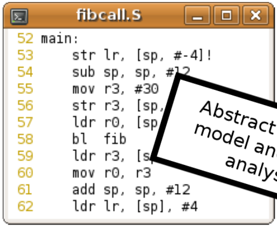
METAMOC: Modular Execution Time Analysis Using Model Checking

Mads Chr. Olesen <mchro@cs.aau.dk>
joint work with
Andreas Engelbredt Dalsgaard, Martin Toft,
René Rydhof Hansen, Kim Guldstrand Larsen

Aalborg University

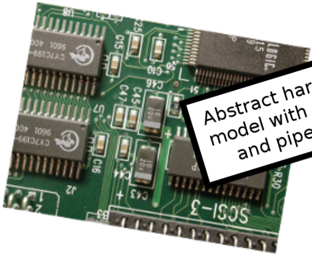
July 6th 2010

Overview of METAMOC



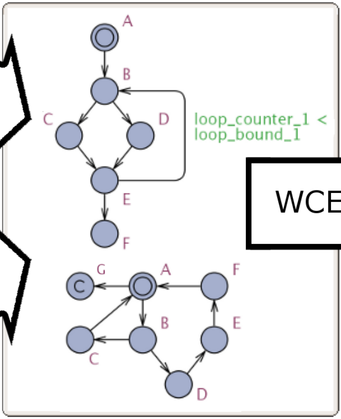
```
main:
        str lr, [sp, #-4]!
        sub sp, sp, #12
        mov r3, #30
        str r3, [sp,
        ldr r0, [sp
        bl  fib
        ldr r3, [s
        mov r0, r3
        add sp, sp, #12
        ldr lr, [sp], #4
```

Abstract process model and value analysis



Abstract hardware model with caching and pipelining

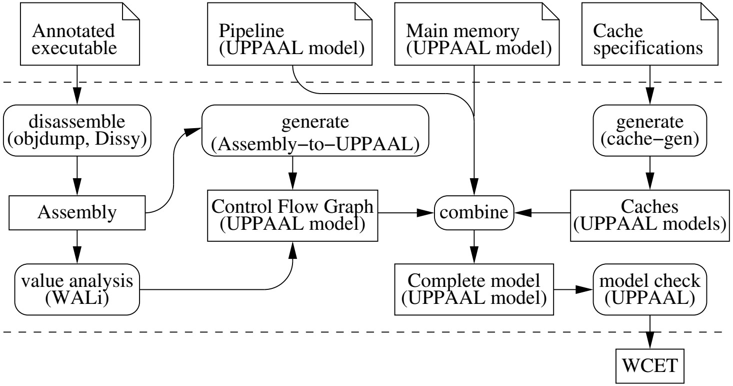
Timed automata models for hardware components and process functions:



