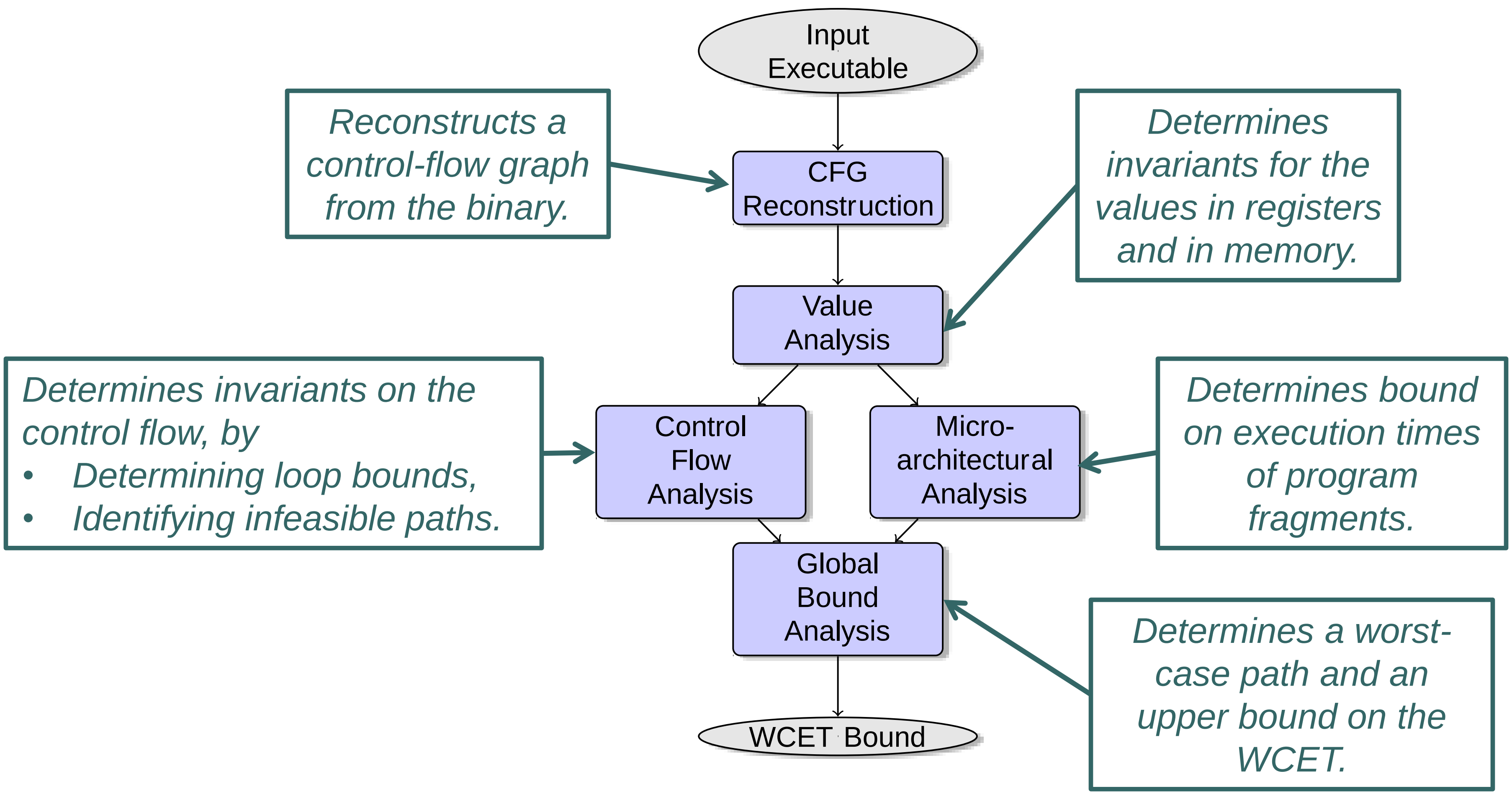
Verification of Real-Time Systems

Global Bound Analysis aka Path Analysis

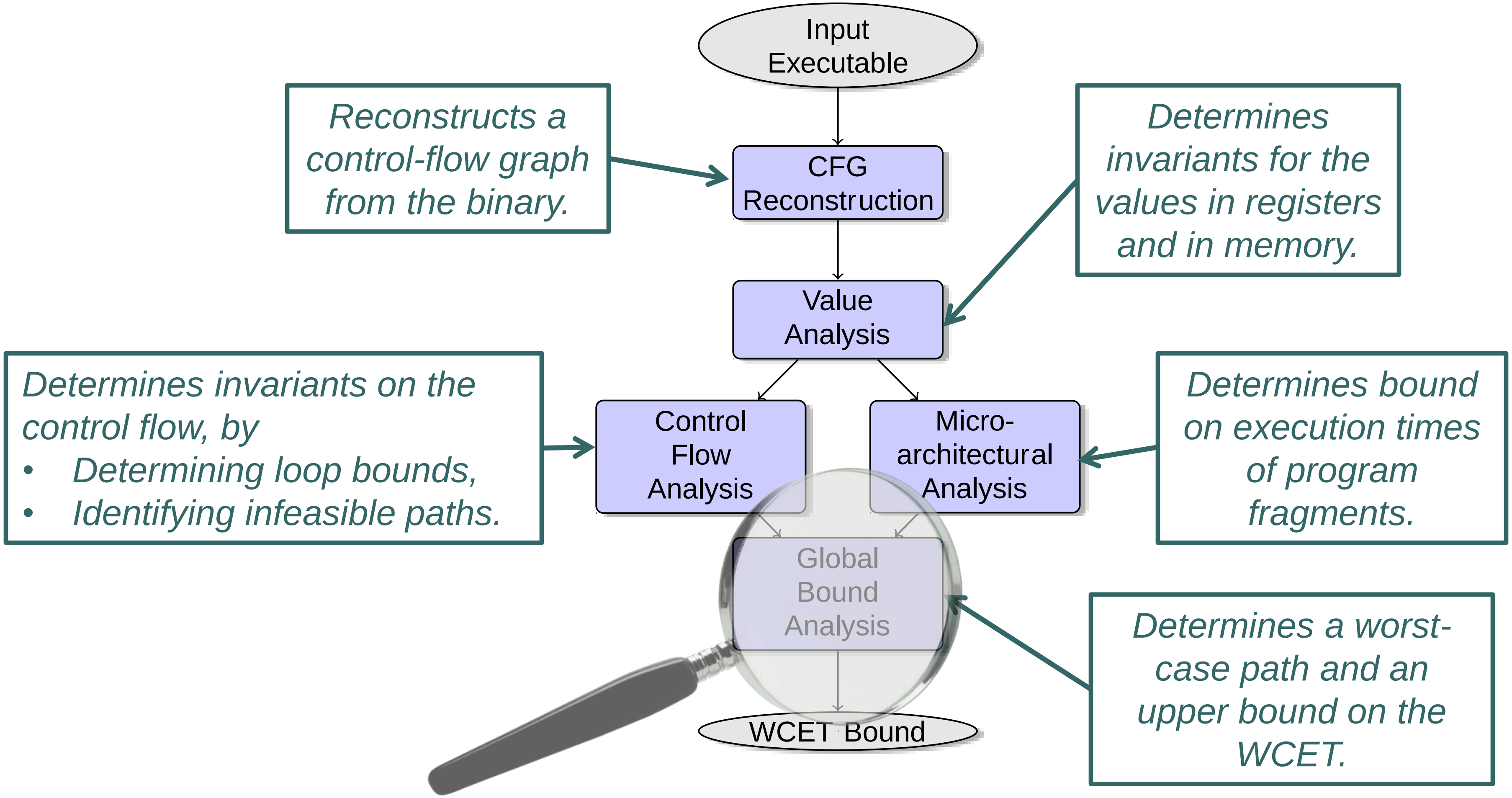
Jan Reineke

Advanced Lecture, Summer 2015

Structure of WCET Analyzers



Structure of WCET Analyzers



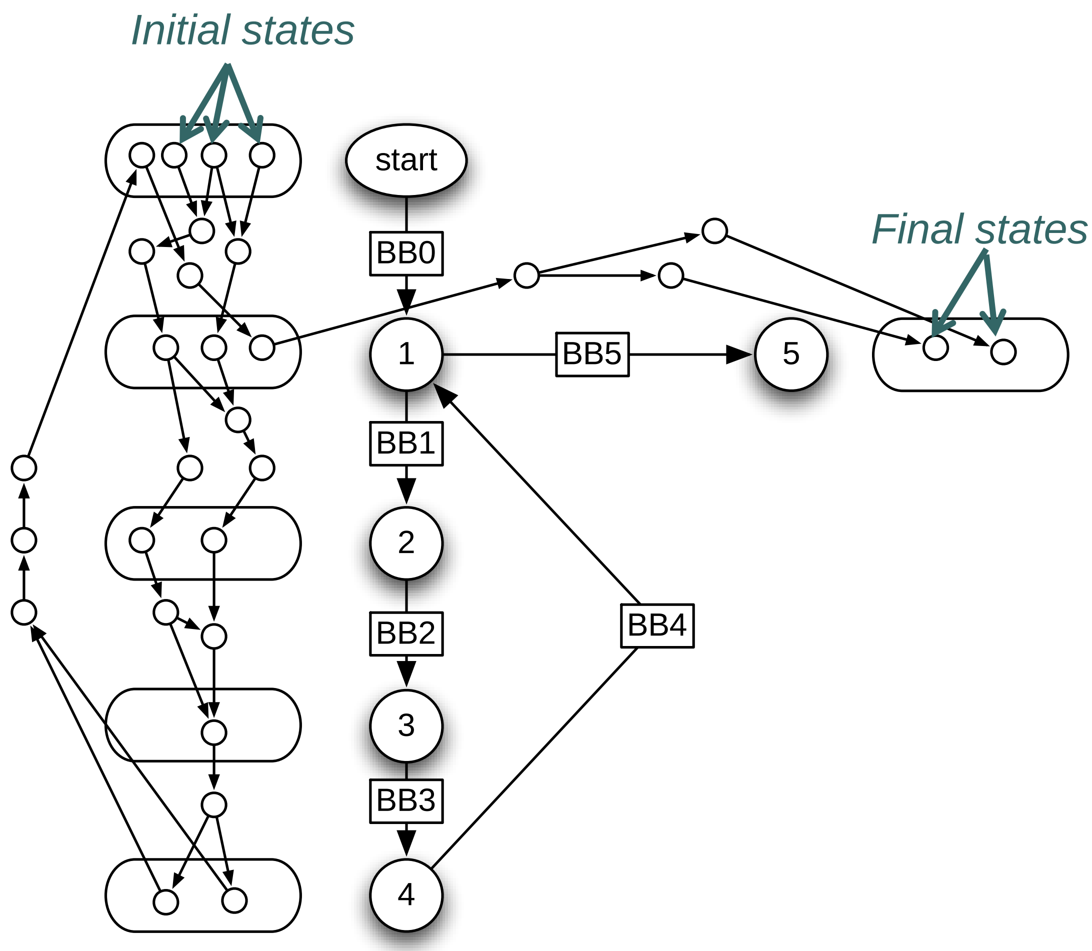
Global Bound Analysis aka Path Analysis

- Combines results of **control-flow analysis** and **microarchitectural analysis** to characterize **all** possible executions of a program on a given microarchitecture
- Searches for **longest execution** among those deemed possible

Result of Microarchitectural Analysis: Abstract Collecting Trace Semantics

Basic Block Execution Times (in cycles):

BB0: 2 or 3
BB1: 2 or 3
BB2: 2 or 3
BB3: 2
BB4: 4
BB5: 3



Result of Control-Flow Analysis

- Loop bounds: how often can the loop body be executed for each execution of the loop?
- Sometimes: infeasible paths, as e.g. in

```
if (a > 0) then
    fast();
else
    slow();//does not modify a
if (a > 1) then
    slow();
```

“Traditional” Path Analysis

- Encode problem as (Integer) Linear Program
  - Introduce one variable x_e for each edge e in the control-flow graph that captures the execution frequency of that edge
  - Structural constraints: “Kirchhoff’s law”:

    inflow = outflow at every program point
  - Loop bounds and knowledge about infeasible paths as additional constraints
  - Objective function:

