WCET Analysis of ARM Processors using Real-Time Model Checking

Andreas Engelbredt Dalsgaard Mads Christian Olesen
Martin Toft René Rydhof Hansen Kim Guldstrand Larsen

{andrease,mchro,mt,rrh,kgl}@cs.aau.dk

Department of Computer Science
Aalborg University
Denmark

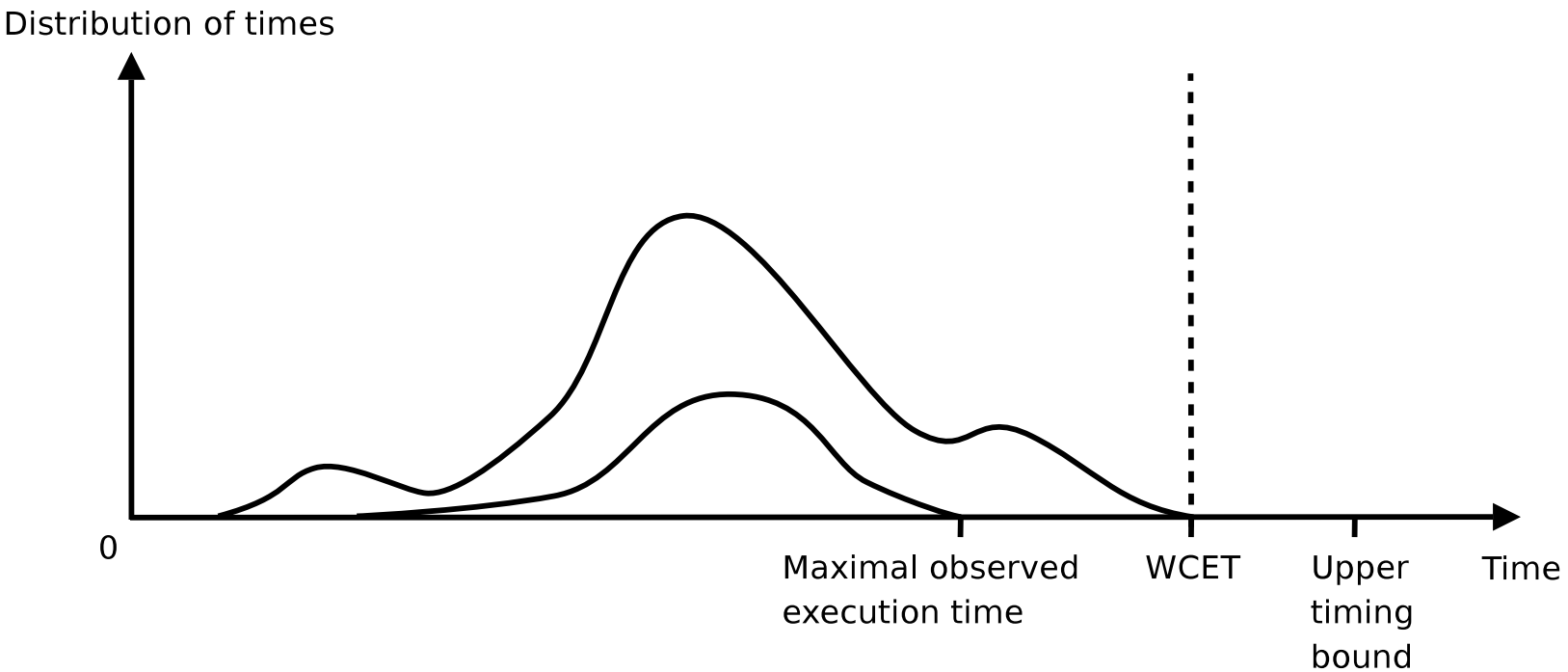
June 22, 2009

Our Contribution

- **Modular** method for finding Worst-Case Execution Times
- Handles a **real-world** modern processor
- Tested on **real** programs; the Mälardalen benchmark programs
- **Efficient** implementation