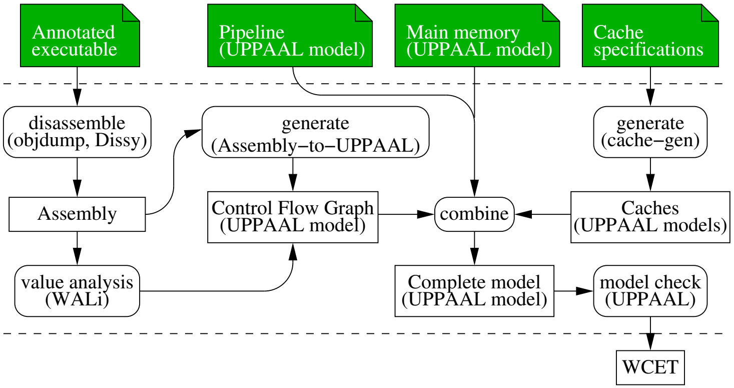
WCET

42 cycles

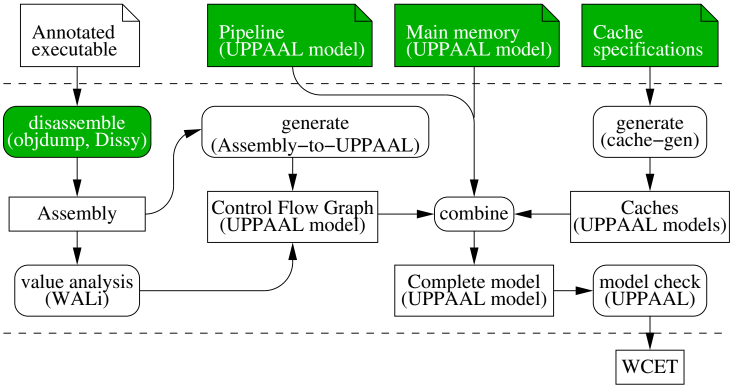
Overview of METAMOC



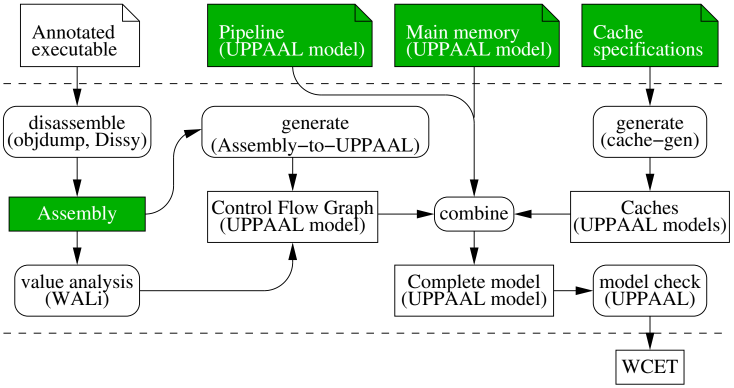
Overview of METAMOC



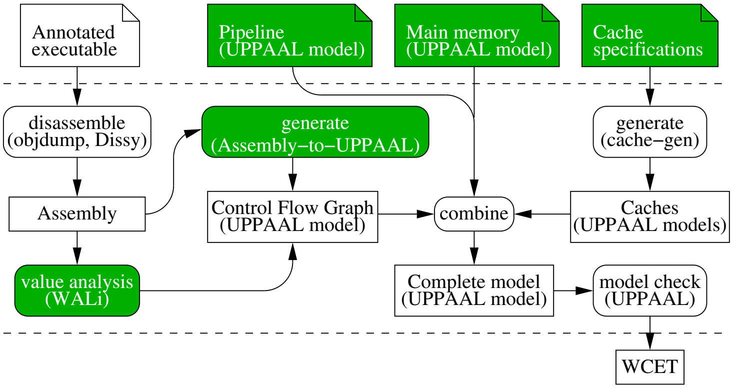
Overview of METAMOC



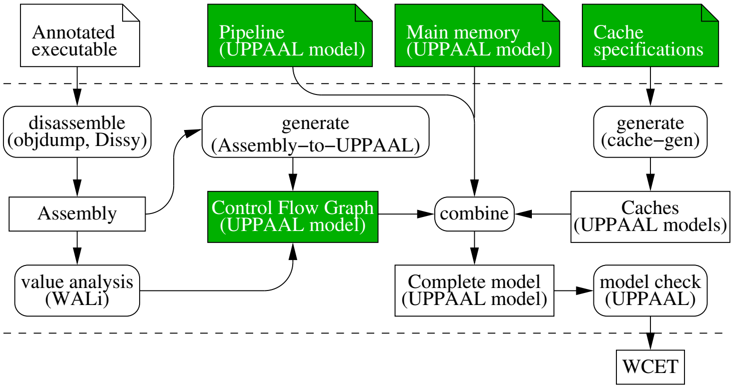
Overview of METAMOC



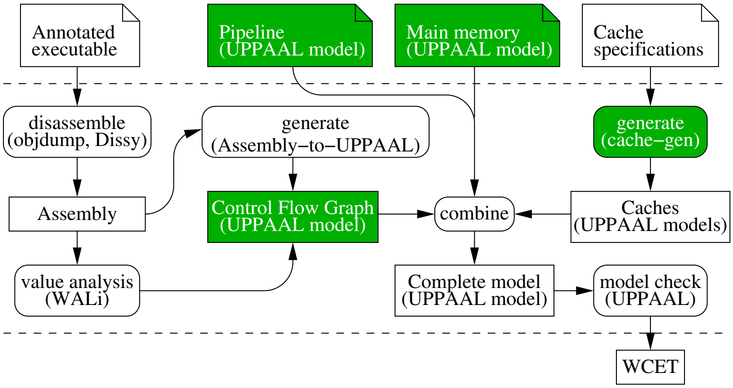
Overview of METAMOC



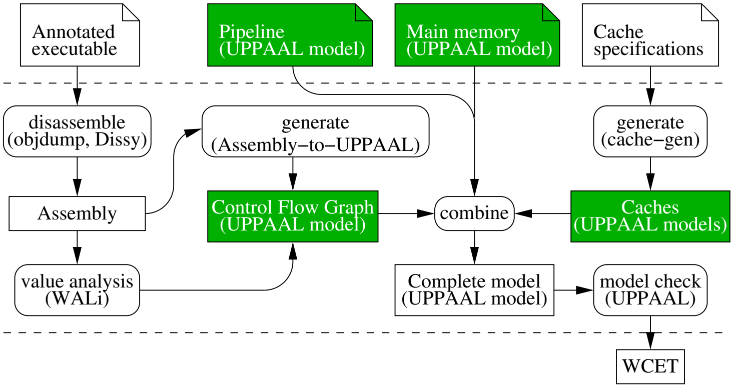
Overview of METAMOC



