TUHH
Hamburg University of Technology

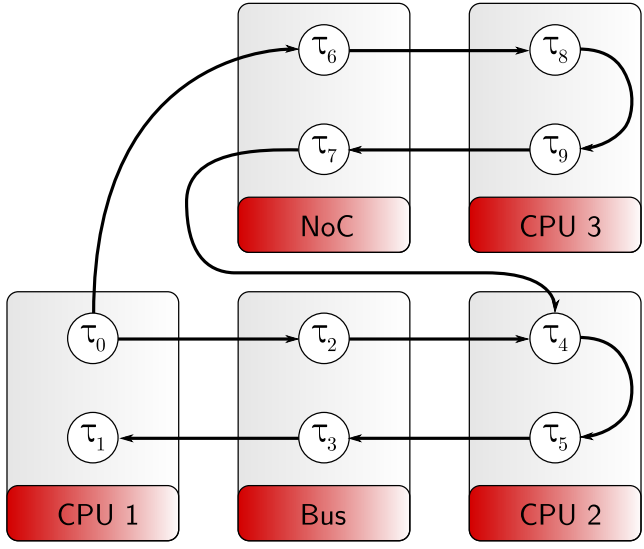
Compiler-based Extraction of Event Arrival Functions for Real-Time Systems Analysis

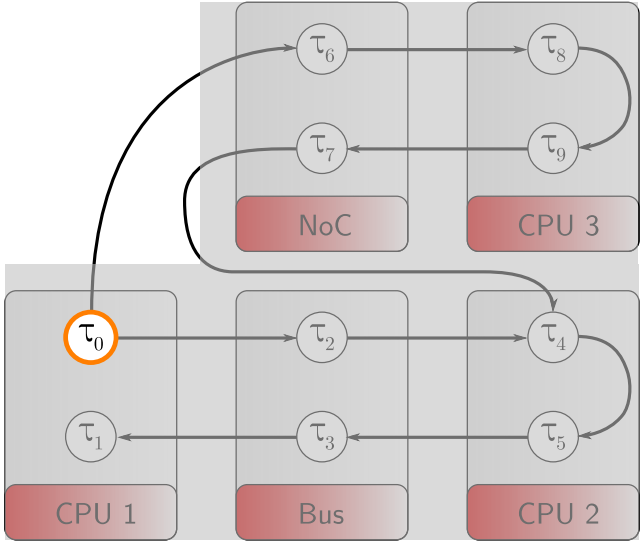
Dominic Oehlert Selma Saidi Heiko Falk

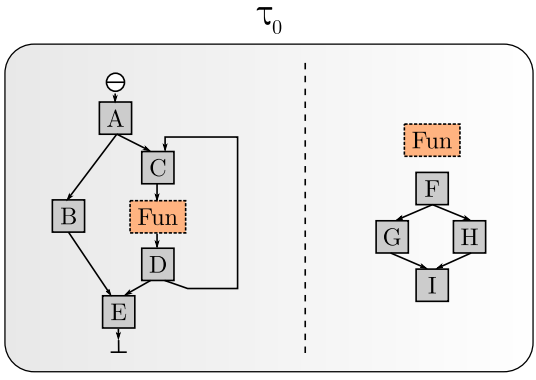
Institute of Embedded Systems
Hamburg University of Technology

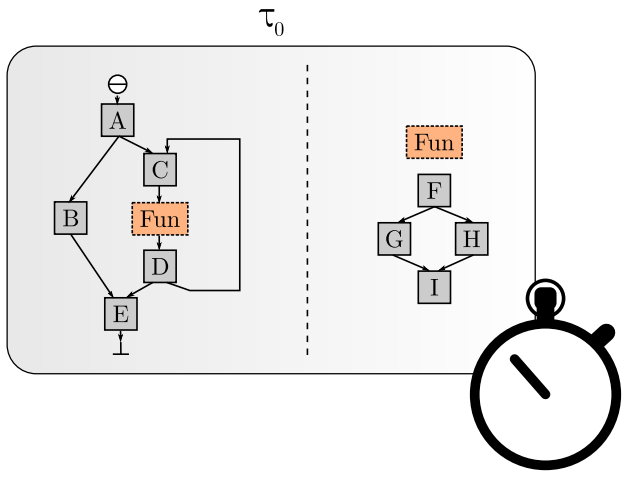
firstname.surname@tuhh.de

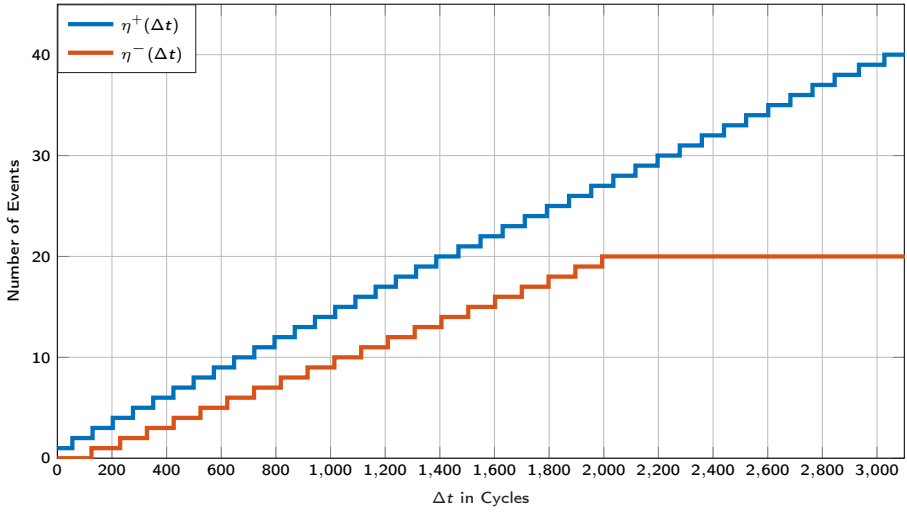
30th Euromicro Conference on Real-Time Systems (ECRTS), 2018