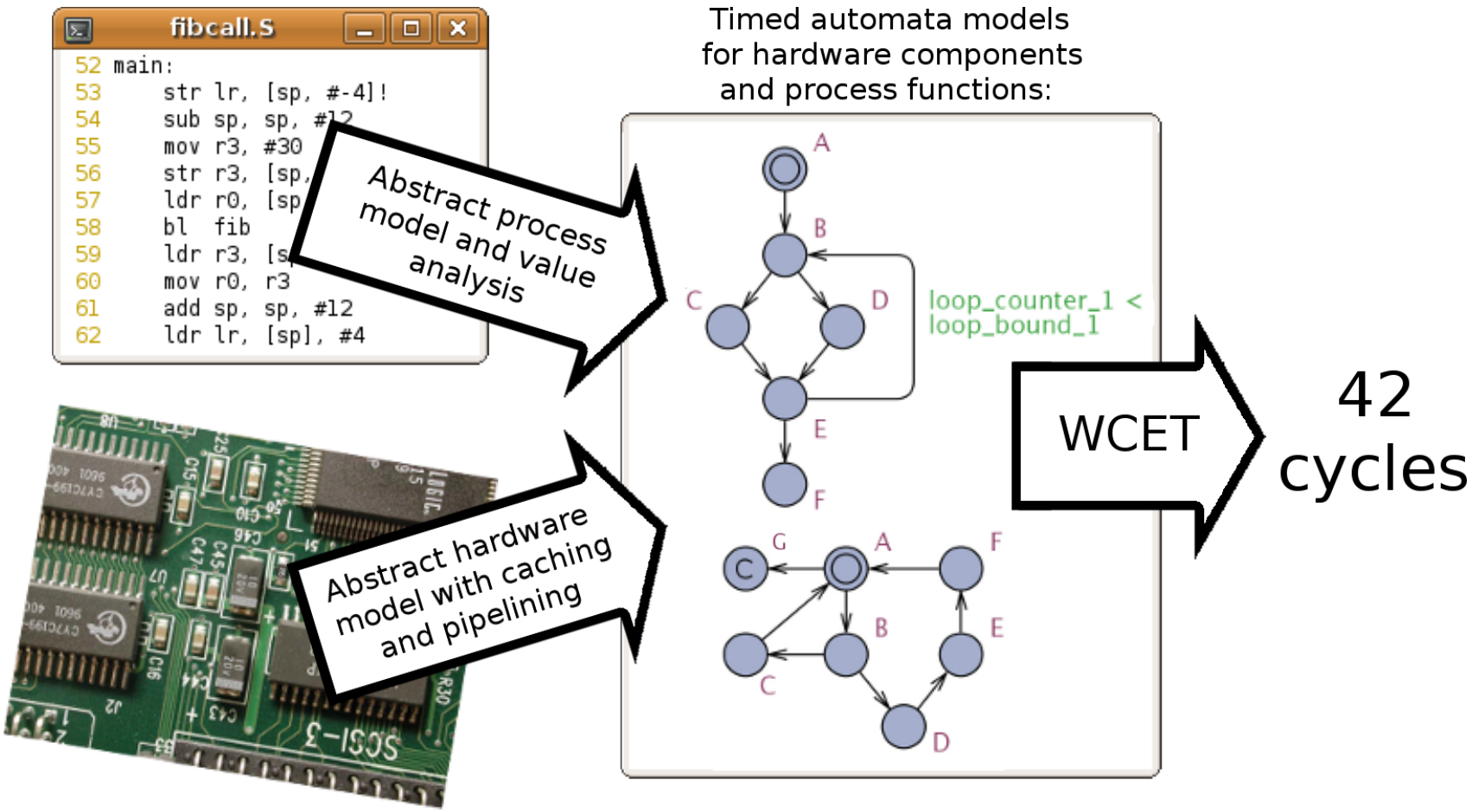
Introduction

- RTSs need WCETs for all processes, for reliable scheduling



- WCETs need to be approximated: Overapproximation, but not pessimistic

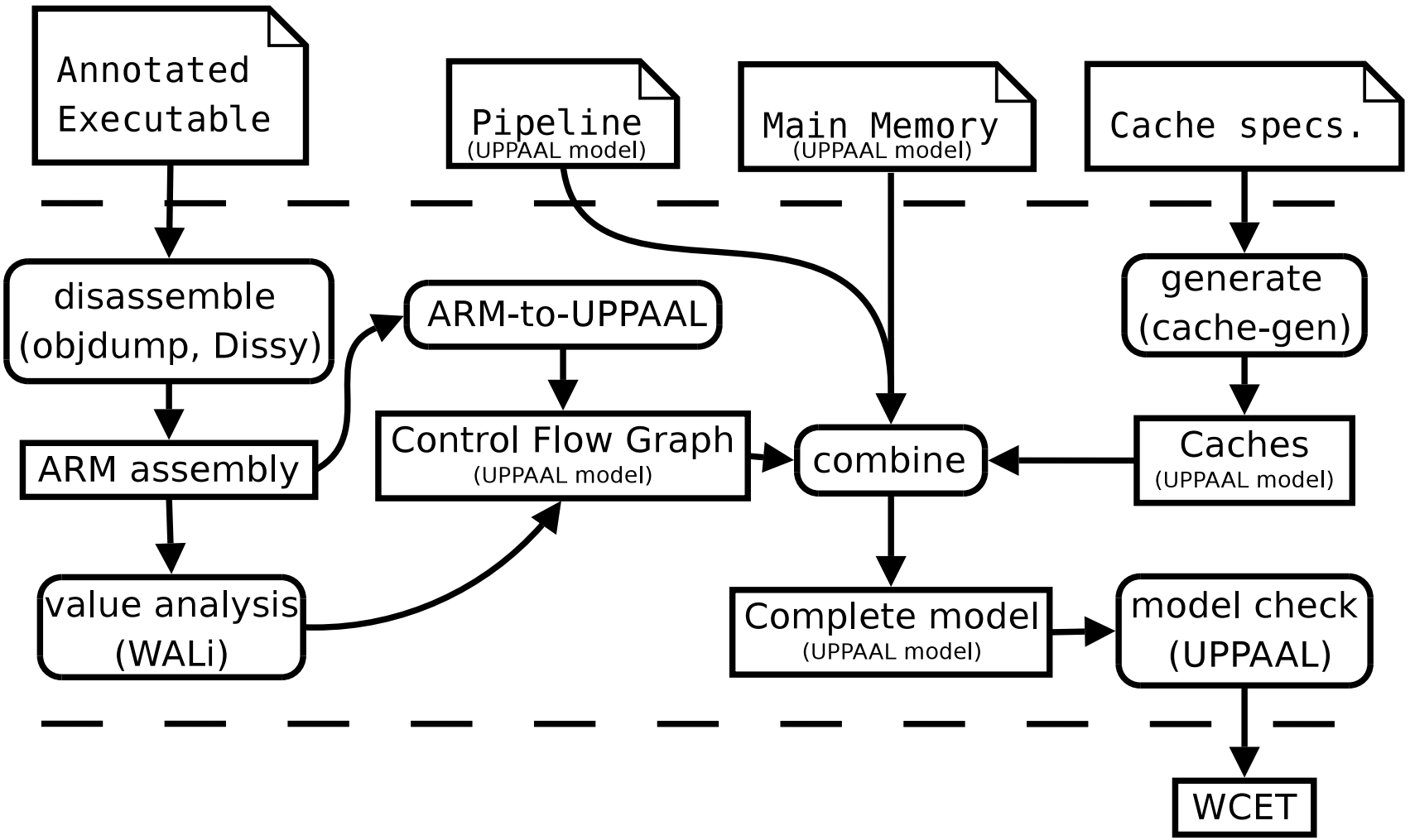
The METAMOC Method



