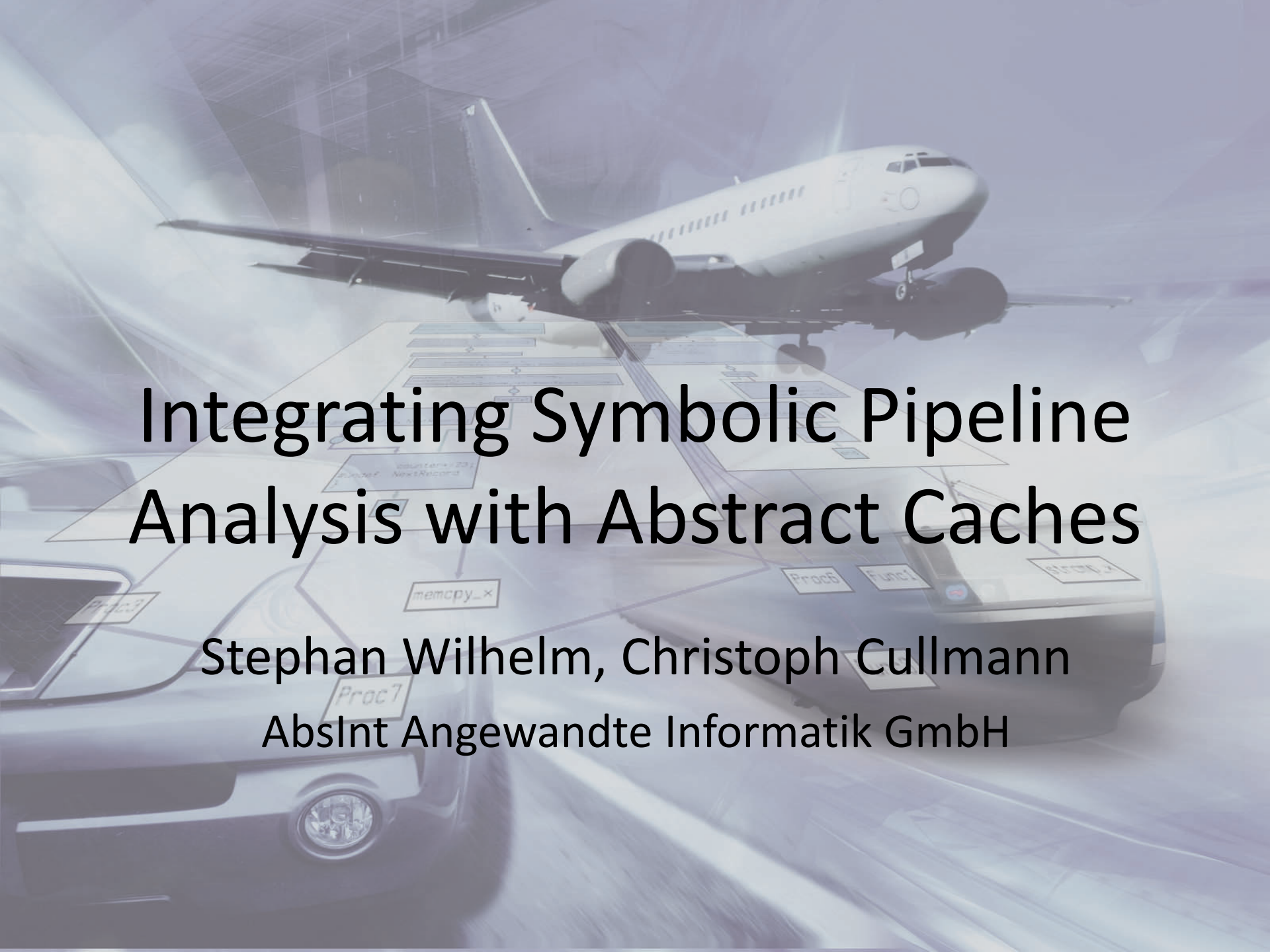


Integrating Symbolic Pipeline Analysis with Abstract Caches

Stephan Wilhelm, Christoph Cullmann

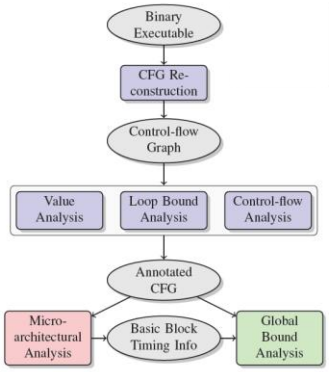
AbsInt Angewandte Informatik GmbH

Static WCET Analysis

- **Computes**: Safe upper bounds on WCET of a task
  - safety critical applications, hard WCET