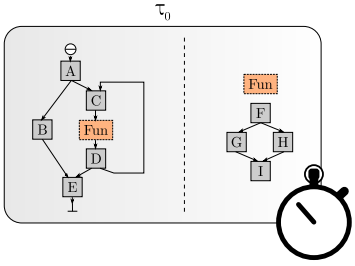

τ_6
τ_8
τ_7
τ_9
NoC
CPU 3
τ_0
τ_2
τ_4
τ_1
τ_3
τ_5
CPU 1
Bus
CPU 2

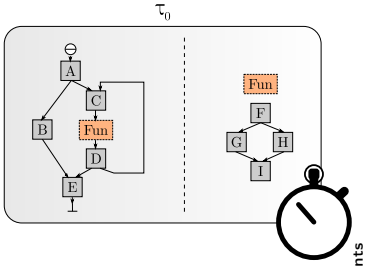

τ_6
τ_8
τ_7
τ_9
NoC
CPU 3
τ_0
τ_2
τ_4
τ_1
τ_3
τ_5
CPU 1
Bus
CPU 2

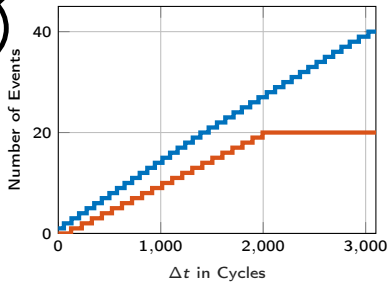

τ_0
A
C
B
Fun
D
E
Fun
F
G
H
I

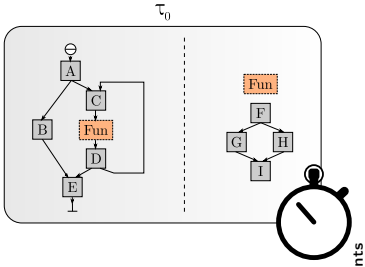

τ_0
A
C
B
Fun
D
E
Fun
F
G
H
I

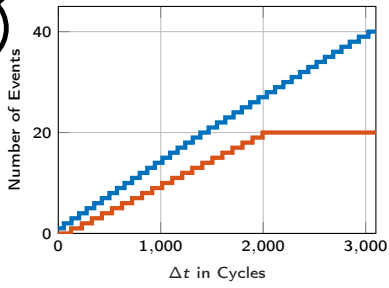

$\eta^+(\Delta t)$
$\eta^-(\Delta t)$
Number of Events
0
10
20
30
40
0
200
400
600
800
1,000
1,200
1,400
1,600
1,800
2,000
2,200
2,400
2,600
2,800
3,000
Δt in Cycles