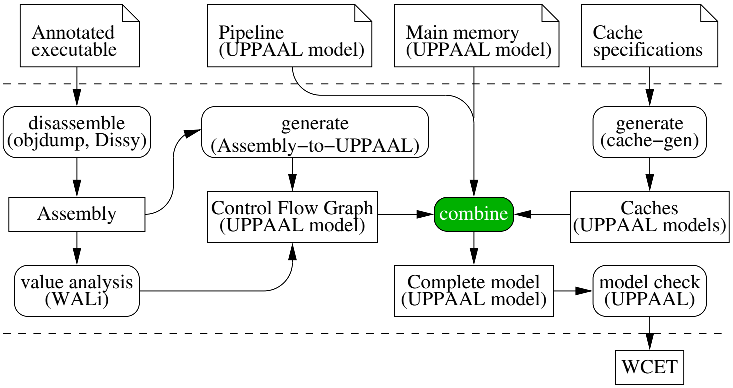
Overview of METAMOC



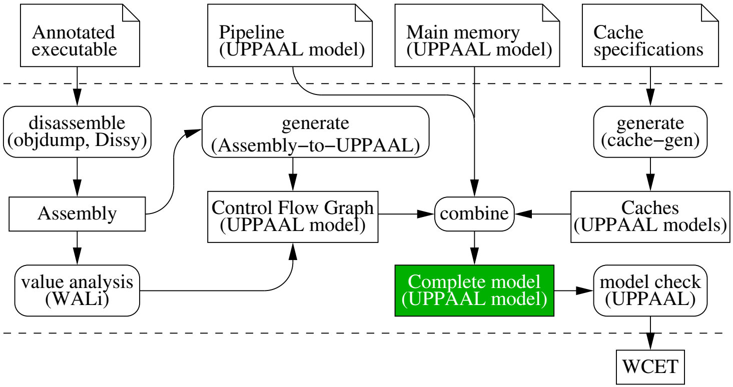
Overview of METAMOC



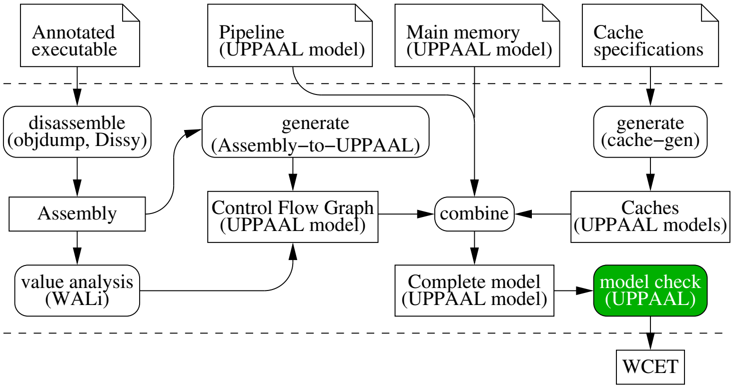
Overview of METAMOC



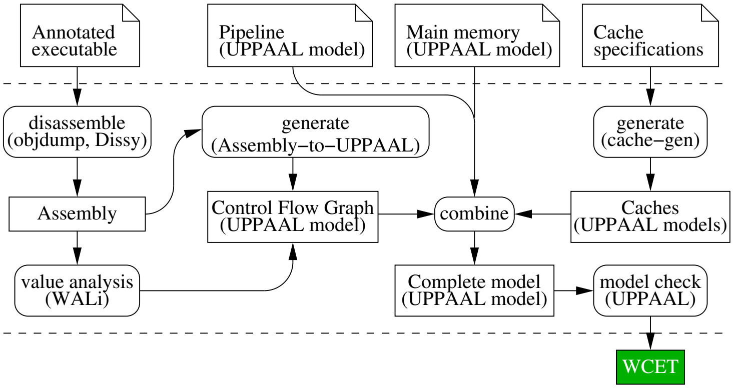
Overview of METAMOC



Overview of METAMOC



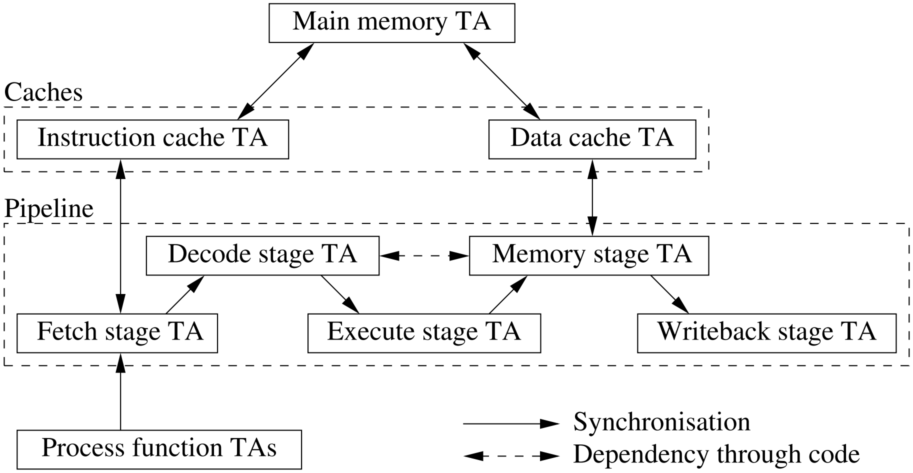
Overview of METAMOC



Current Work

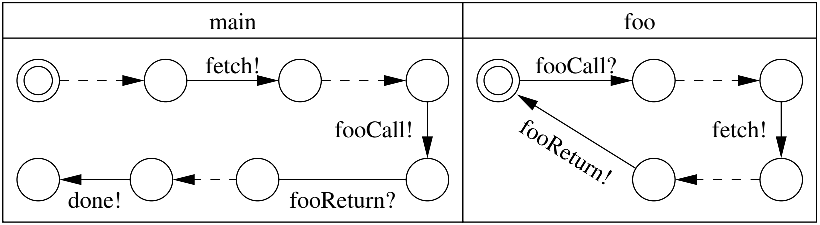
- Support for pipelines
  - ARM9TDMI
  - ARM7TDMI
  - ATMEL AVR 8-BIT
- Support for instruction/data caches
  - Automatically generated
  - LRU/FIFO replacement policy
- Value analysis for predicting memory accesses
  - Implemented using Weighted Push-Down Systems
  - Inter-procedural
  - Currently syntactic constant-propagation
- Timing anomalies cannot be (consistently) handled
  - Experiments with caches are with LRU caches, not FIFO as on the real ARM9