- **Requires**: Full coverage of
  - feasible program paths
  - inputs
  - hardware states (pipeline, caches)

- **Method**: AI + ILP [aiT WCET analyzer]
  - abstract interpretation
  - integer linear programming
  - several interacting analyses



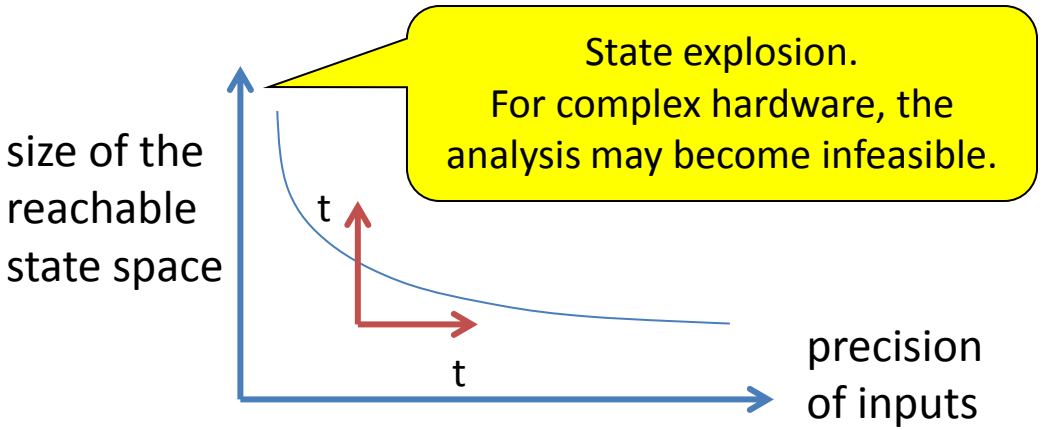
Micro-architectural Analysis

- **Pipeline** analysis
  - abstract model
    - pipelining, speculation, branch prediction
  - computes an over-approximates reaching states
  - counts cycles on the basic block level
- **Cache** analysis
  - computes an over-approximation of possible cache contents
  - predicts sure hit or miss, or don't know
- Analysis **inputs**
  - from control flow and value analysis
  - branch and call targets, possible register contents
  - all inputs may be *imprecise*

Imprecise Inputs

- **Intervals and sets** instead of precise values for
  - possible register contents
  - call and branch targets

- **Effect** on hardware-level analysis
  - must consider more possibilities



Efficient State Space Coverage

Cache analysis

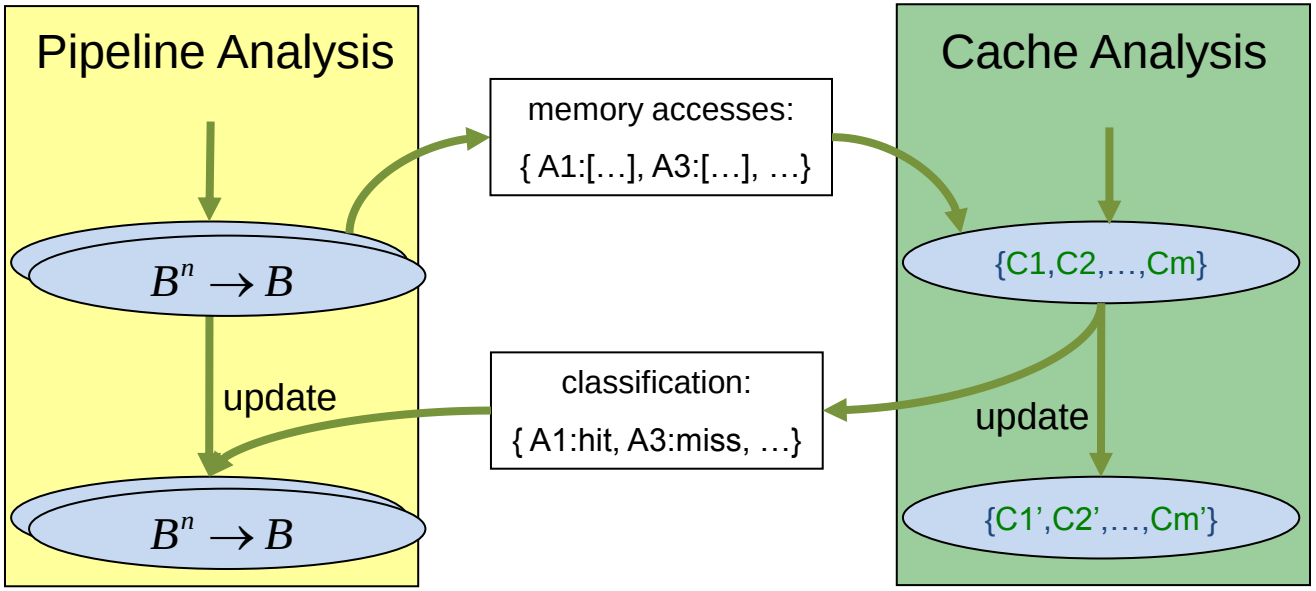
- Abstract interpretation
- Efficient join operator
  - safe over-approximation
  - loses precision