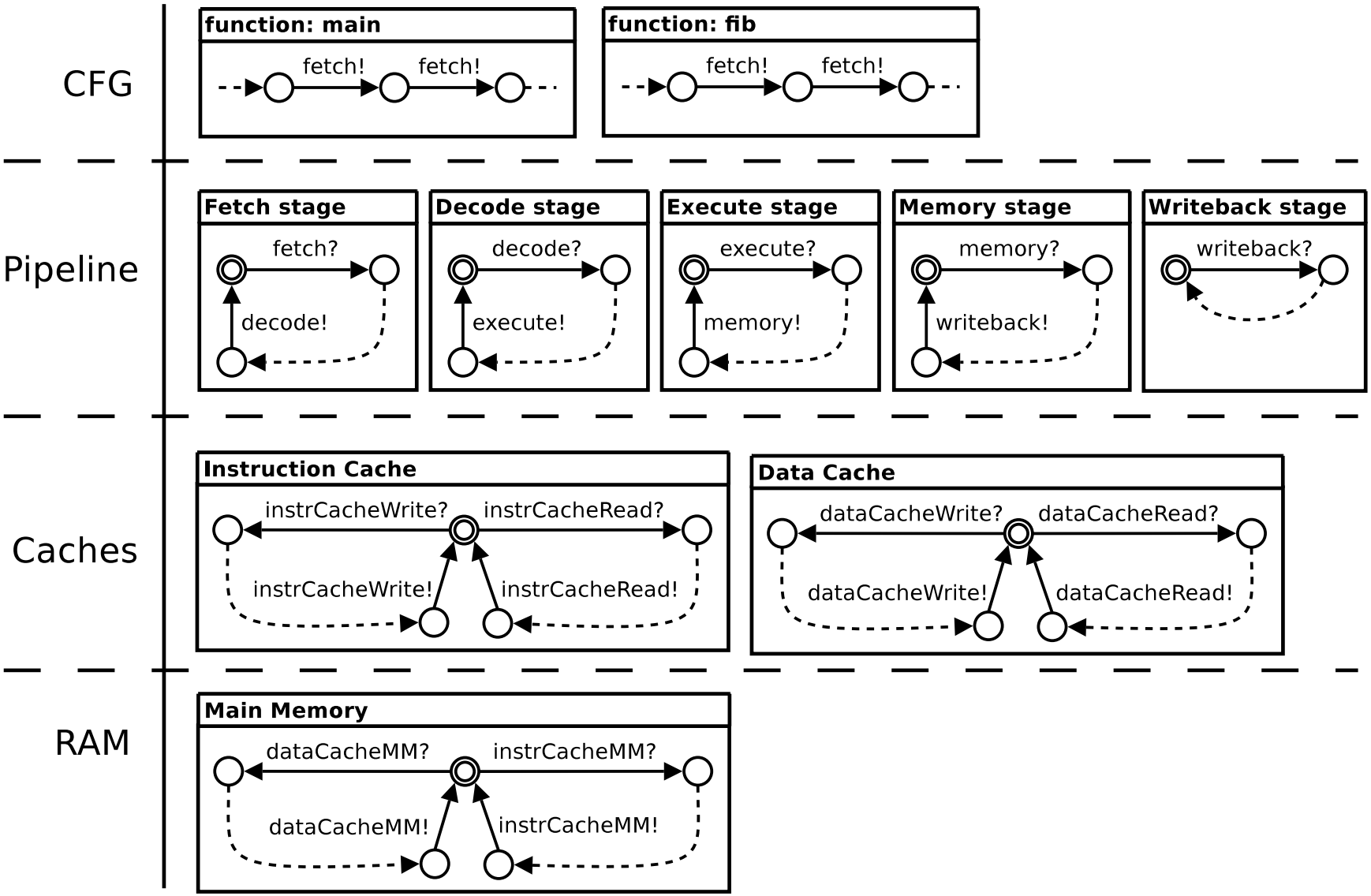
Prototype Implementation

- ARM9TDMI processor core
  - ARM920T
  - ARM922T
  - ARM940T
- Five stage pipeline
- Separate instruction and data caches
- Does not suffer from timing anomalies
  - Assume local worst-case