Modelling in METAMOC



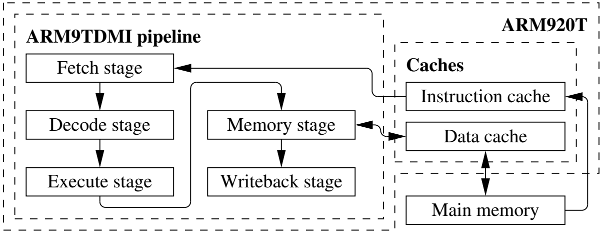
Overview of the ARM9 automata

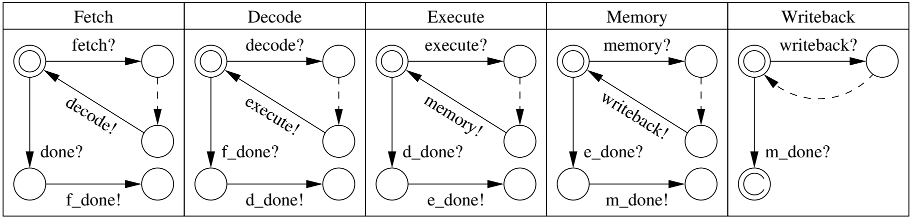
Modelling in METAMOC



Sketch of the function automata for a process

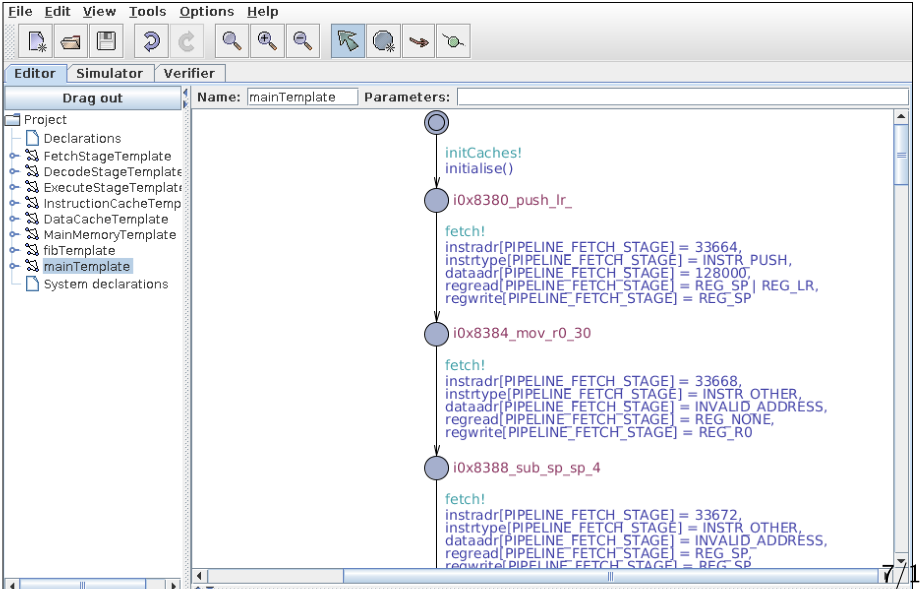
Modelling in METAMOC





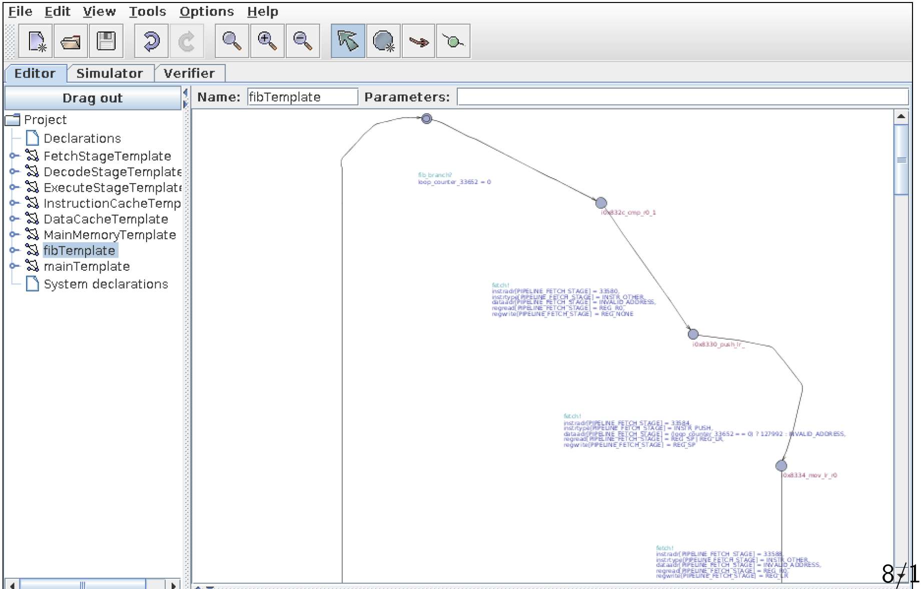
ARM9 overview and sketch of pipeline automata