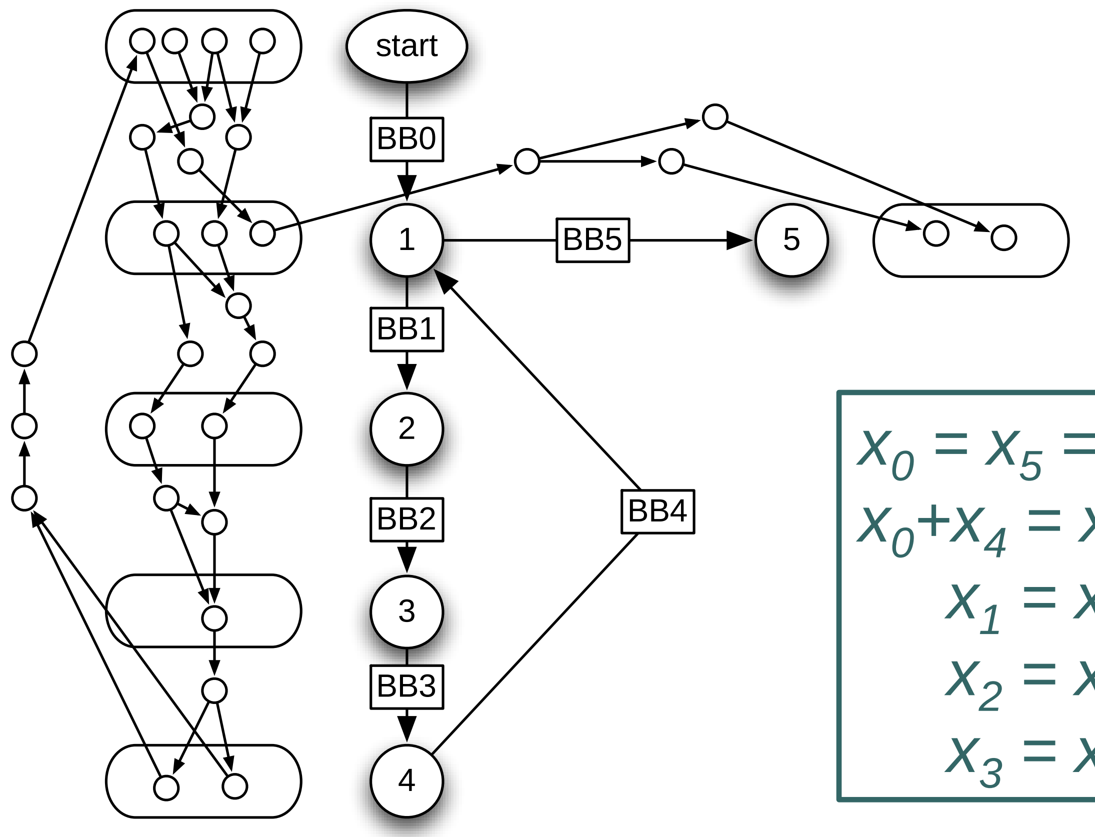
$$\max \sum_{e} c_e x_e$$

$$\text{s.t. } \textit{structural constraints + loop bounds, etc. hold}$$

Traditional Path Analysis: Example Structural Constraints

Basic Block Execution Times (in cycles):