Prototype Implementation



Prototype Implementation

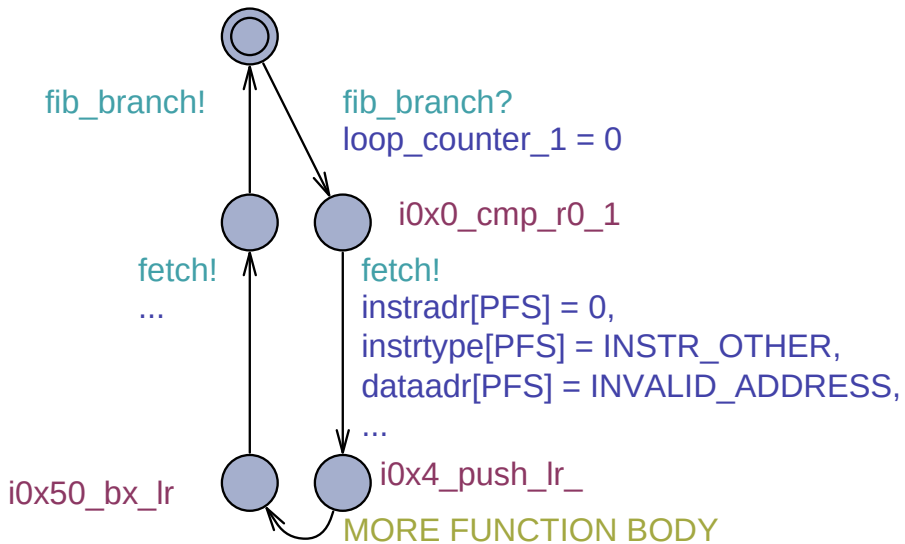


Path Analysis

- Reconstruct CFG from binary
- Construct path model based on CFG
- Combine with pipeline, cache and main memory models
- Model check combined timed automata
  - `sup: cyclecounter`

Path Analysis

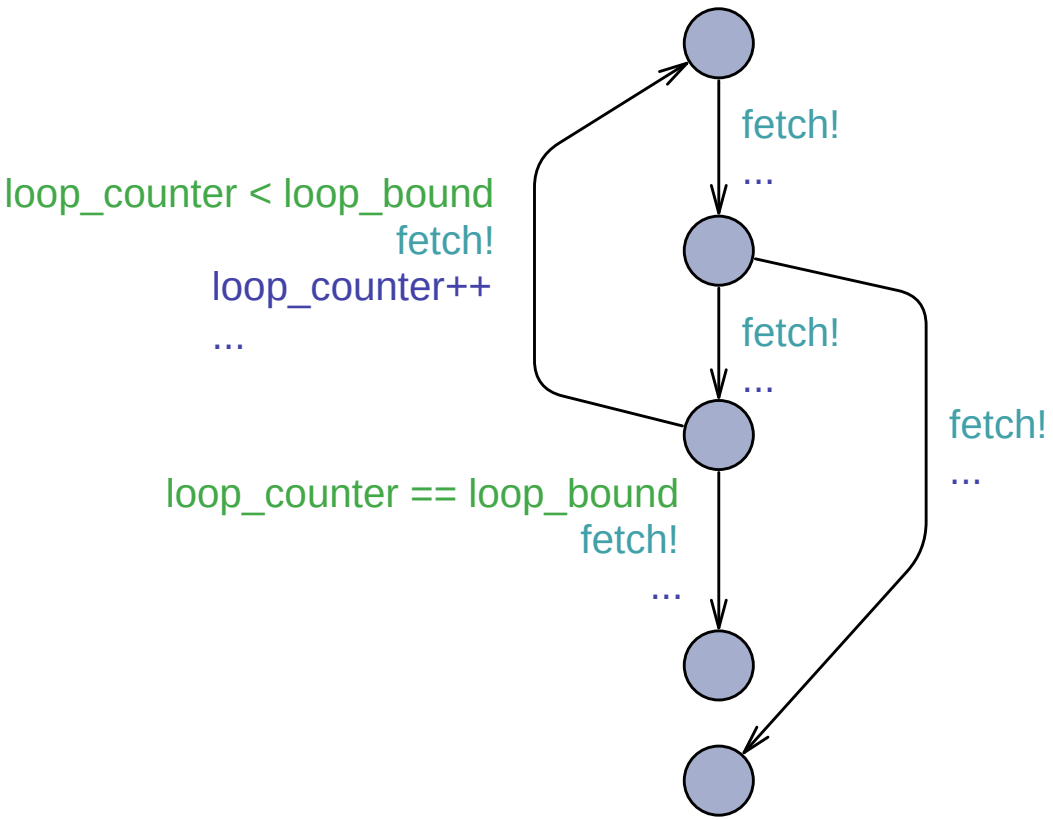
- Timed automaton for every function
- Transitions emulate instruction execution