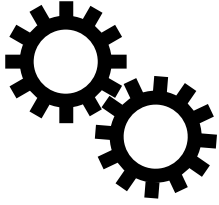

τ_0
A
C
B
Fun
D
E
Fun
F
G
H
I

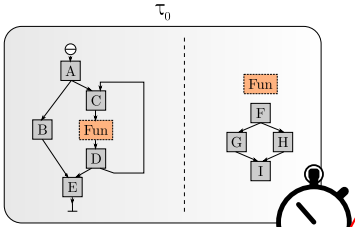

τ_0
A
C
B
Fun
D
E
Fun
F
G
H
I

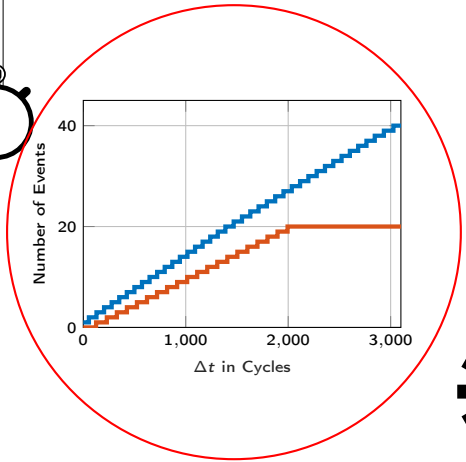

Number of Events
40
20
0
0
1,000
2,000
3,000
Δt in Cycles

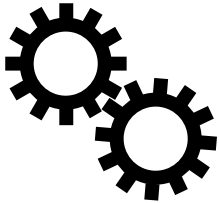

τ_0
A
B
C
Fun
D
E
Fun
F
G
H
I

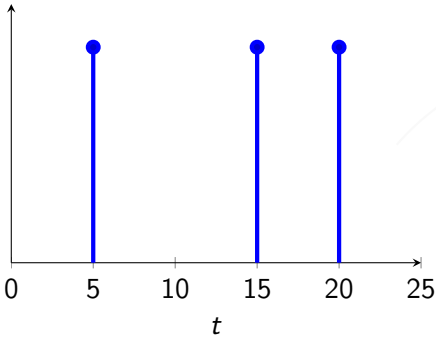

Number of Events
0
20
40
0
1,000
2,000
3,000
Δt in Cycles



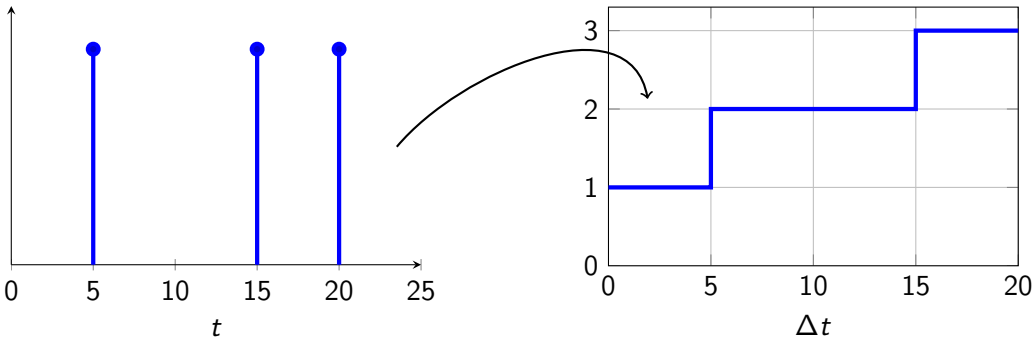

τ_0
A
C
B
Fun
D
E
Fun
F
G
H
I


Number of Events
40
20
0
0
1,000
2,000
3,000
Δt in Cycles



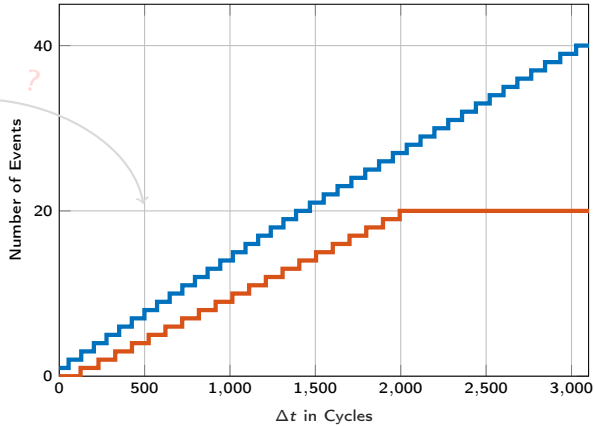

0
5
10
15
20
25
t

3
2
1
0
0
5
10
15
20
Δt

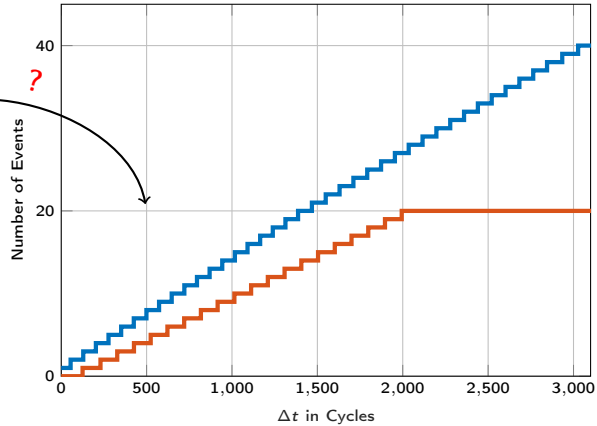

0
5
10
15
20
25
t
3
2
1
0
0
5
10
15
20
Δt

```
int globalData[ 2 ] = -1, 1;
volatile int comm;

int main() {

  for ( int i = 0; i < 20; ++i ) {

    if ( comm == 0 ) {
      globalData[ i % 2 ] = -1;
    }

  }

  return 0;
}
```

```
int globalData[ 2 ] = -1, 1;
volatile int comm;

int main() {

  for ( int i = 0; i < 20; ++i ) {

    if ( comm == 0 ) {
      globalData[ i % 2 ] = -1;
    }

  }

  return 0;
}
```



```
int globalData[ 2 ] = -1, 1;
volatile int comm;

int main() {

  for ( int i = 0; i < 20; ++i ) {

    if ( comm == 0 ) {
      globalData[ i % 2 ] = -1;
    }

  }

  return 0;
}
```



Extracting Curves?

- Capture traces
  - ⇒ Potentially unsafe
- Rely on specifications
  - ⇒ Potentially overly pessimistic
- Extraction based on the low-level representation

Extracting Curves?

- Capture traces
  - ⇒ Potentially unsafe
- Rely on specifications
  - ⇒ Potentially overly pessimistic
- Extraction based on the low-level representation

Extracting Curves?

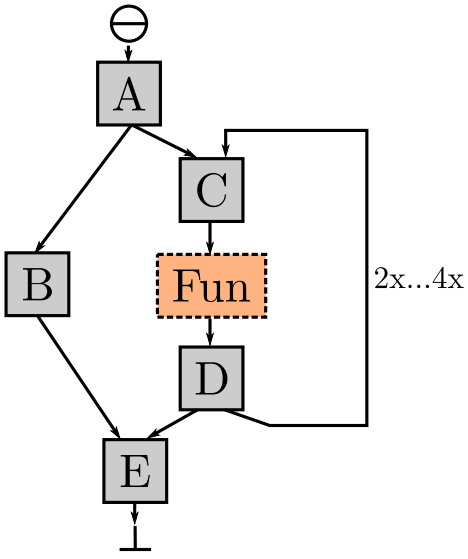
- Capture traces
  - ⇒ Potentially unsafe
- Rely on specifications
  - ⇒ Potentially overly pessimistic
- Extraction based on the low-level representation

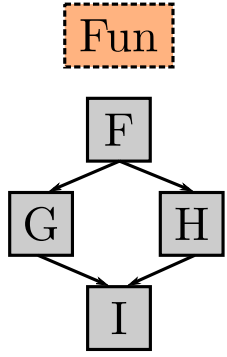
Extracting Curves?

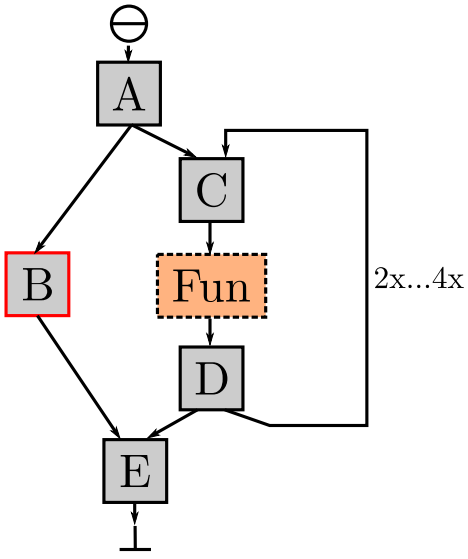
- Capture traces
  - ⇒ Potentially unsafe
- Rely on specifications
  - ⇒ Potentially overly pessimistic
- Extraction based on the low-level representation

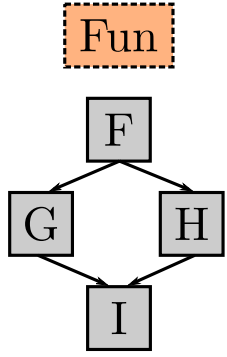
Extracting Curves?

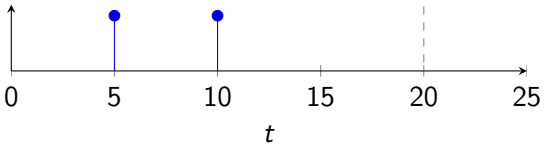
- Capture traces
  - $\Rightarrow$ Potentially unsafe
- Rely on specifications
  - $\Rightarrow$ Potentially overly pessimistic
- **Extraction based on the low-level representation**

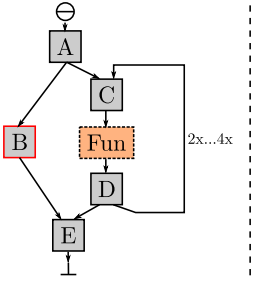

A
B
C
Fun
D
E
2x...4x


Fun
F
G
H
I

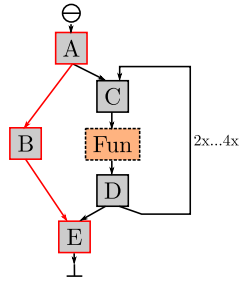

A
B
C
Fun
D
E
2x...4x

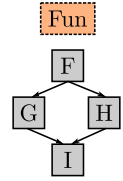

Fun
F
G
H
I

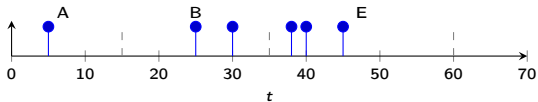

0
5
10
15
20
25
t


A
B
C
Fun
D
E
2x...4x
Fun
F
G
H
I

A
B
E
0
10
20
30
40
50
60
70
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t
. . .


A
C
B
Fun
D
E
2x...4x


Fun
F
G
H
I

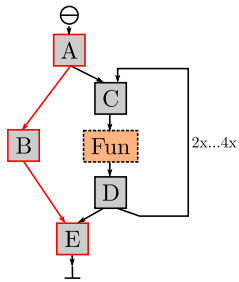

A
B
E
0
10
20
30
40
50
60
70
t

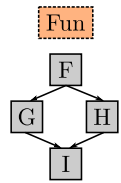
A
C
F
G
...
0
20
40
60
80
t

A
C
F
H
...
0
20
40
60
80
t

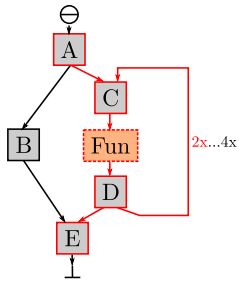
A
C
F
G
...
0
20
40
60
80
t

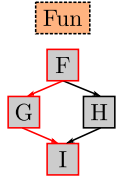
. . .