- Efficient **abstract** representation

Pipeline analysis

- Enumerates all states
  - Timing anomalies
  - Domino effects
- Symbolic representation
  - Implicit, Boolean functions
  - Binary decision diagrams
  - Exploits redundancies
- Efficient i**mplicit** representation

Pipeline-Cache-Interaction

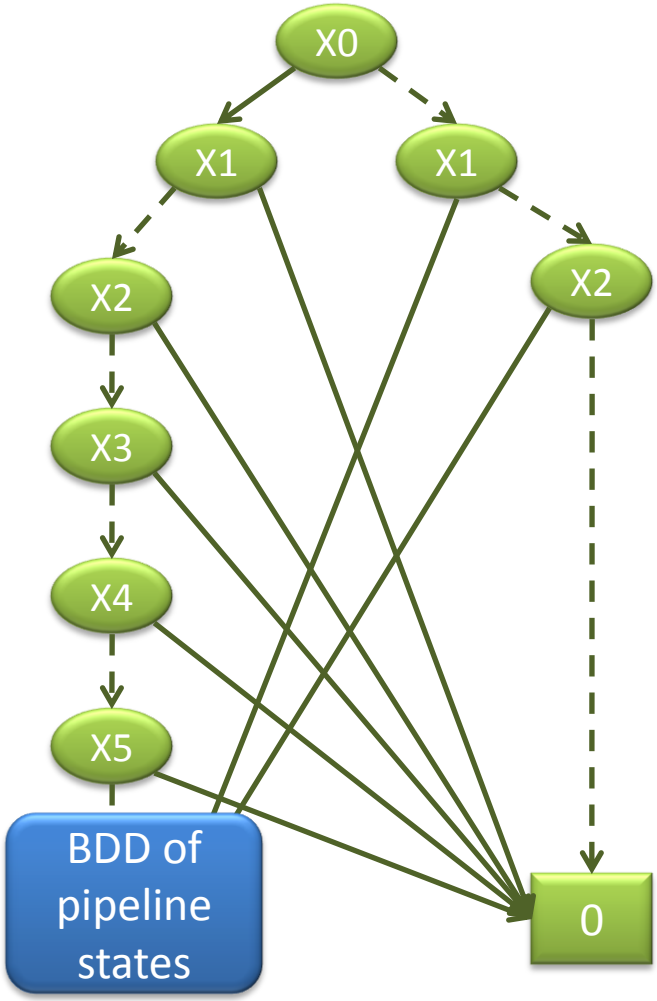


- Exchange information only between pipeline and cache states which cover the same execution history.
  - Requires a combination of pipeline and cache states.
- Explicit-state pipeline analysis uses tuples (P_k,C_k), (P_k',C_k') ...
- What to do for symbolic pipeline analysis?

Proposed Solution

- Combine **many** pipeline states with **one** cache state
  - semi-symbolic domain
  - 1 domain element = set of tuples: $(B^n \rightarrow B, \hat{C})$
- **Maintain property**
  - all pipeline states in the same tuple access the same cached memory interval
- **Requirement**
  - memory is addressed via the first variables in the BDD representation of pipeline states