- Functions handled flow-sensitively

Path Analysis

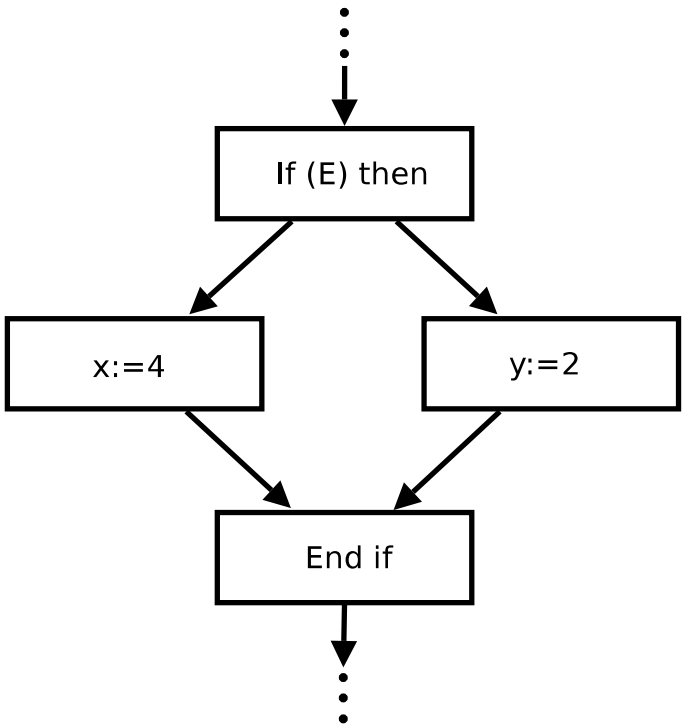
- Assembly level jumps



Cache Analysis

- Concrete cache model
  - Unknown memory block
  - Write allocate/Write back
  - No write allocate/Write through
  - Replacement policy
  - Size parameters
- Always miss cache model
- Abstract cache model
  - Abstract cache analysis like Wilhelm et al.

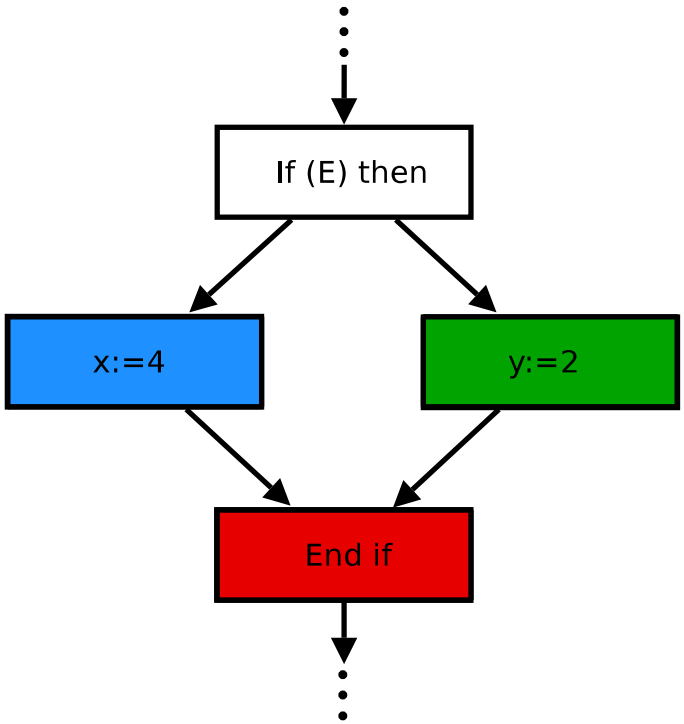
Abstract Cache Analysis

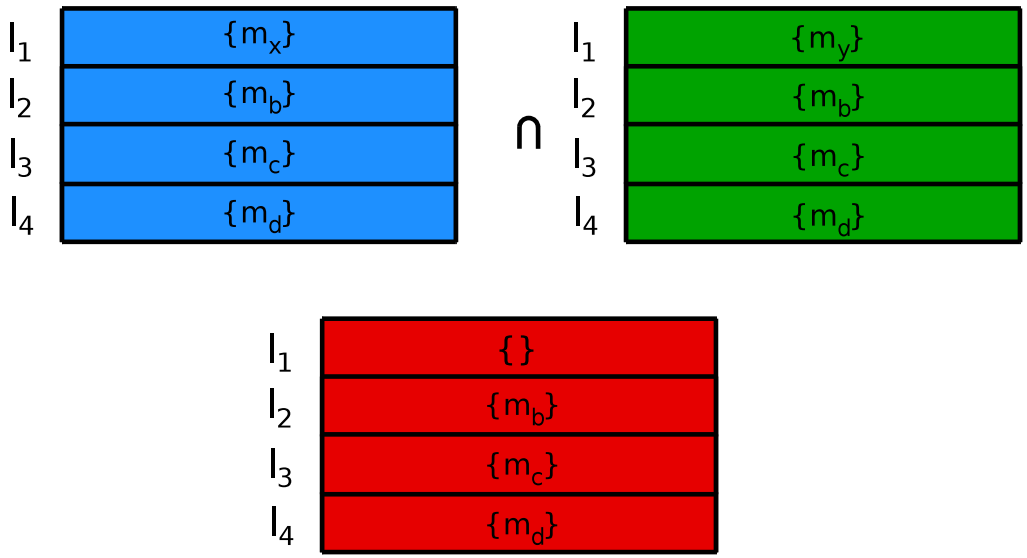


- Avoid non-determinism
- Smaller state space

- Calculate which memory blocks MUST be in the cache at a CFG-node
- At join-points merge the results from the predecessors

