Formal Methods Group ETH Zürich

June 2003

Armin Biere, Cyrille Artho, Malek Haroud, Viktor Schuppan

Computer Systems Institute

ETH Zürich, Switzerland

FMICS'03, Røros, Norway

Vision

Formal method tools are used like **compilers:**

In the context of (formal)

1. verification.
2. synthesis.
3. analysis.

Overview

1. Model Checking, SAT, and QBF.
2. Translation of liveness into safety.
3. High-level data races.
4. Replaying of multi-threaded executions.
5. Equivalence checking of SDL vs C.

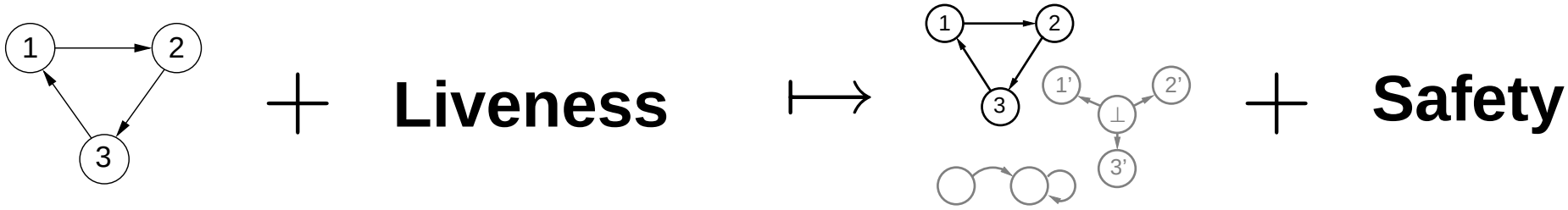
Model Checking

- BDD based mu-calculus model checker mu-cke
  - Efficient implementation.
  - Input language with C++ syntax for specifying model and properties.
- Performance study of BDD based model checking
- Bounded Model Checking
  - Leverages power of SAT solvers for model checking purposes.
  - Wide industrial acceptance.

SAT and QBF

- SAT (propositional satisfiability solvers)
  - **Continuing** increase in reasoning power.
  - Instances with million of variables can often be handled.
  - Dedicated heuristics for bounded model checking possible.
- Solvers for QBF (quantified boolean formula), e.g., $\forall x \exists y \left[(\bar{x} \vee y) \wedge (x \vee \bar{y})\right]$
  - Start to become practical ...
  - ... although more practical research necessary (efficient implementations).
  - Potentially allow to make bounded model checking complete.
- Applications of QBF and SAT in other domains (e.g., SW checking).

Translating Liveness into Safety: Finite State Systems

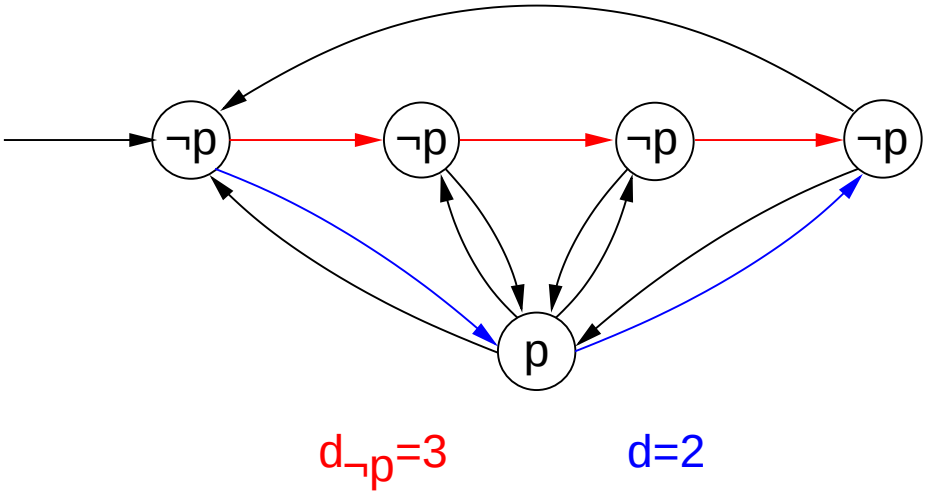


If the number of states is finite:

1. A system with a liveness property can be transformed into a system with an equivalent safety property.

2. The transformed system can be model-checked **efficiently**.

Translating Liveness into Safety: Predicated Radius/Diameter

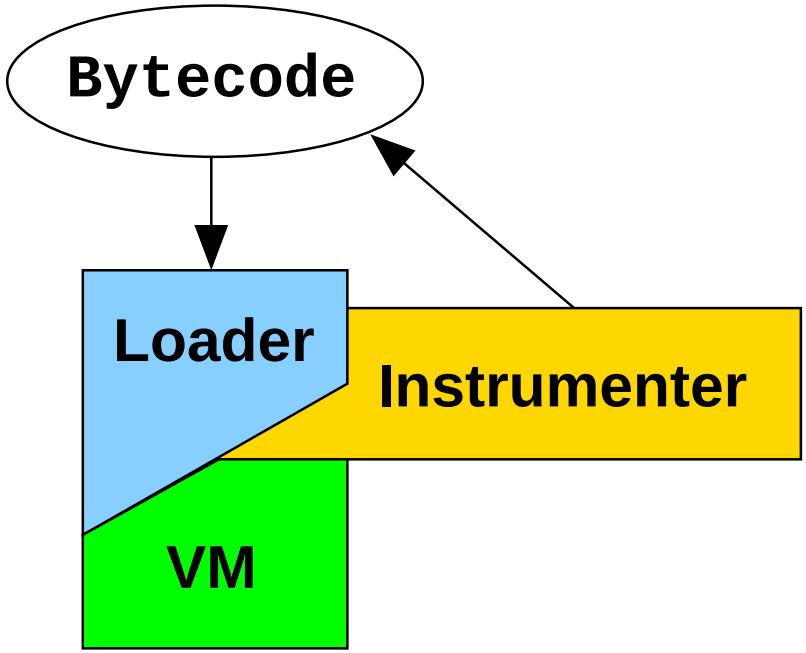


Bounds stated at FMICS'02 require further restrictions:

- Search for counterexample traverses paths where $\neg p$ holds.
- Notion of predicated radius and diameter.
- Leads to tight bound for bounded model checking of $\mathbf{F}p$.