- *BB0: 2 or 3*
- *BB1: 2 or 3*
- *BB2: 2 or 3*
- *BB3: 2*
- *BB4: 4*
- *BB5: 3*

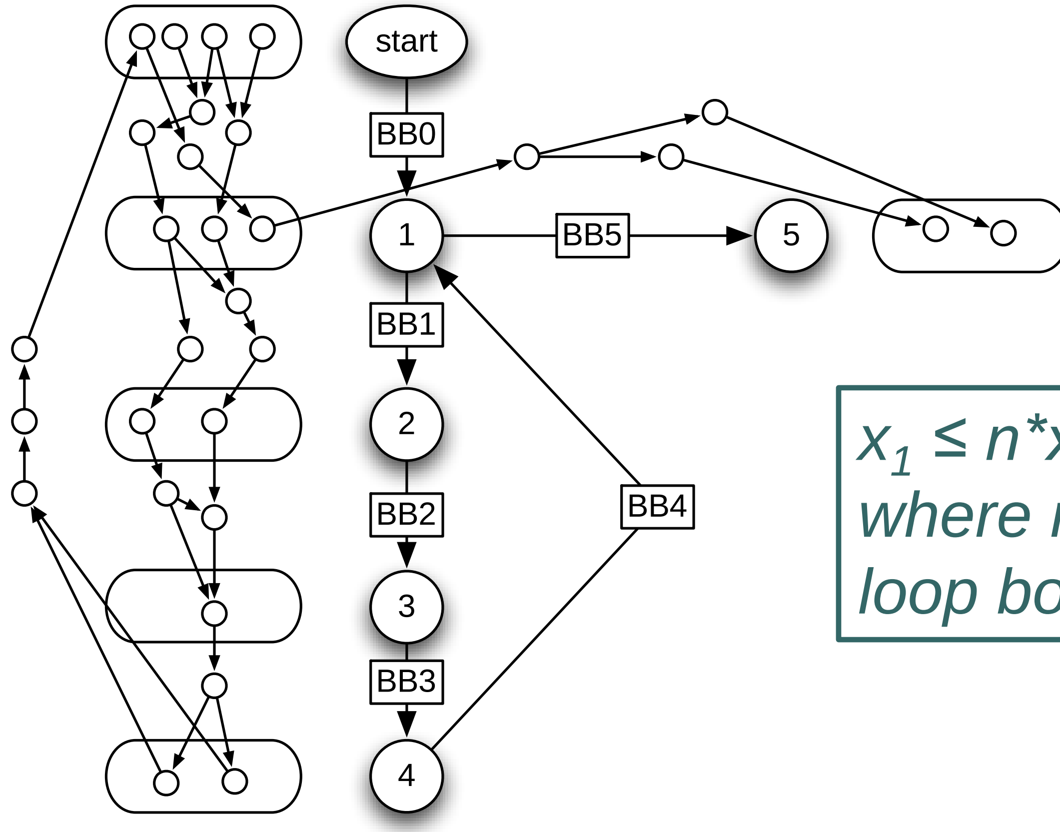


$$x_0 = x_5 = 1$$
$$x_0 + x_4 = x_1 + x_5$$
$$x_1 = x_2$$
$$x_2 = x_3$$
$$x_3 = x_4$$

Traditional Path Analysis: Example Loop Bounds

Basic Block Execution Times (in cycles):

BB0: 2 or 3
BB1: 2 or 3
BB2: 2 or 3
BB3: 2
BB4: 4
BB5: 3

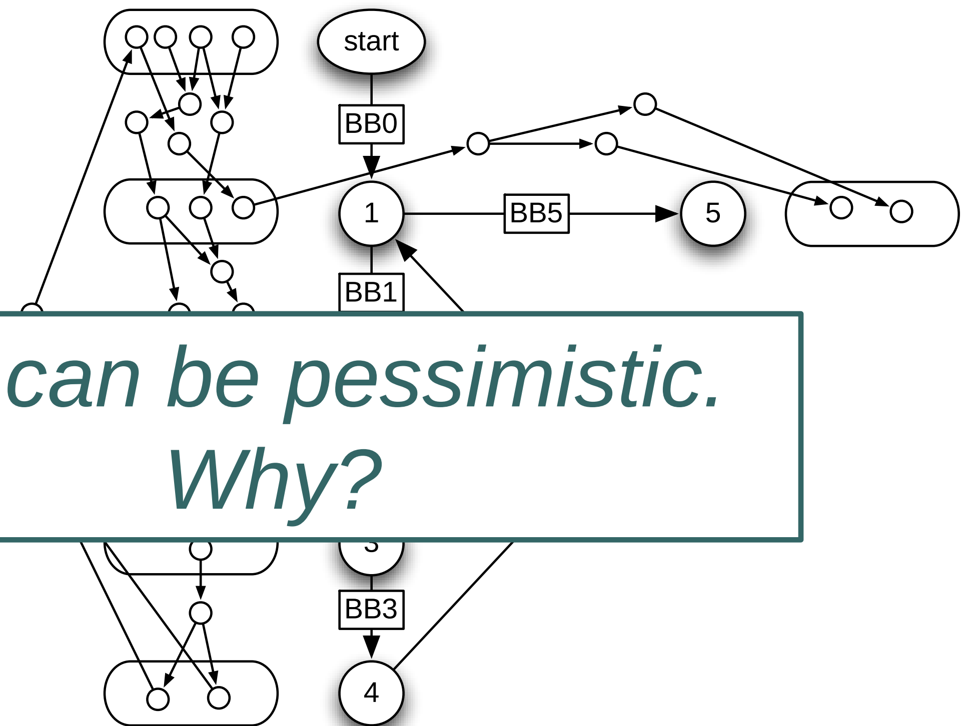


$x_1 \leq n*x_0$, where n is the loop bound

Traditional Path Analysis: Example Objective Function

Basic Block Execution Times (in cycles):