Abstract Cache Analysis



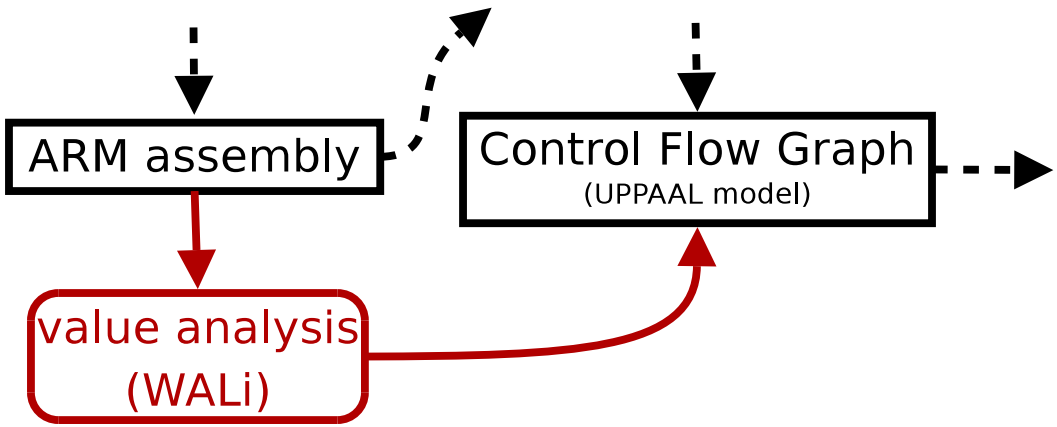


Abstract Cache Analysis

- Implemented using model checking
  - Clocks and stop watches
    - Already done
    - Works well with FIFO replacement policy
    - Not integrated (difficult)
  - New data type in UPPAAL

Value Analysis

- The cache analysis needs concrete memory addresses
- Registers are used as base and offset in many memory accesses



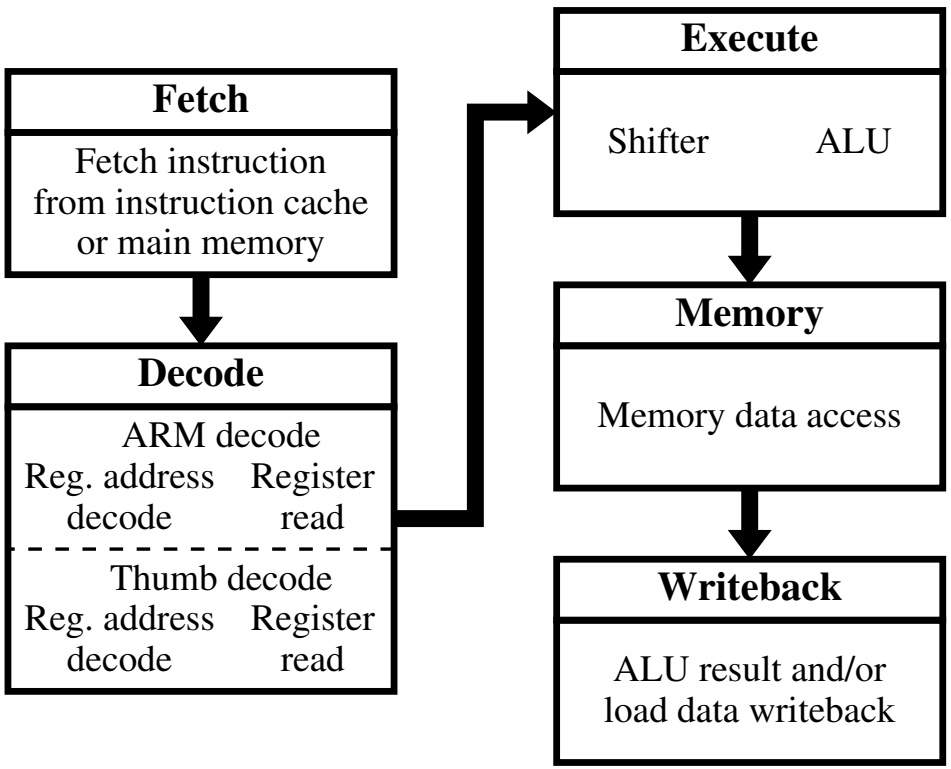
- Value analysis:
  Find an overapproximation of possible register values at all execution points of a process
- Weighted push-down systems (WPDSs) used for inter-procedural, control-flow sensitive value analysis
- Presented by Reps et al. in
  Program Analysis using Weighted Push-Down Systems