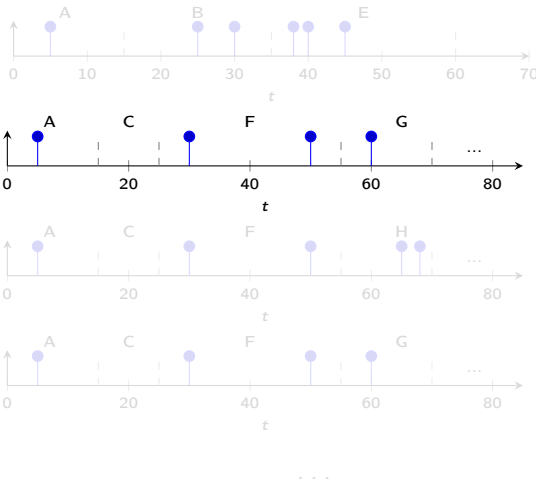

A
B
C
Fun
D
E
2x...4x

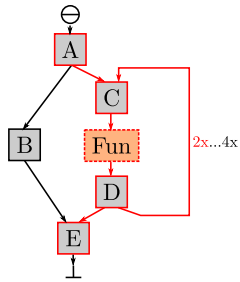

Fun
F
G
H
I

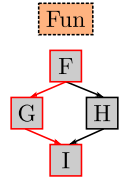
A
B
E
0
10
20
30
40
50
60
70
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t
. . .


A
C
Fun
D
B
E
2x...4x

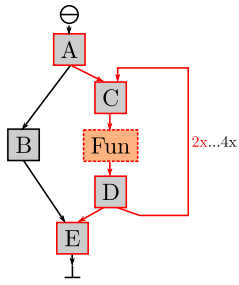

Fun
F
G
H
I

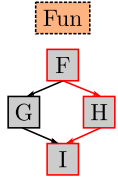

A
B
E
0
10
20
30
40
50
60
70
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t
. . .

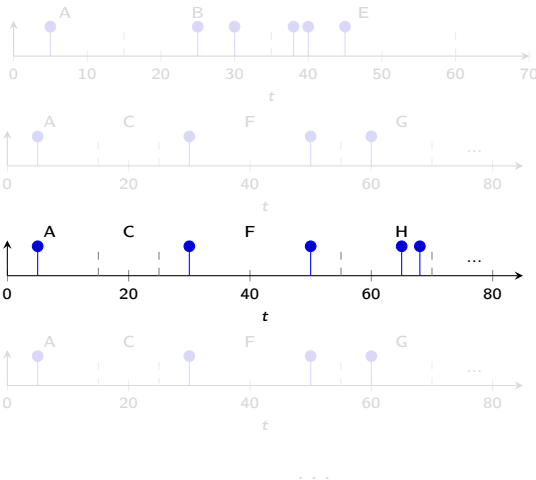

A
C
B
Fun
D
E
2x...4x

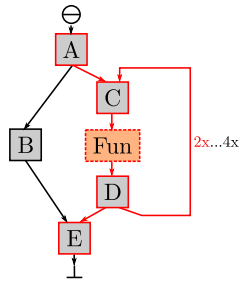

Fun
F
G
H
I

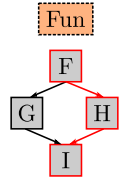
A
B
E
0
10
20
30
40
50
60
70
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t
. . .


A
C
Fun
D
B
E
2x...4x

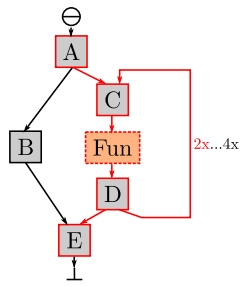

Fun
F
G
H
I

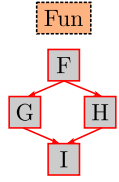

A
B
E
0
10
20
30
40
50
60
70
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t
. . .

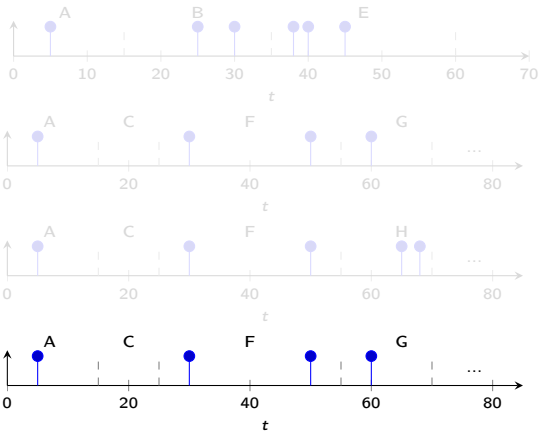

A
B
C
Fun
D
E
2x...4x

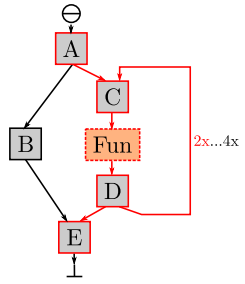

Fun
F
G
H
I

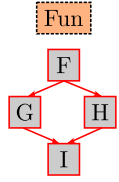
A
B
E
0
10
20
30
40
50
60
70
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t
. . .


A
C
Fun
D
B
E
2x...4x


Fun
F
G
H
I

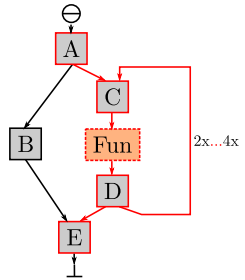

A
B
E
0
10
20
30
40
50
60
70
t
A
C
F
G
...
0
20
40
60
80
t
A
C
F
H
...
0
20
40
60
80
t
A
C
F
G
...
0
20
40
60
80
t

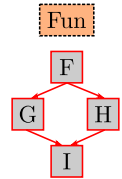

A
C
Fun
D
B
E
2x...4x


Fun
F
G
H
I

A
B
E
0
10
20
30
40
50
60
70
t
A
C
F
G
0
20
40
60
80
t
A
C
F
H
0
20
40
60
80
t
A
C
F
G
0
20
40
60
80
t

...


A
C
Fun
D
B
E
2x...4x


Fun
F
G
H
I

Path Analysis for Arrival Functions

- Explicit path analysis quickly becomes practically infeasible
  - ⇒ Sliding window for all traces

⇒ Adapt path analysis techniques

⇒ Introduce granularity
  - Trade-off between precision and runtime

Path Analysis for Arrival Functions

- Explicit path analysis quickly becomes practically infeasible
  - $\Rightarrow$ Sliding window for all traces

$\Rightarrow$ Adapt path analysis techniques

$\Rightarrow$ Introduce granularity
  - Trade-off between precision and runtime

Path Analysis for Arrival Functions

- Explicit path analysis quickly becomes practically infeasible
  - $\Rightarrow$ Sliding window for all traces

$\Rightarrow$ Adapt path analysis techniques

$\Rightarrow$ Introduce granularity
  - Trade-off between precision and runtime

Path Analysis for Arrival Functions

- Explicit path analysis quickly becomes practically infeasible
  - $\Rightarrow$ Sliding window for all traces

$\Rightarrow$ Adapt path analysis techniques

$\Rightarrow$ Introduce granularity
  - Trade-off between precision and runtime

Path Analysis for Arrival Functions

- Explicit path analysis quickly becomes practically infeasible
  - $\Rightarrow$ Sliding window for all traces

$\Rightarrow$ Adapt path analysis techniques

$\Rightarrow$ Introduce granularity
  - Trade-off between precision and runtime

Overview

Overview

Event Arrival Functions

- Abstract notion of interfering events (e.g., task activations, shared memory accesses, ...)