UPPAAL



```
initCaches!
initialise()

i0x8380_push_lr_

fetch!
instradr[PIPELINE_FETCH_STAGE] = 33664,
instrtype[PIPELINE_FETCH_STAGE] = INSTR_PUSH,
dataadr[PIPELINE_FETCH_STAGE] = 128000,
regread[PIPELINE_FETCH_STAGE] = REG_SP | REG_LR,
regwrite[PIPELINE_FETCH_STAGE] = REG_SP

i0x8384_mov_r0_30

fetch!
instradr[PIPELINE_FETCH_STAGE] = 33668,
instrtype[PIPELINE_FETCH_STAGE] = INSTR_OTHER,
dataadr[PIPELINE_FETCH_STAGE] = INVALID_ADDRESS,
regread[PIPELINE_FETCH_STAGE] = REG_NONE,
regwrite[PIPELINE_FETCH_STAGE] = REG_R0

i0x8388_sub_sp_sp_4

fetch!
instradr[PIPELINE_FETCH_STAGE] = 33672,
instrtype[PIPELINE_FETCH_STAGE] = INSTR_OTHER,
dataadr[PIPELINE_FETCH_STAGE] = INVALID_ADDRESS,
regread[PIPELINE_FETCH_STAGE] = REG_SP,
regwrite[PIPELINE_FETCH_STAGE] = REG_SP
```

UPPAAL



Name: fibTemplate Parameters:

```
fib_branch?
loop_counter_33652 = 0
```

```
i0x832c_cmp_r0_1
```

```
fetch!
instradr[PIPELINE_FETCH_STAGE] = 33580,
instrtype[PIPELINE_FETCH_STAGE] = INSTR_OTHER,
dataadr[PIPELINE_FETCH_STAGE] = INVALID_ADDRESS,
regread[PIPELINE_FETCH_STAGE] = REG_R0,
regwrite[PIPELINE_FETCH_STAGE] = REG_NONE
```

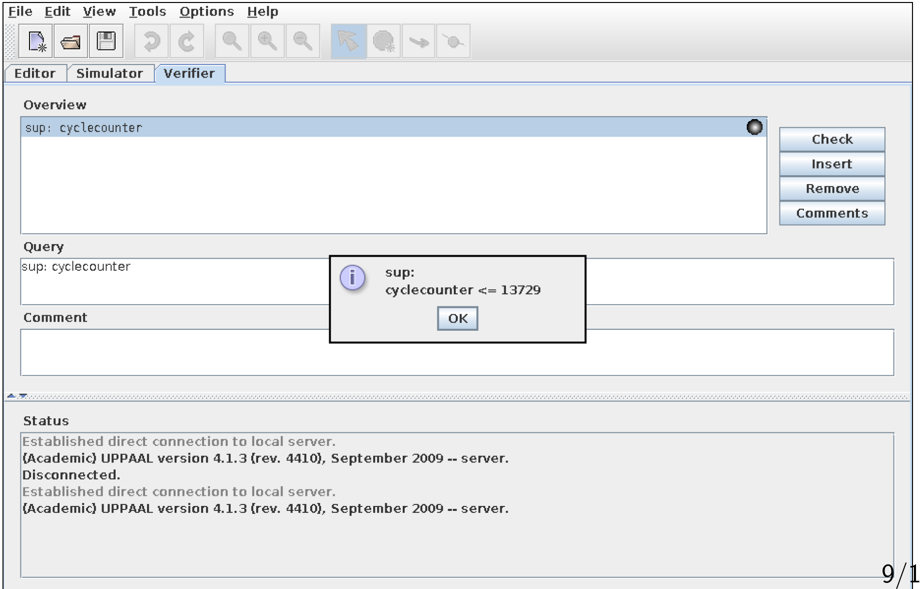
```
i0x8330_push_lr_
```

```
fetch!
instradr[PIPELINE_FETCH_STAGE] = 33584,
instrtype[PIPELINE_FETCH_STAGE] = INSTR_PUSH,
dataadr[PIPELINE_FETCH_STAGE] = (loop_counter_33652 = = 0) ? 127992 : INVALID_ADDRESS,
regread[PIPELINE_FETCH_STAGE] = REG_SP | REG_LR,
regwrite[PIPELINE_FETCH_STAGE] = REG_SP
```

```
0x8334_mov_lr_r0
```

```
fetch!
instradr[PIPELINE_FETCH_STAGE] = 33588,
instrtype[PIPELINE_FETCH_STAGE] = INSTR_OTHER,
dataadr[PIPELINE_FETCH_STAGE] = INVALID_ADDRESS,
regread[PIPELINE_FETCH_STAGE] = REG_R0,
regwrite[PIPELINE_FETCH_STAGE] = REG_LR
```