Value Analysis

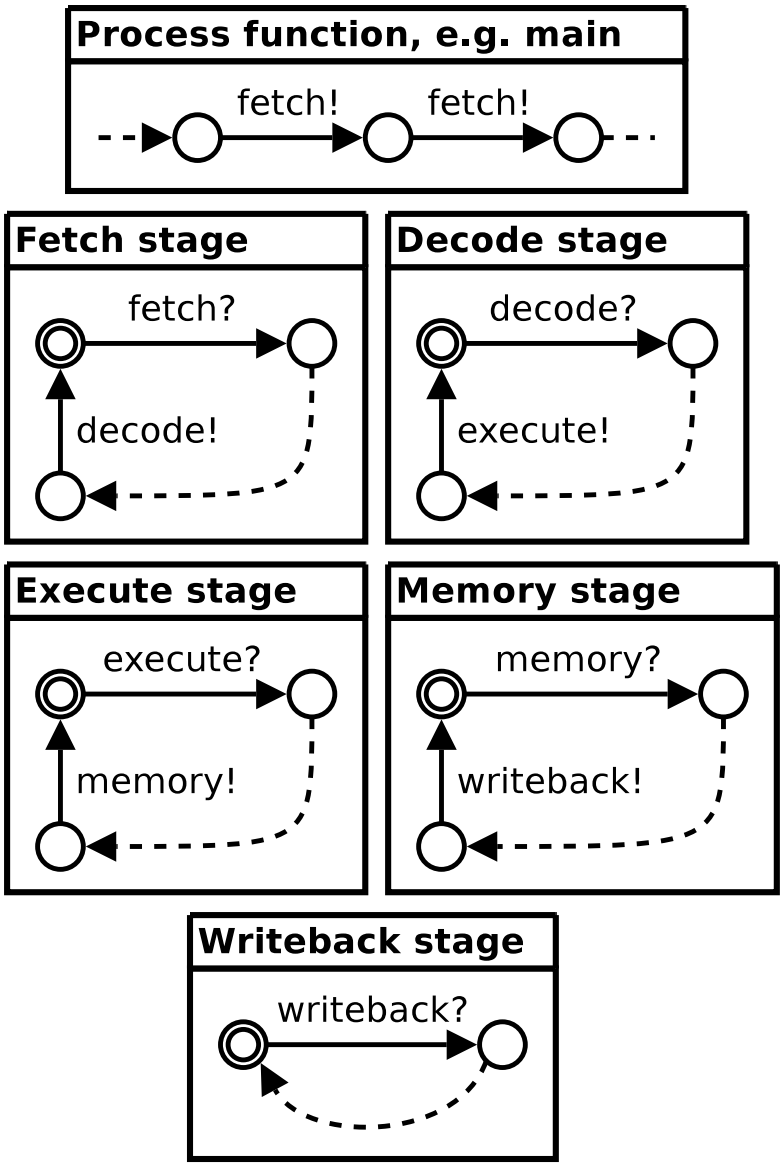
- WPDS: Push-down system (PDS), weight domain, and mapping between PDS rules and weight domain elements
- Weighted Automata Library (WALi) implements a number of WPDS algorithms
- WPDSs allow taking the **inter-procedural** control-flow into account
- Implemented simple value analysis, using:
  - Loop unrolling
  - Simple register-value tracking
  - No tracking of values in memory
  - Finds good amount of values for some programs, but could be much better

Pipeline Analysis

- Pipeline analysis:
  Take the effect of pipelining into account in order to determine sharper WCETs
- Five stages in the ARM9TDMI processor core
- Stalls due to inter-dependencies



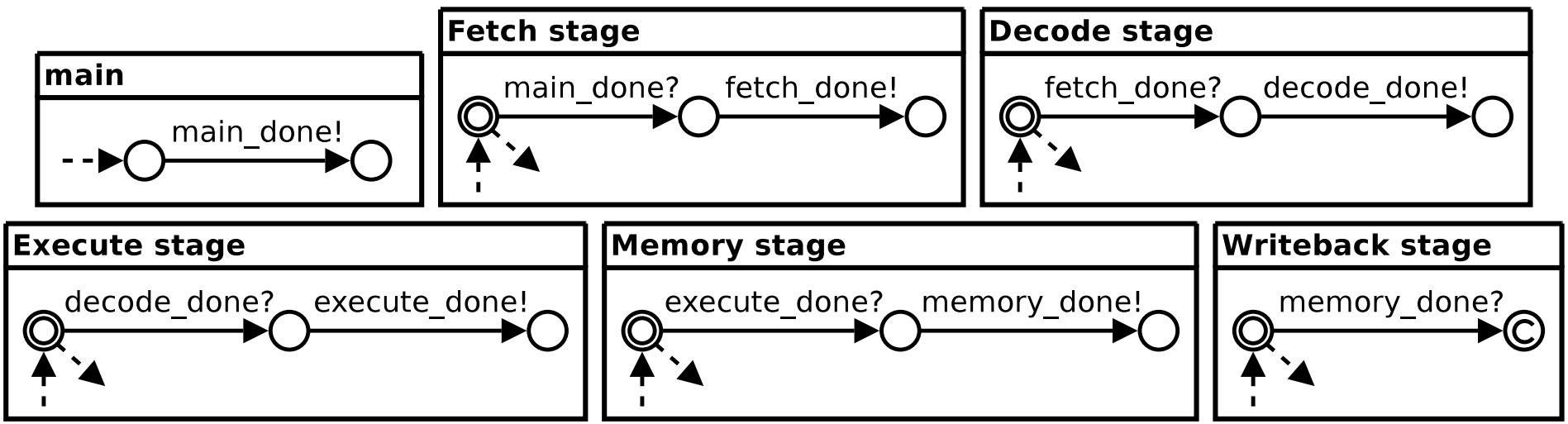
Pipeline Analysis



- Modelled as a network of timed automata in UPPAAL
- Synchronisation between function automata and the fetch stage automaton
- Synchronisation between stage automata for the instructions to "flow" through the data path
- Cyclic stage automata

Pipeline Analysis

- Time must be bounded for `sup: cyclecounter` to give non-trivial guarantees
- A signaling system is needed