Definition

Let $\eta_i^+(\Delta t)$ and $\eta_i^-(\Delta t)$ denote for each task i the maximum and minimum number of events issued within a time window of size Δt.
Their pseudo-inverse counterparts $\delta^+(n)$ and $\delta^-(n)$, return the maximum/minimum time interval between the first and the last event in any sequence of n event arrivals.

Event Arrival Functions

- Abstract notion of interfering events (e.g., task activations, shared memory accesses, ...)

Definition

Let $\eta_i^+(\Delta t)$ and $\eta_i^-(\Delta t)$ denote for each task i the maximum and minimum number of events issued within a time window of size Δt.
Their pseudo-inverse counterparts $\delta^+(n)$ and $\delta^-(n)$, return the maximum/minimum time interval between the first and the last event in any sequence of n event arrivals.

Event Arrival Functions

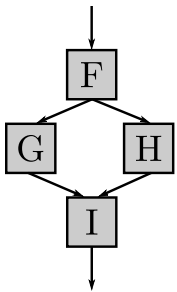
- Abstract notion of interfering events (e.g., task activations, shared memory accesses, ...)

Definition

Let $\eta_i^+(\Delta t)$ and $\eta_i^-(\Delta t)$ denote for each task i the maximum and minimum number of events issued within a time window of size Δt.
Their pseudo-inverse counterparts $\delta^+(n)$ and $\delta^-(n)$, return the maximum/minimum time interval between the first and the last event in any sequence of n event arrivals.

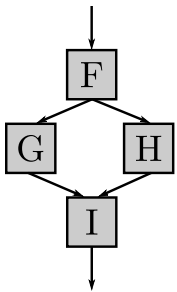
Implicit Path Enumeration Technique

- Describe all feasible paths by formulating an Integer Linear Program (ILP) [DAC95]
- Enforcing complete path through a program using junction rules



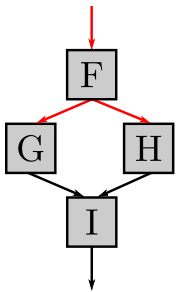
Implicit Path Enumeration Technique

- Describe all feasible paths by formulating an Integer Linear Program (ILP) [DAC95]
- Enforcing complete path through a program using junction rules



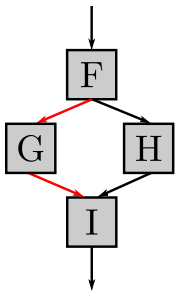
Implicit Path Enumeration Technique

- Describe all feasible paths by formulating an Integer Linear Program (ILP) [DAC95]
- Enforcing complete path through a program using junction rules



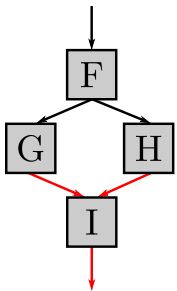
Implicit Path Enumeration Technique

- Describe all feasible paths by formulating an Integer Linear Program (ILP) [DAC95]
- Enforcing complete path through a program using junction rules



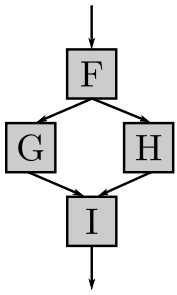
Implicit Path Enumeration Technique

- Describe all feasible paths by formulating an Integer Linear Program (ILP) [DAC95]
- Enforcing complete path through a program using junction rules



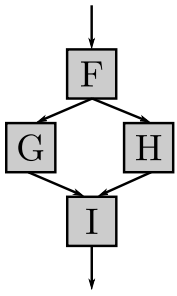
Adapted IPET

- First introduced by Jacobs et al. [RTNS15]
- Introduce "movable" sources and sinks
- Determine max. (resp. min.) number of events per basic block
- Maximize (resp. minimize) accumulated number of events over sub-path
  - $\Rightarrow$ While accumulated time $w \leq \Delta t$ (resp. $w \geq \Delta t$)



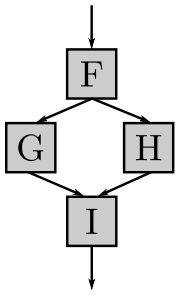
Adapted IPET

- First introduced by Jacobs et al. [RTNS15]
- Introduce “movable” sources and sinks
- Determine max. (resp. min.) number of events per basic block
- Maximize (resp. minimize) accumulated number of events over sub-path
  - $\Rightarrow$ While accumulated time $w \leq \Delta t$ (resp. $w \geq \Delta t$)



Adapted IPET

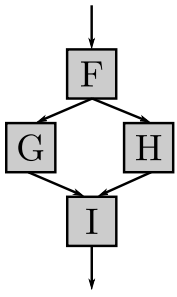
- First introduced by Jacobs et al. [RTNS15]
- Introduce "movable" sources and sinks
- Determine max. (resp. min.) number of events per basic block
- Maximize (resp. minimize) accumulated number of events over sub-path
  $\Rightarrow$ While accumulated time $w \leq \Delta t$ (resp. $w \geq \Delta t$)



Adapted IPET

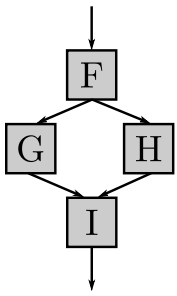
- First introduced by Jacobs et al. [RTNS15]
- Introduce "movable" sources and sinks
- Determine max. (resp. min.) number of events per basic block
- Maximize (resp. minimize) accumulated number of events over sub-path

  $\Rightarrow$ While accumulated time $w \leq \Delta t$ (resp. $w \geq \Delta t$)



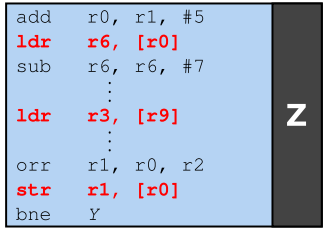
Adapted IPET

- First introduced by Jacobs et al. [RTNS15]
- Introduce "movable" sources and sinks
- Determine max. (resp. min.) number of events per basic block
- Maximize (resp. minimize) accumulated number of events over sub-path
  $\Rightarrow$ While accumulated time $w \leq \Delta t$ (resp. $w \geq \Delta t$)