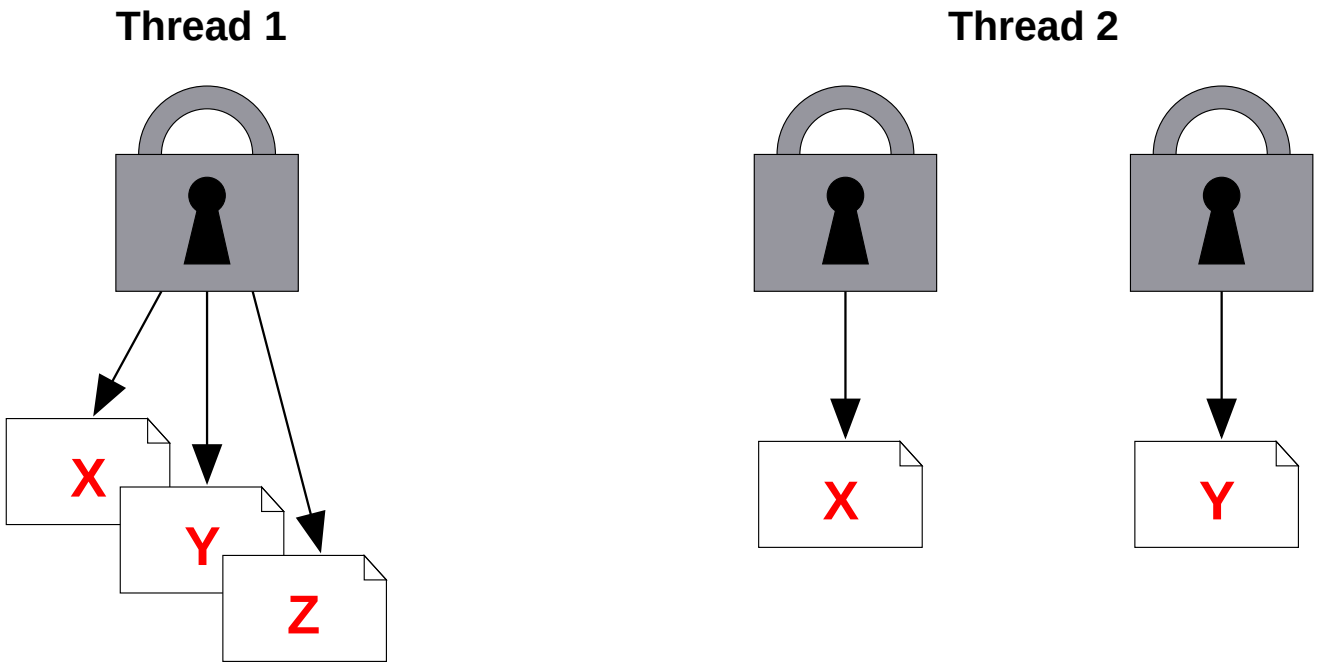
JNuke: Our own Java Virtual Machine

Platform for static and dynamic analysis



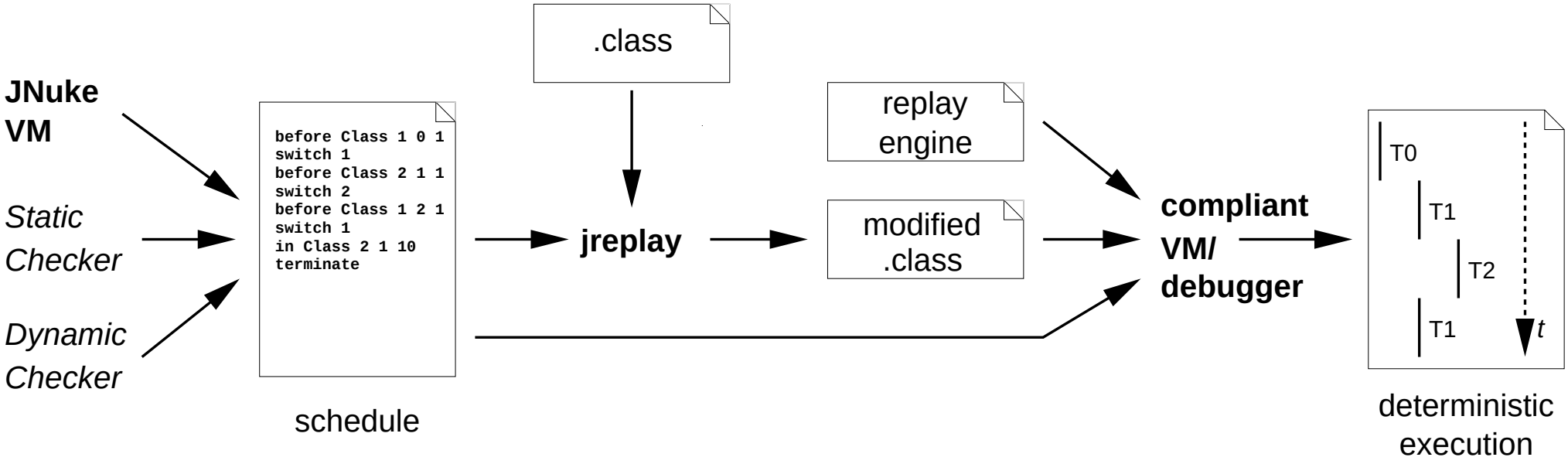
- Java VM written in C.
- API for run-time analysis.
- Small state representation.
- Rollback (undo) operations.
- “Exhaustive” scheduling possible (Rivet).
- Instrumentation: reproducing counterexamples.

JNuke: High-level data race analysis



- Both accesses are protected by a common lock (Eraser).
- Different atomicity assumptions by the two threads.
- New source of potential errors, found by **view consistency**.