Experiments

- Conducted on the concrete implementation for the ARM920T processor
- Examine three qualities:
  - Size and complexity of processes
  - How much sharper WCETs are found by taking caching into account
  - Resource usage (time and memory)
- No evaluation of the pipeline
- No reference WCETs available
- Benchmark programs from the WCET Analysis Project by Mälardalen Real-Time Research Center
  - Wide selection of computation tasks
  - Used to benchmark WCET analysis methods

Experiments

- The most interesting findings:
  - Taking the **instruction cache** into account yields WCETs that are up to 97% sharper (78% on average at -O2)
  - Taking the **data cache** into account yields WCETs that are up to 68% sharper (31% on average at -O2)
  - Almost all results are obtained within five minutes

Experiments

- The most interesting findings:
  - Taking the **instruction cache** into account yields WCETs that are up to 97% sharper (78% on average at -O2)
  - Taking the **data cache** into account yields WCETs that are up to 68% sharper (31% on average at -O2)
  - Almost all results are obtained within five minutes
- Some programs fail due to
  - State space explosion (9)
  - Write to program counter (2)
  - Floating point operations
  - Value analysis problems

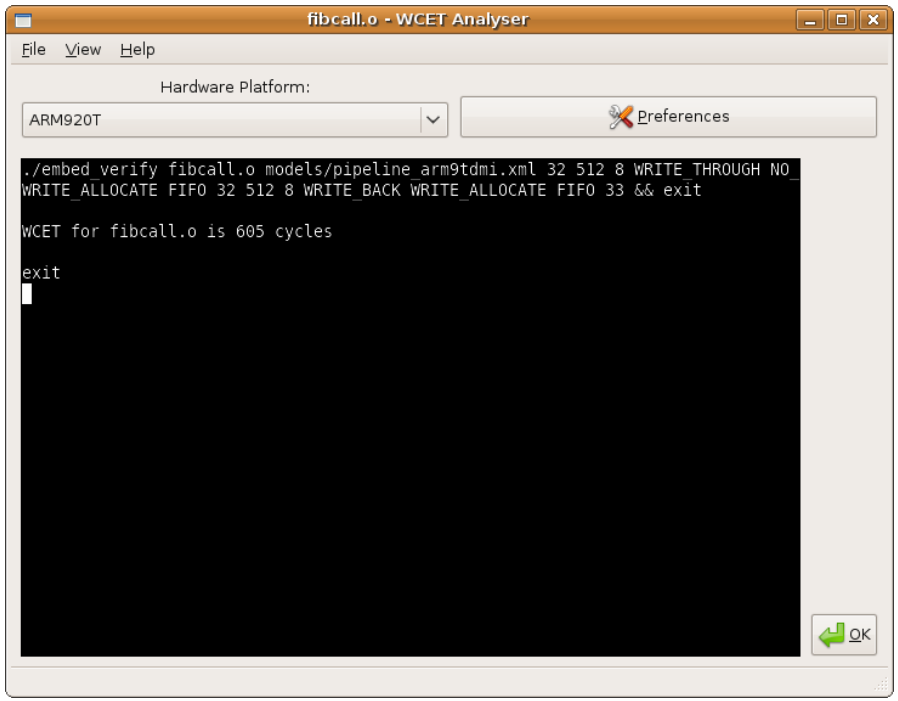
Experiments

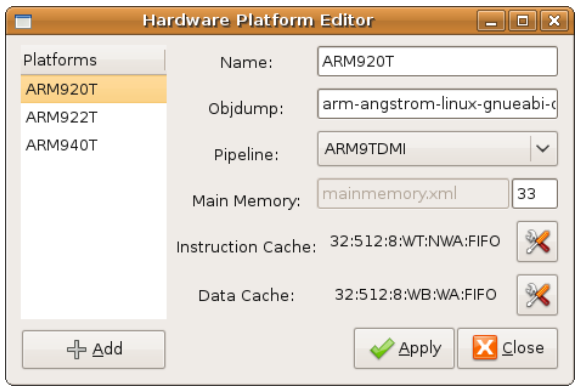
- The most interesting findings:
  - Taking the **instruction cache** into account yields WCETs that are up to 97% sharper (78% on average at -O2)
  - Taking the **data cache** into account yields WCETs that are up to 68% sharper (31% on average at -O2)
  - Almost all results are obtained within five minutes
- Some programs fail due to
  - State space explosion (9)
  - Write to program counter (2)
  - Floating point operations
  - Value analysis problems
- **We are able to analyse 14 out of the 25 non-floating point benchmarks!**

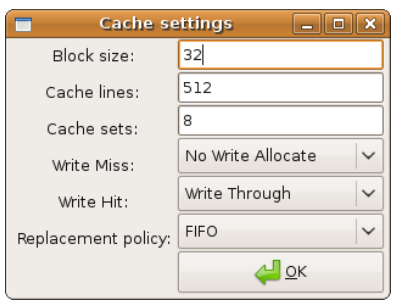
Future Work

- Integration of abstract caches
- Improve the path analysis
- Better value analysis
- Explore other ways to model the hardware platform
- Support for floating point operations
- Support for other hardware architectures
- Incorporate schedulability analysis
  - Reducing schedulability analysis to reachability like in the *Schedulability Analyzer for Real-Time Systems (SARTS)* tool by Bøgholm et al.

Further Information







- The extended abstract, our master's thesis, the accompanying source code, and these slides are available at

  http://metamoc.martintoft.dk

- Questions?

THE END

THE END