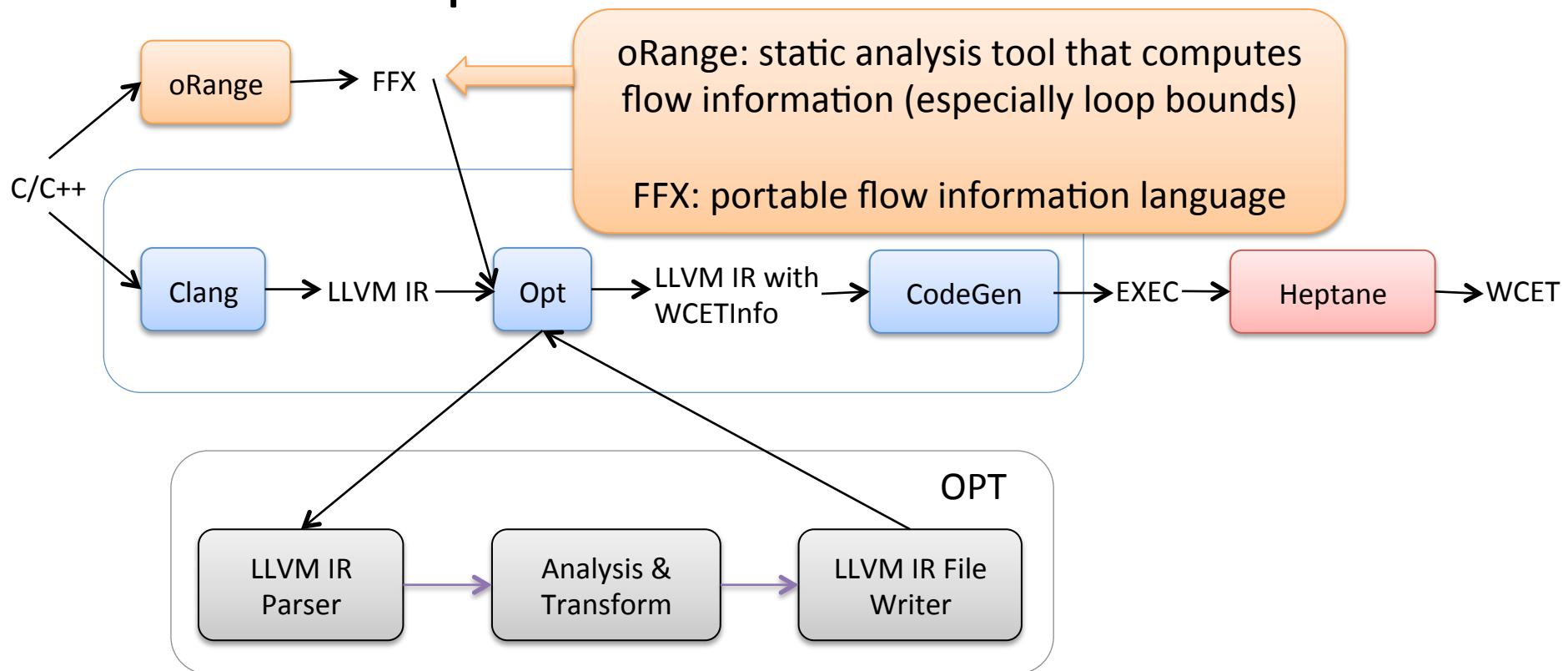
Implementation

- LLVM compiler infrastructure



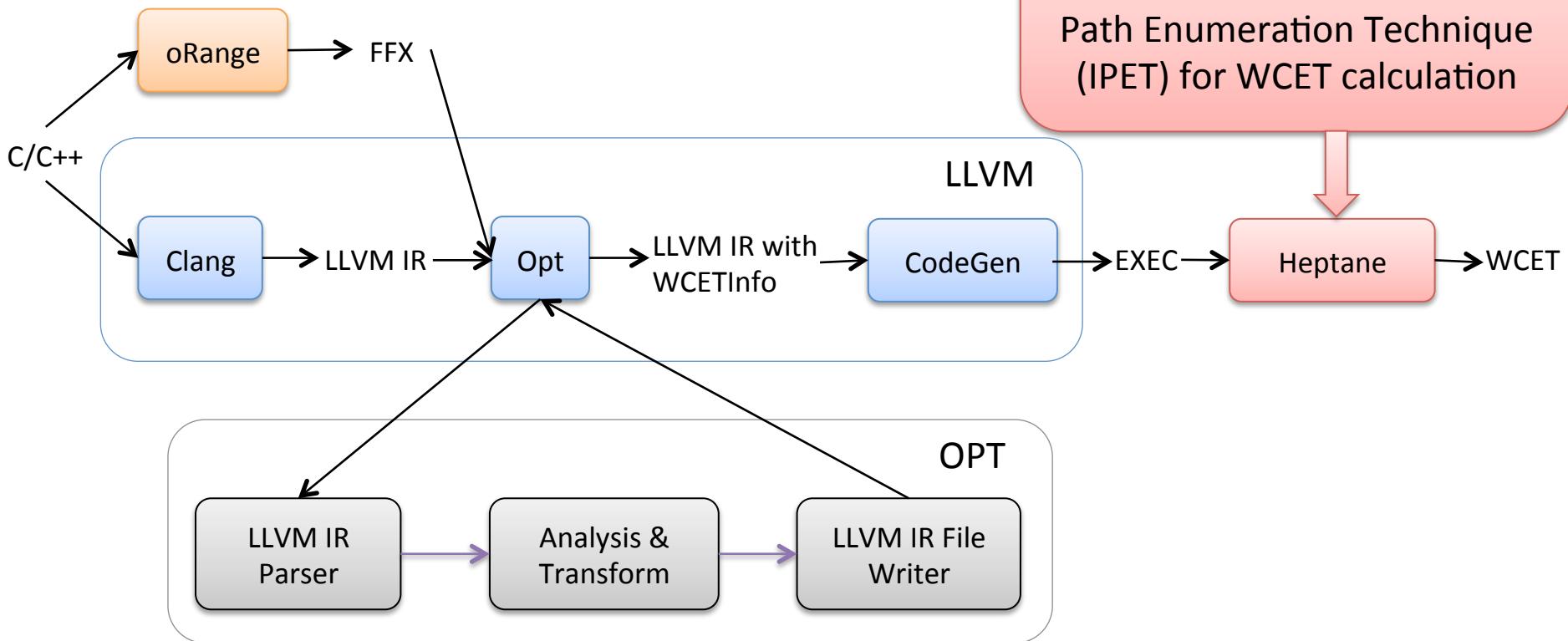
Implementation

- LLVM compiler infrastructure



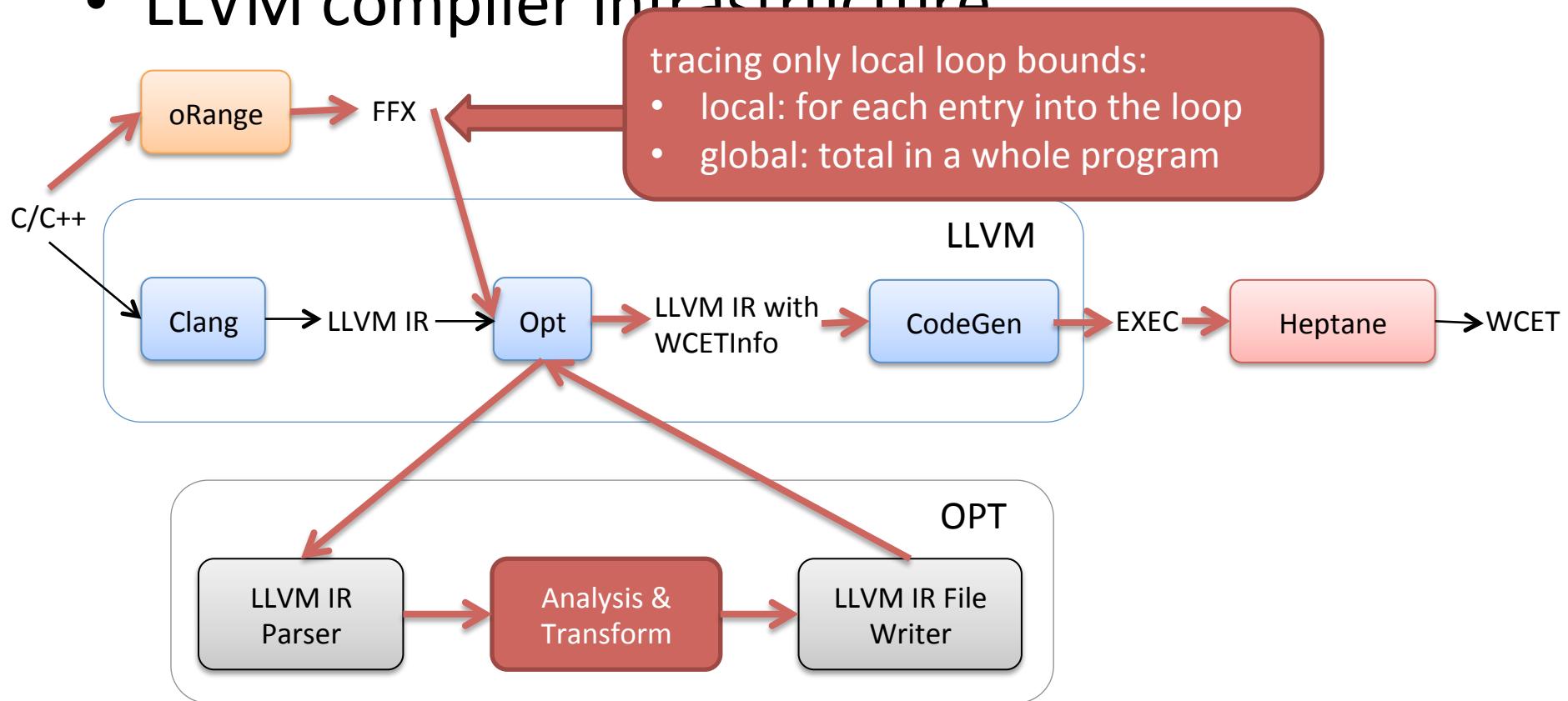
Implementation

- LLVM compiler infrastructure



Implementation

- LLVM compiler infrastructure



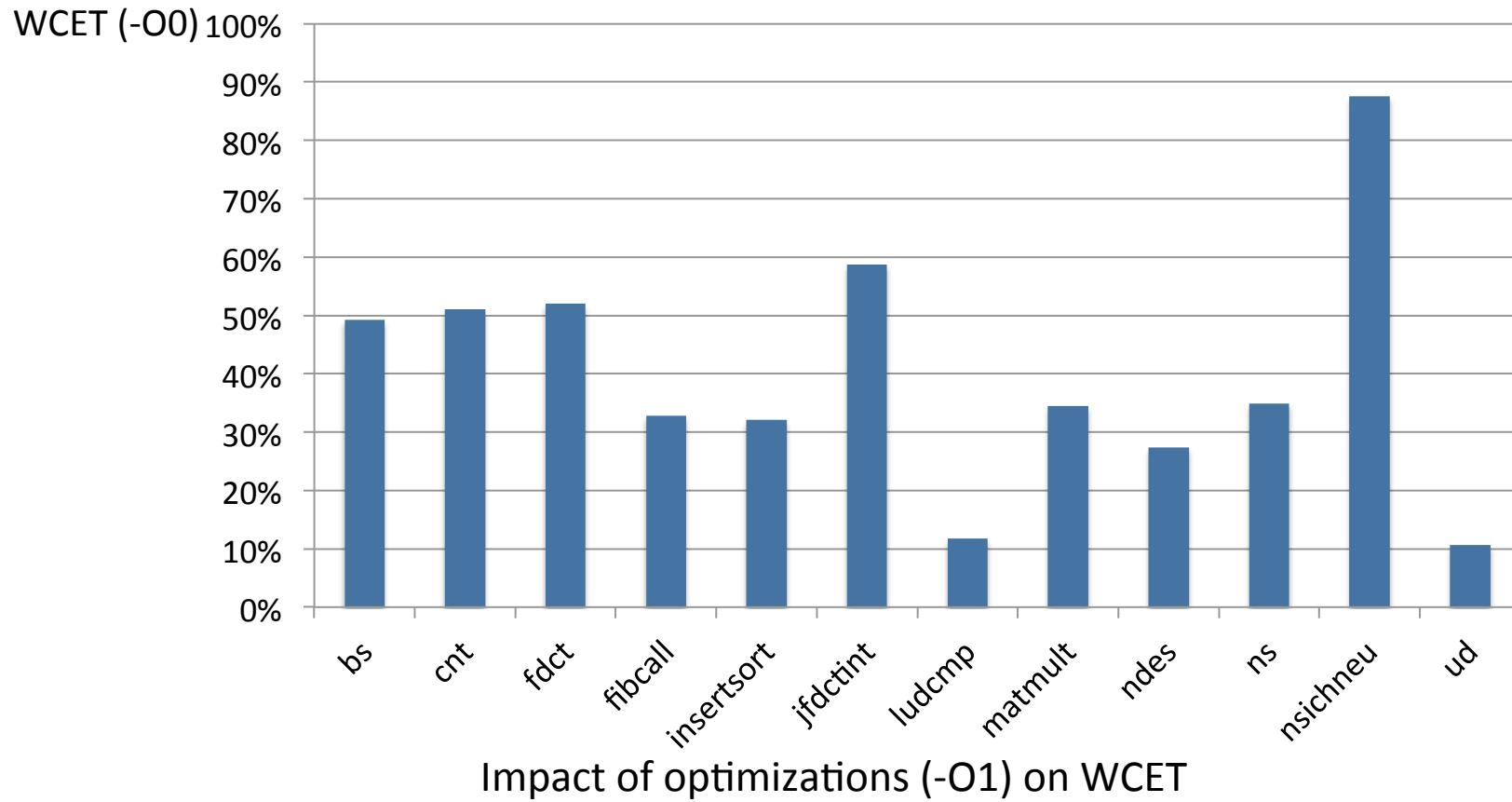
Experimental Environment

- Standardized set of WCET benchmarks from Mälardalen University
- ILP solver: CPLEX
- Hardware:
 - 32-bit MIPS processor
 - L1, L2 cache and Memory

Objective of Experiment

- Examine the impact of LLVM opt (-O1)
 - LLVM has most of its optimizations in O1
- Distinguish the individual impact among different optimizations

Experimental Results



- Y-axis: WCET (-O1), normalized with respect to the WCET (-O0)
- -O1 reduces estimated WCET: 60% in average

Impact of optimizations

- Individual impact of optimizations (1-off):
 - Loop Invariant Code Motion:
 - hoist a few instructions outside the loops
 - disabling it: WCET from 100% to 198% on **ud** (depth-3 loop nests)
- Combined impact of optimizations (2-off):
 - Sroa and Mem2reg: overlapping effects
 - Mem2reg: replace costly memory accesses by much faster register uses
 - Sroa: identify promotable elements of an aggregate alloca, and promote them to registers

Conclusion and Future Work

- Conclusion:
 - Transformation rules to transform flow information
 - Trace local loop bounds within compiler optimizations
 - Provide insight about the impact of optimizations on WCET
- Future work:
 - Extending traceability information beyond loop bound information
 - Contextual information
 - Global loop bounds (triangular loops)