UPPAAL



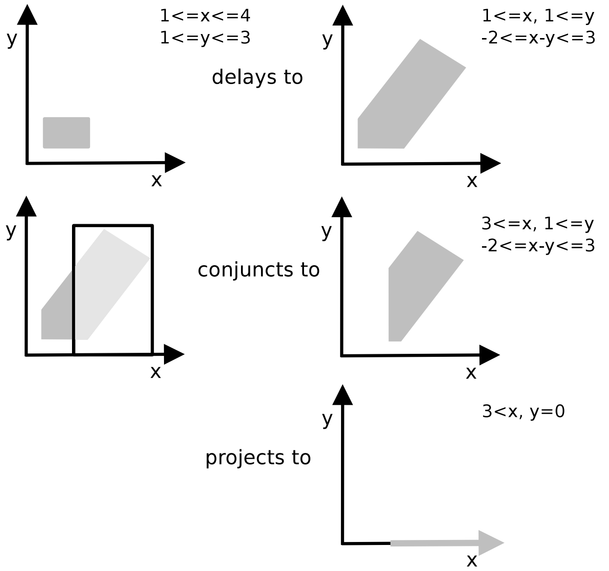
sup: cyclecounter

sup: cyclecounter

sup:
cyclecounter <= 13729

```
Established direct connection to local server.
(Academic) UPPAAL version 4.1.3 (rev. 4410), September 2009 -- server.
Disconnected.
Established direct connection to local server.
(Academic) UPPAAL version 4.1.3 (rev. 4410), September 2009 -- server.
```

UPPAAL Zones



UPPAAL Zones

- Delay is cheap - large zones
  - Resilient to different memory wait delays
- Many small steps expensive - smaller zones
- Zones can be collapsed, overapproximation

Eliminating non-determinism

- Since no timing anomalies, cut down on the number of distinct paths as much as possible
- Pigeonhole optimisations
  - Iterate loops the maximum number of times
  - Don't forward jump if path is subset of not jumping
- "Executing more code increases the execution time"
- Can be disabled if timing anomalies present

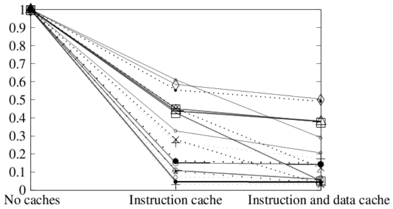
Experiments

- Evaluation using WCET benchmark programs from Mälardalen Real-Time Research Centre
  - Applicability
  - Performance
- Discarded a number of programs
  - Floating point operations handled by software routines
  - Dynamic jumps
  - Some programs do not compile for our architectures
- 21 programs for ARM and 19 programs for AVR
- Manually annotated loop bounds

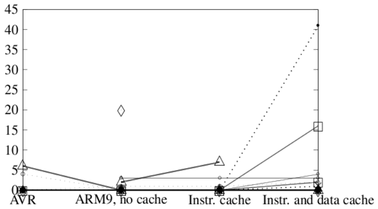
Experiments

ARM9 w. LRU caches, 21 benchmarks	
Analysable without caches	21
Analysable with instruction cache	20
Unanalysable, state space explosion	1
Manual modification of instruction cache size	1
Analysable with data and instruction cache	19
Unanalysable, state space explosion	2
Manual modification of data cache size	2
Manual syntax fix of model	1

ATMEL AVR 8-bit, 19 benchmarks	
Analysable	16
Unanalysable, state space explosion	3



Relative improvement in WCET for ARM9.



Analysis times in minutes for AVR and ARM9.

Future Work

- Improvements in model checker technology
    - Our models atypical: more deterministic, longer paths, larger
    - Summarizing long deterministic paths - "short-cuts"
    - Parallel/Distributed model checking
    - Guiding the search - A*
- Data sensitivity/flow facts
    - Track values of registers in model
- Timing anomalies
    - Introduces more non-determinism
    - Improving model checker technology
- Schedulability instead of WCET analysis
    - SARTS project has done this for Java bytecode on the JOP processor

Thank you for your attention!

Questions?

`http://metamoc.dk`