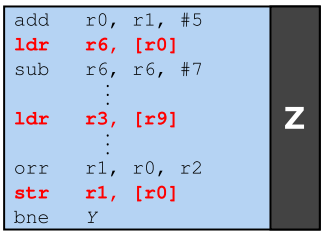
Automated Extraction

- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe



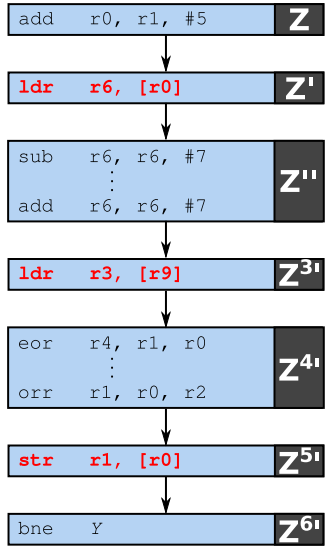
Automated Extraction

- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe



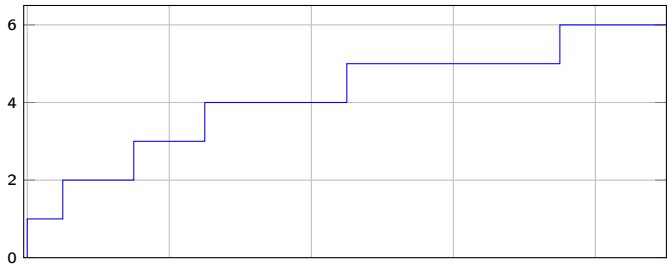
Automated Extraction

- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe



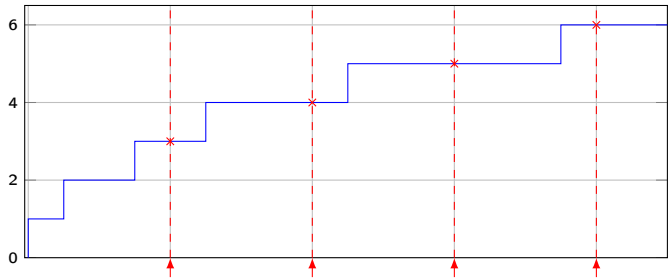
Automated Extraction

- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe



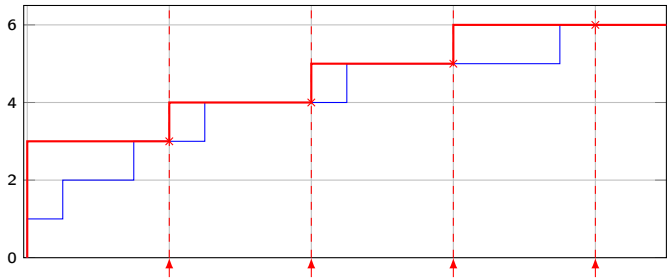
Automated Extraction

- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe



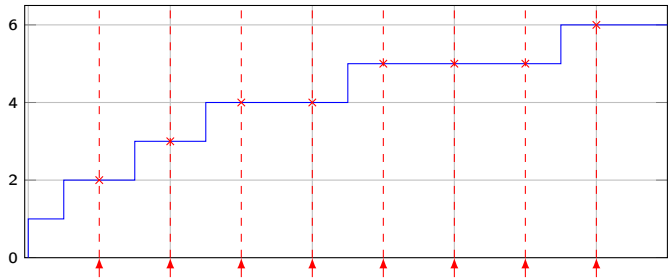
Automated Extraction

- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe



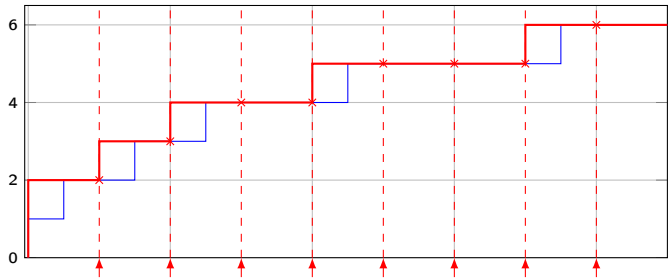
Automated Extraction

- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe



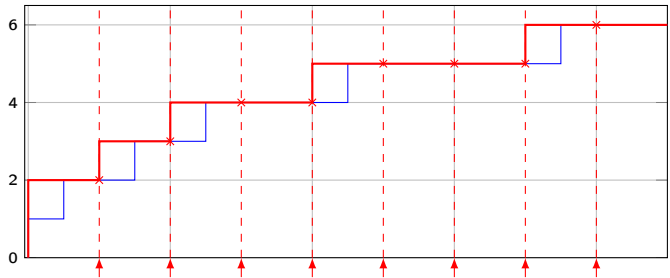
Automated Extraction

- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe



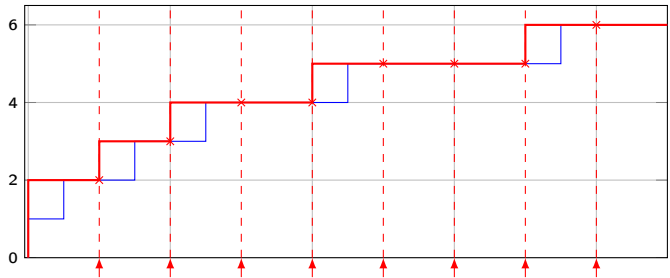
Automated Extraction

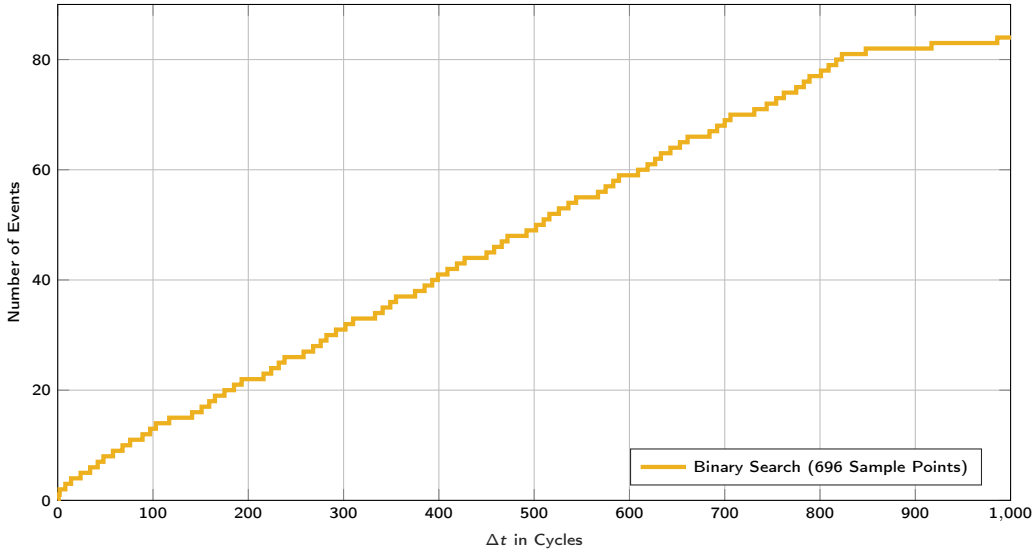
- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe

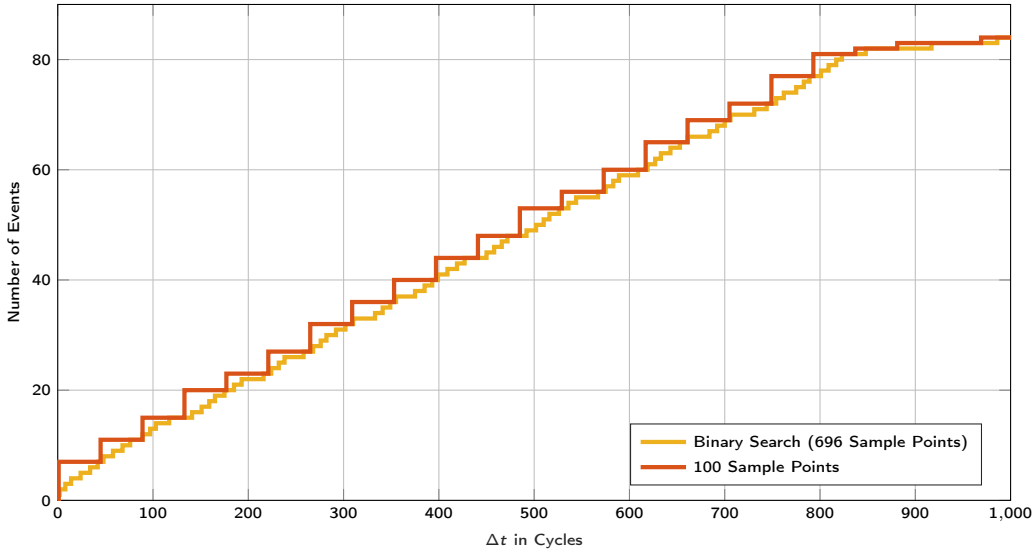


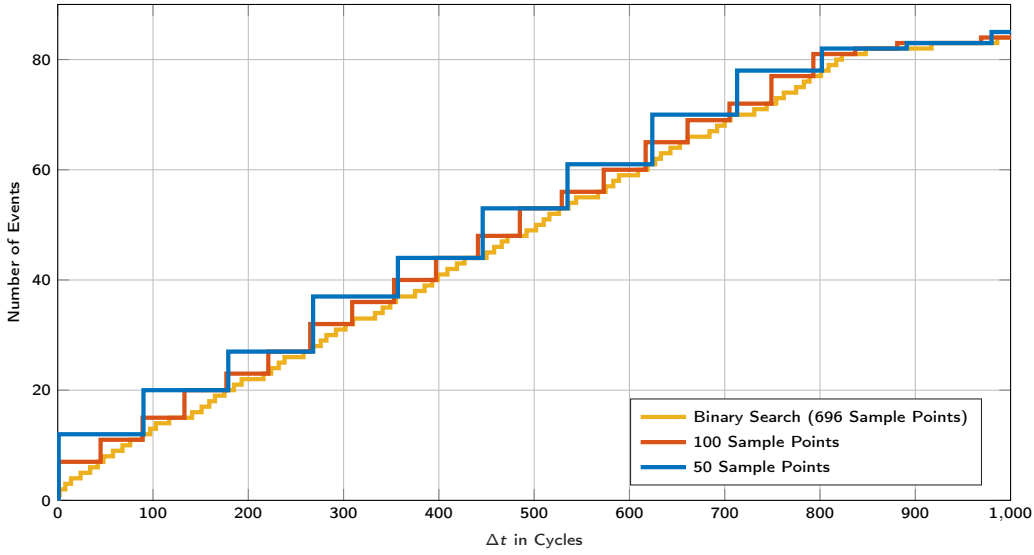
Automated Extraction

- Two dimensions of granularity:
  - Events per basic block
  - Sample rate
- Adjustable trade-off between runtime and tightness
  - Albeit, arrival functions will be safe



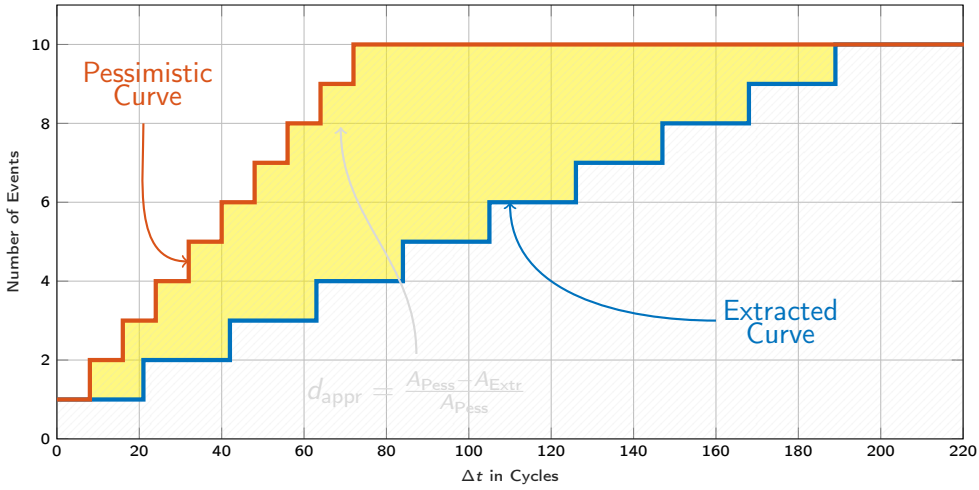

Number of Events
0
20
40
60
80
0
100
200
300
400
500
600
700
800
900
1,000
Δt in Cycles
Binary Search (696 Sample Points)


Number of Events
0
20
40
60
80
0
100
200
300
400
500
600
700
800
900
1,000
Δt in Cycles
Binary Search (696 Sample Points)
100 Sample Points

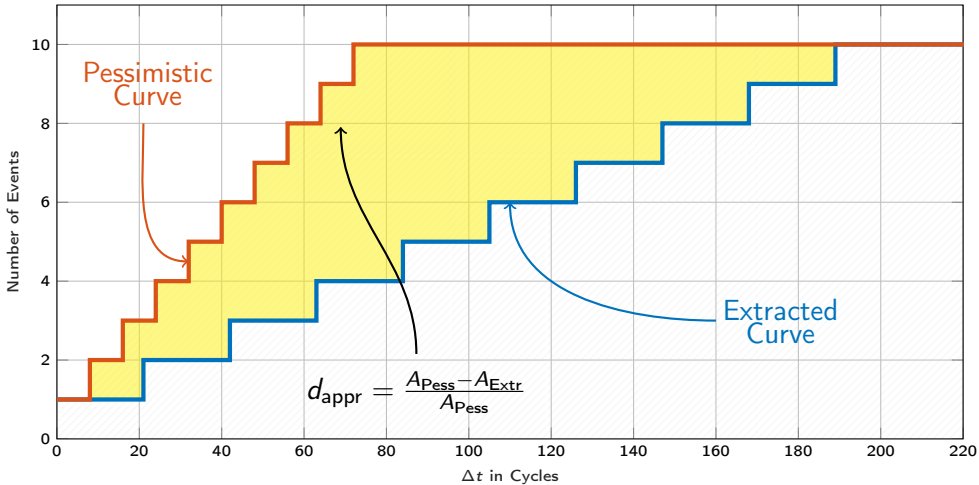

Number of Events
0
20
40
60
80
0
100
200
300
400
500
600
700
800
900
1,000
Δt in Cycles
Binary Search (696 Sample Points)
100 Sample Points
50 Sample Points