JNuke: Replay of Multi-Threaded Executions

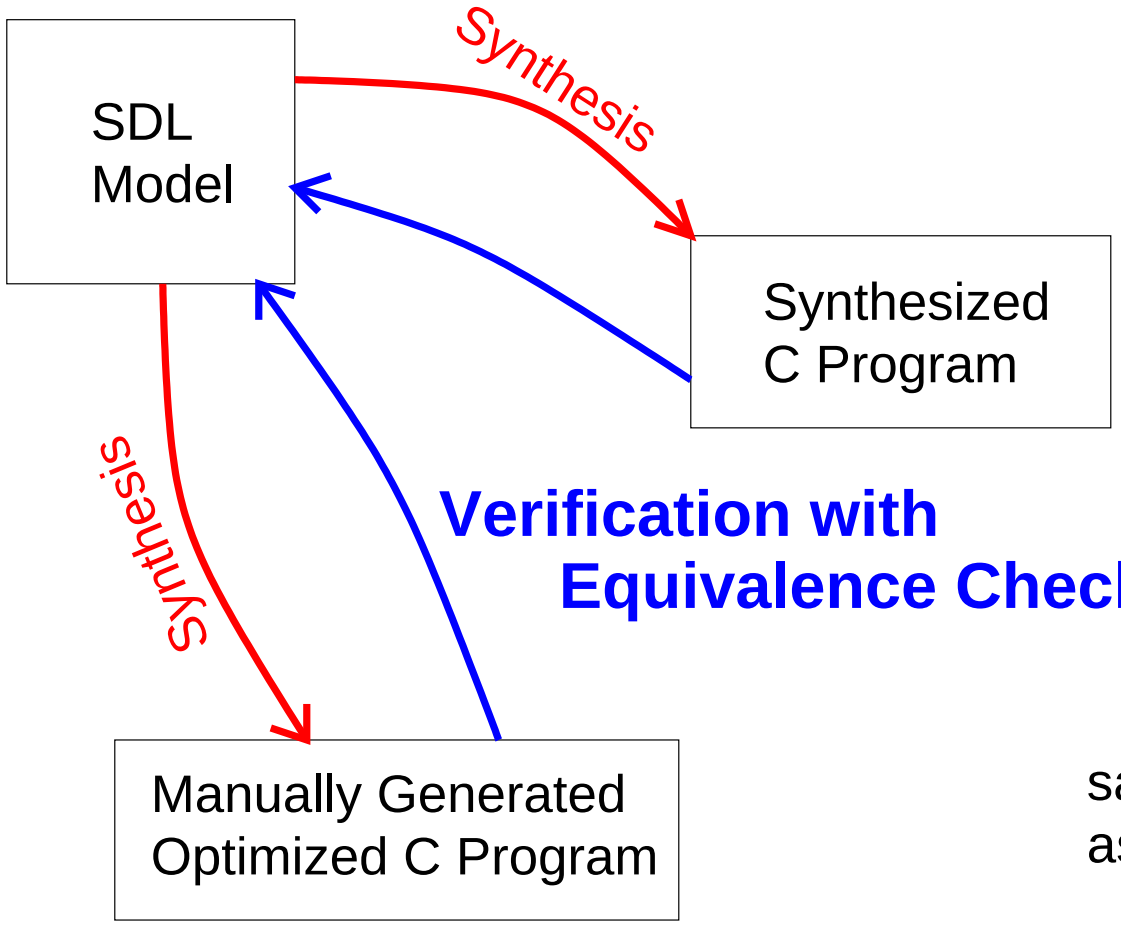


- Enables replay of thread schedules independently of specific VM.
  → Off-the-shelf debuggers.

- Schedule format not tailored to JNuke VM.
  → Usable by other tools.

Equivalence Checking SDL vs C

SDL as modelling language in telecommunication applications
(or more general for embedded SW)



same motivation
as in HW equivalence checking

Establishing formal methods

Short-term:

Scalability, light-weight process.

Long-term:

Formal loop: Formal methods on all levels.