- *BB0: 2 or 3*
- *BB1: 2 or 3*
- *BB2: 2 or 3*
- *BB3: 2*
- *BB4: 4*
- *BB5: 3*



This can be pessimistic. Why?

$$\max \sum_{e} c_e x_e$$

$$= \max c_0 x_0 + c_1 x_1 + c_2 x_2 + c_3 x_3 + c_4 x_4 + c_5 x_5$$

$$= \max 3x_0 + 3x_1 + 3x_2 + 2x_3 + 4x_4 + 3x_5$$

State-Sensitive Path Analysis aka “Prediction-File” based Path Analysis

Idea: Distinguish different microarchitectural paths if they exhibit different timing

→ Excludes impossible combinations of worst-case timings of different basic blocks

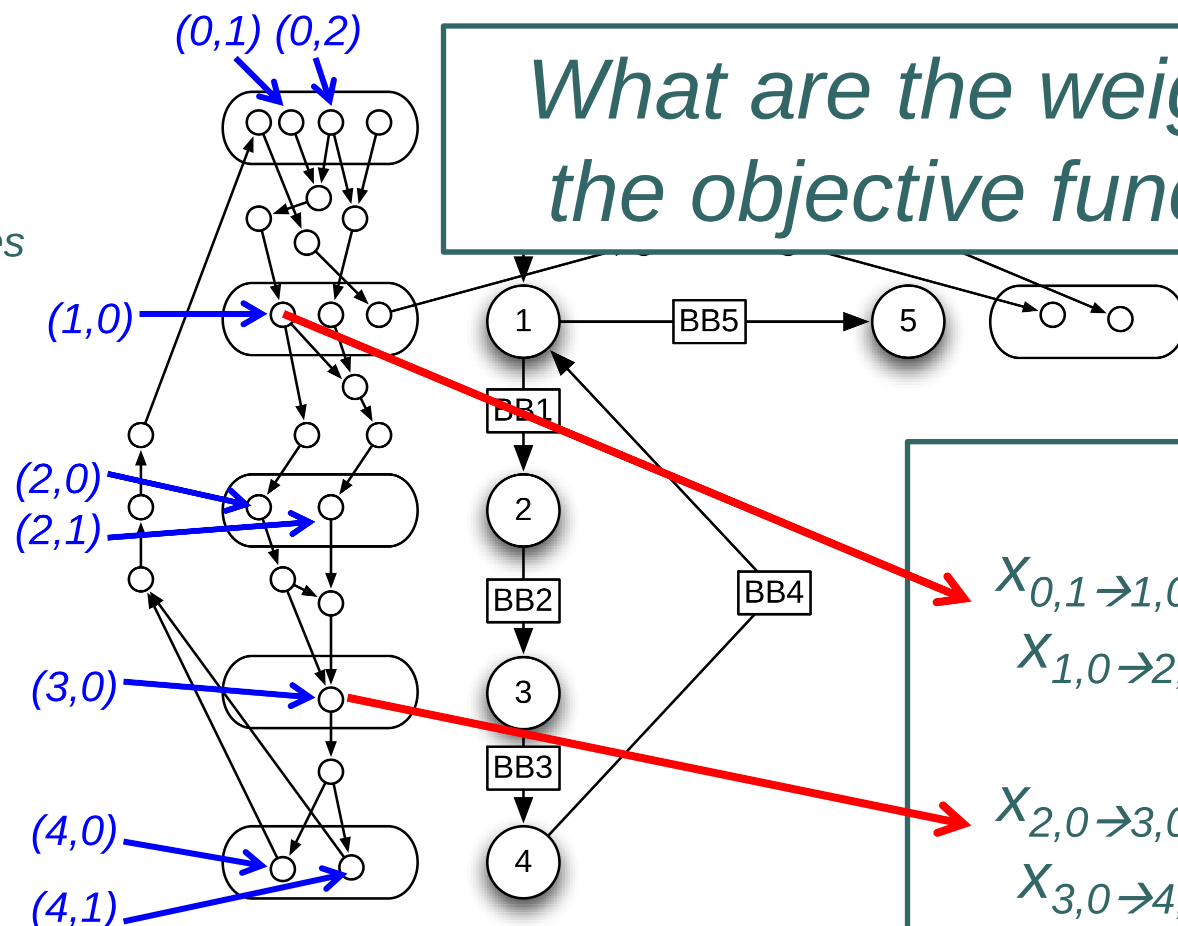
Approach:

- Microarchitectural states at the beginning of each basic block take the role of program points in the traditional analysis
- Introduce “frequency variable” for each *non-dominated* path from one such state to another.

State-Sensitive Path Analysis: Example Structural Constraints

Basic Block Execution Times (in cycles):

- *BB0: 2 or 3*
- *BB1: 2 or 3*
- *BB2: 2 or 3*
- *BB3: 2*
- *BB4: 4*
- *BB5: 3*



What are the weights in the objective function?

…

$$x_{0,1\to 1,0}+x_{0,2\to 1,0} = x_{1,0\to 2,0}+x_{1,0\to 2,1}$$

…

$$x_{2,0\to 3,0}+x_{2,1\to 3,0} = x_{3,0\to 4,0}+x_{3,0\to 4,1}$$

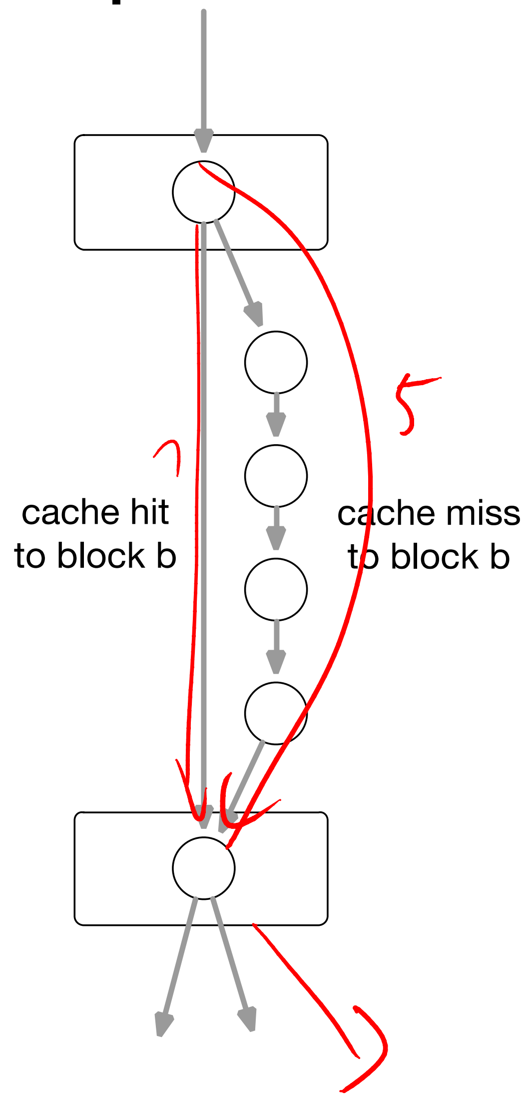
…

How to take into account **cumulative information** such as cache persistence?

Prohibits certain micro-architectural paths:

- → If block *b* is persistent, then at most one edge may be taken that corresponds to a miss to *b*.
- → Need to expose the information that an edge corresponds to a particular event, such as a cache miss to block *b*.

Taking into account cumulative information: Cache Persistence Example



Introduce a variable $x_{b,miss}$ that counts the number of misses to b.

Add persistence constraints for b:

$$x_{b,miss} \le 1 \text{ or } x_{b,miss} \le x_{scope}$$

where x_{scope} is the number of times the scope is entered in which *b* is persistent.

Frequency of edges e that correspond to misses to b should not exceed $x_{b,miss}$:

$$\sum_{e} x_e \le x_{b,miss}$$

Conclusions

High-level ideas of state-of-the-art path analysis:

- Encode all program paths implicitly by set of linear constraints.
- Objective function corresponds to cost of a particular path.

- Take into account microarchitectural states for higher precision → "State-sensitive path analysis"
- Expose events that can be bounded cumulatively, like cache misses.