Overview

Overapproximation Metric

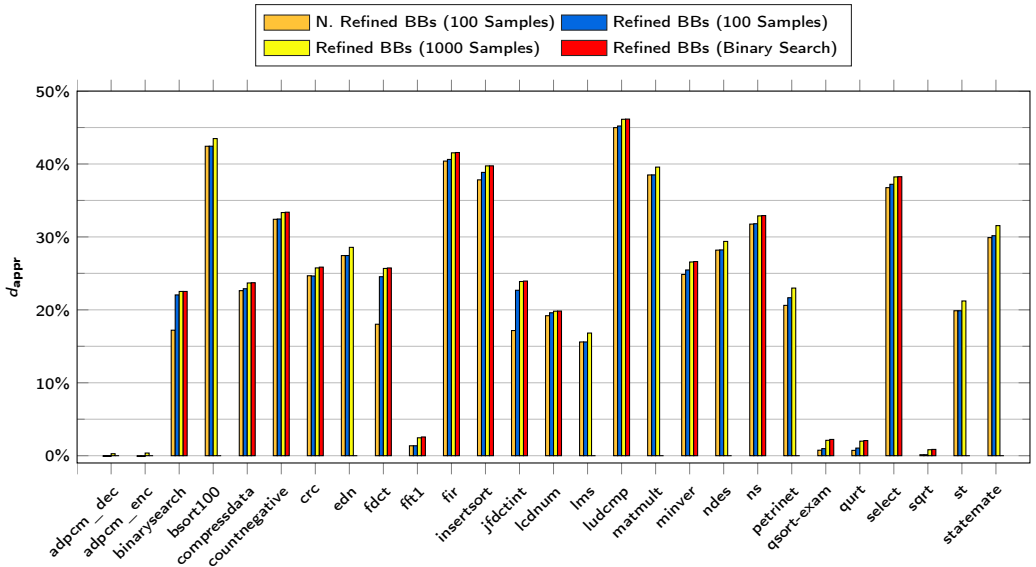


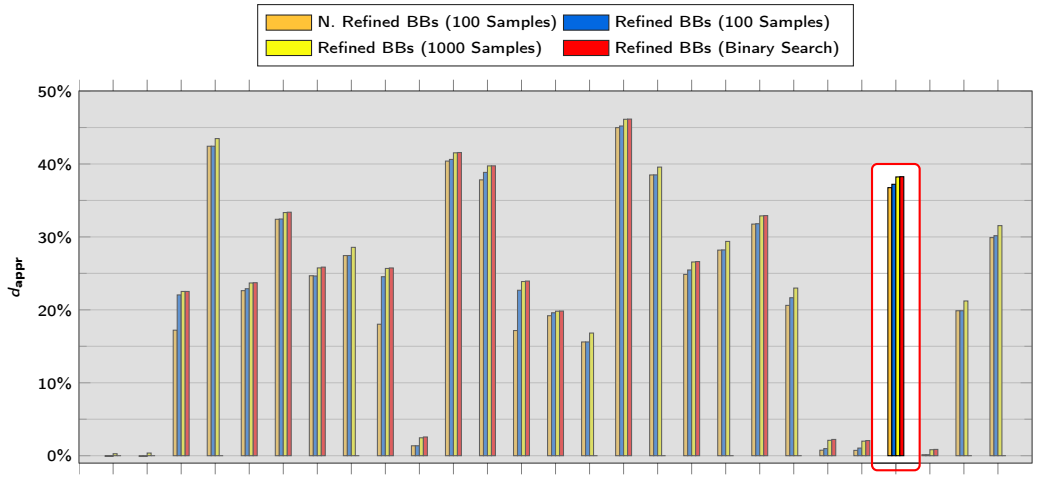
Overapproximation Metric



Evaluation Setup

- MRTC benchmark suite
  - `duff` benchmark excluded (due to timing analysis tool incompatibility)
- Evaluations performed on an Intel Xeon Server (20 cores at 2.3 GHz, 94 GB RAM)
- ILPs solved using Gurobi 7.5.0
- Compiled with the WCET-aware C compiler (WCC) using the `-O2` flag
- ARM7TDMI architecture (without caches)
- Data object access triggers an event


N. Refined BBs (100 Samples)
Refined BBs (100 Samples)
Refined BBs (1000 Samples)
Refined BBs (Binary Search)
50%
40%
30%
20%
10%
0%
d_{appr}
adpcm_dec
adpcm_enc
binarysearch
bsort100
compressdata
countnegative
crc
edn
fdct
fft1
fir
insertsort
jfdctint
lcdnum
lms
ludcmp
matmult
minver
ndes
ns
petrinet
qsort-exam
qurt
select
sqrt
st
statemate


N. Refined BBs (100 Samples)
Refined BBs (100 Samples)
Refined BBs (1000 Samples)
Refined BBs (Binary Search)
50%
40%
30%
20%
10%
0%
d_{appr}
adpcm_dec
adpcm_enc
binarysearch
bsort100
compressdata
countnegative
crc
edn
fdct
fft1
fir
insertsort
jfdctint
lcdnum
lms
ludcmp
matmult
minver
ndes
ns
petrinet
qsort-exam
qurt
select
sqrt
st
statemate

Overview

Conclusion

- Automated upper and lower event arrival function extraction from code-level analysis
- Two adjustable levels of granularity

Outlook

- Adding calling contexts
- Optimizations based on the extracted functions

Conclusion

- Automated upper and lower event arrival function extraction from code-level analysis
- Two adjustable levels of granularity

Outlook

- Adding calling contexts
- Optimizations based on the extracted functions

Conclusion

- Automated upper and lower event arrival function extraction from code-level analysis
- Two adjustable levels of granularity

Outlook

- Adding calling contexts
- Optimizations based on the extracted functions

Conclusion

- Automated upper and lower event arrival function extraction from code-level analysis
- Two adjustable levels of granularity

Outlook

- Adding calling contexts
- Optimizations based on the extracted functions

Conclusion

- Automated upper and lower event arrival function extraction from code-level analysis
- Two adjustable levels of granularity

Outlook

- Adding calling contexts
- Optimizations based on the extracted functions