Memory Access Interval

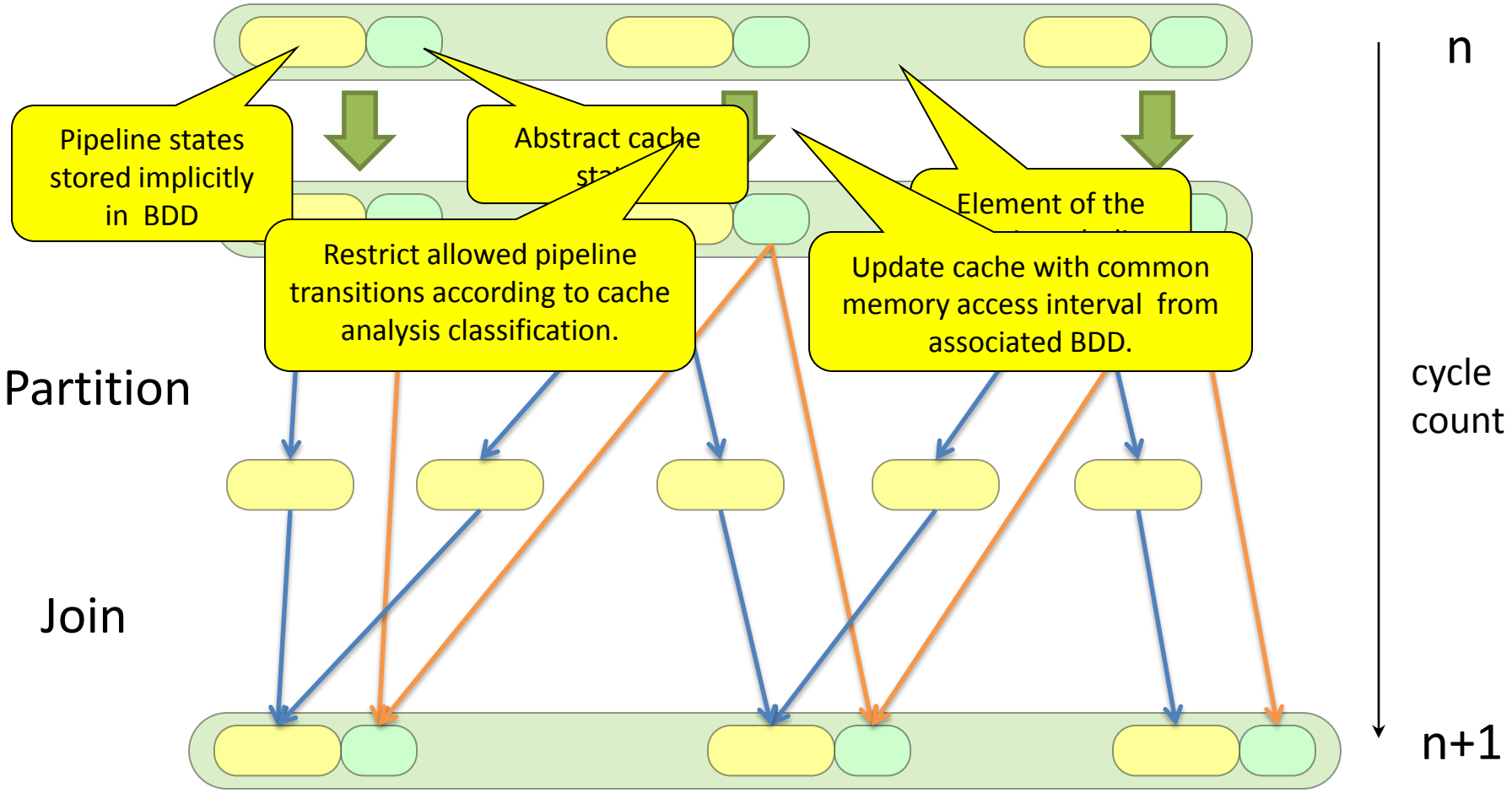


X0	X1	X2	X3	X4	X5
1	0	0	0	0	0
0	1	-	-	-	-
0	0	1	-	-	-

X0	X1	X2	X3	X4	X5	ub
1	0	0	0	0	0	32
0	1	1	1	1	1	31
0	0	1	1	1	1	15

X0	X1	X2	X3	X4	X5	lb
1	0	0	0	0	0	32
0	1	0	0	0	0	16
0	0	1	0	0	0	8

Update + Balancing



Experiments

- **Goal:** asses the efficiency of the proposed domain
  - efficient if combining **many** pipeline states with single cache state
- **Instrumented** the (explicit-state) **MPC755 pipeline** model to report for each access into cached memory:
  - accessed memory address or interval
  - type of access: instruction or data
  - cycle count since start of the current basic block
- **6 tasks** of a typical industrial software project
  - fully unrolled & annotated (avoid serious state explosion)
  - still reaches up to a few thousand states per cycle

Results

Explicit-state experiment

- cycle count since start of the current basic block
- # states with same cycle count and memory access
- # different memory accesses with same cycle count

Semi-symbolic domain

- state space exploration layer
- # pstates per BDD
  - Average : 27 / 20
  - Maximum : 8544 / 8115
- # tuples per domain element
  - Average : 2 / 1
  - Maximum : 42 / 6

Conclusion

- Covering the state space of complex hardware can become a problem in static WCET analysis.
- **Solutions** for caches and pipelines **already exist**.
  - abstract interpretation of caches, symbolic pipeline analysis
- Proposed an approach for integration.
  - **Efficient if combining many pipeline states with a single abstract cache state**.
  - Experimental data indicates that favorable combinations can be expected.
  - May lose precision due to additional joins of cache states.

Q & A