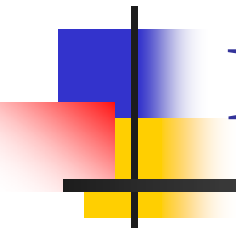
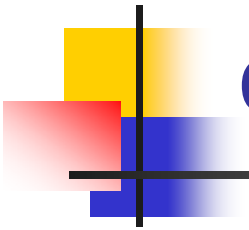


Formal Verification and Conformance Testing for Reactive Systems

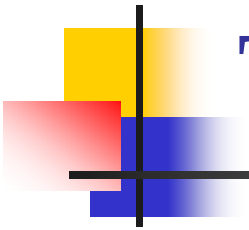
Vlad Rusu, Irisa/Inria Rennes
Projet Vertecs

December 8, 2006

Formal verification : proving correctness



- Verification by “paper/pencil”
- Algorithmic techniques
  - Model checking
  - Abstract interpretation
- Deductive techniques
  - Interactive theorem proving
- Various combinations of the above.

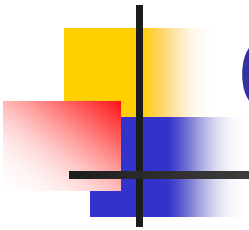


Testing: finding errors

- What is tested:
  - *White box:* Source code
  - *Black box:* Executable code
- Type of testing:
  - Functional (against specification/oracle)
  - Structural (against coverage criteria)
  - Robustness, performance, real time…

Combining verification and testing: best of both worlds?

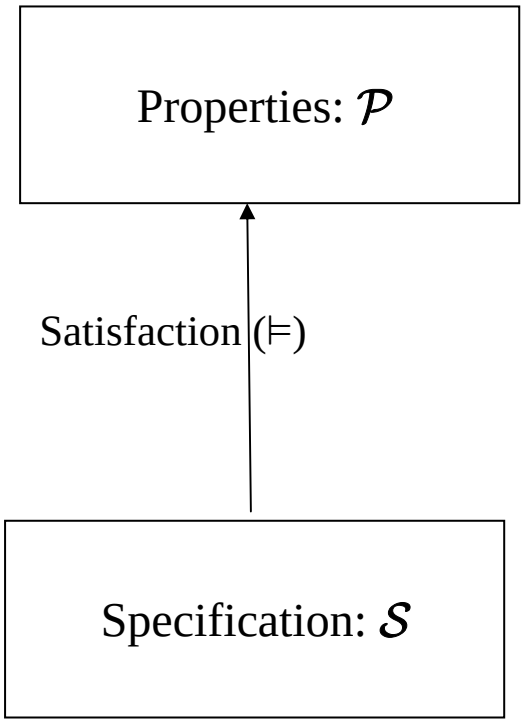
- Testing using verification techniques
  - White box
    - Using a model checker to derive structural tests [Ammann][Heitmeyer]
    - Definition of coverage using temporal logic/observers [Lee][Jonsson]
    - Abstraction for structural testing: “predicate coverage” [Henzinger] “abstract path coverage” [Ball]
  - Black box
    - Test generation for conformance using model checking techniques [Jéron] [Tretmans]
    - Test generation for conformance using symbolic simulation [Le Gall]
    - Test generation for properties using model checking [Fernandez]
- Combining verification and testing
  - The ESC/Java toolset.



Outline

- A closer look at verification vs. conformance testing
- Contributions to verification (esp. *theorem proving*)
- Contributions to conformance testing (esp. *symbolic test generation*)
- Integrating verification and conformance testing
- Conclusion and perspectives.

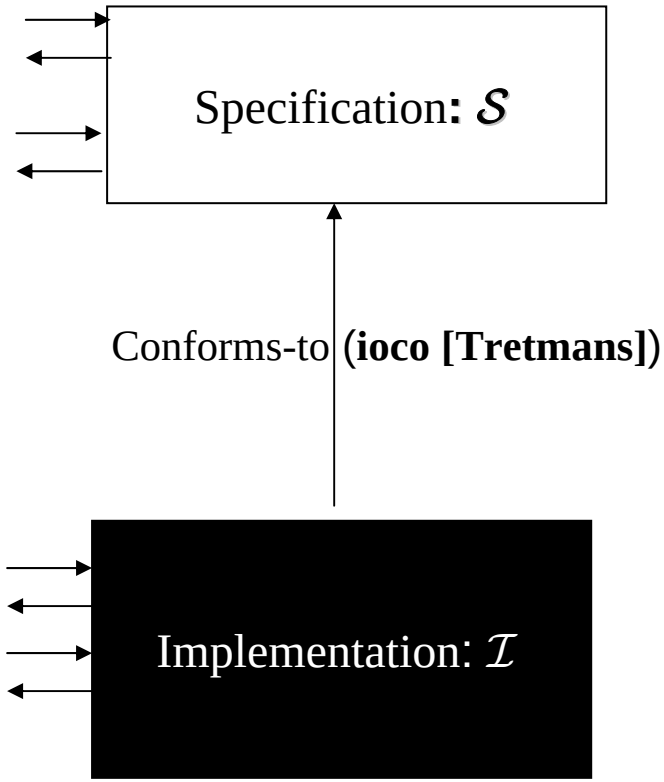
Formal Verification



Verification Problem: $\mathcal{S} \sqsubseteq \mathcal{P}$

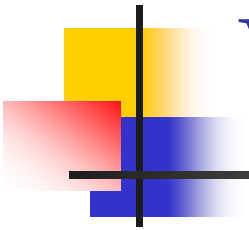
- Can be reformulated as $\mathcal{S} \times \overline{\mathcal{P}} \not\mapsto$ 
- Basic operations involved:
  - Product $\times$
  - Complementation (determinisation)
  - **Reachability.**

Conformance Testing



Conformance Testing Problem: $\mathcal{I}$ **ioco** $\mathcal{S}$

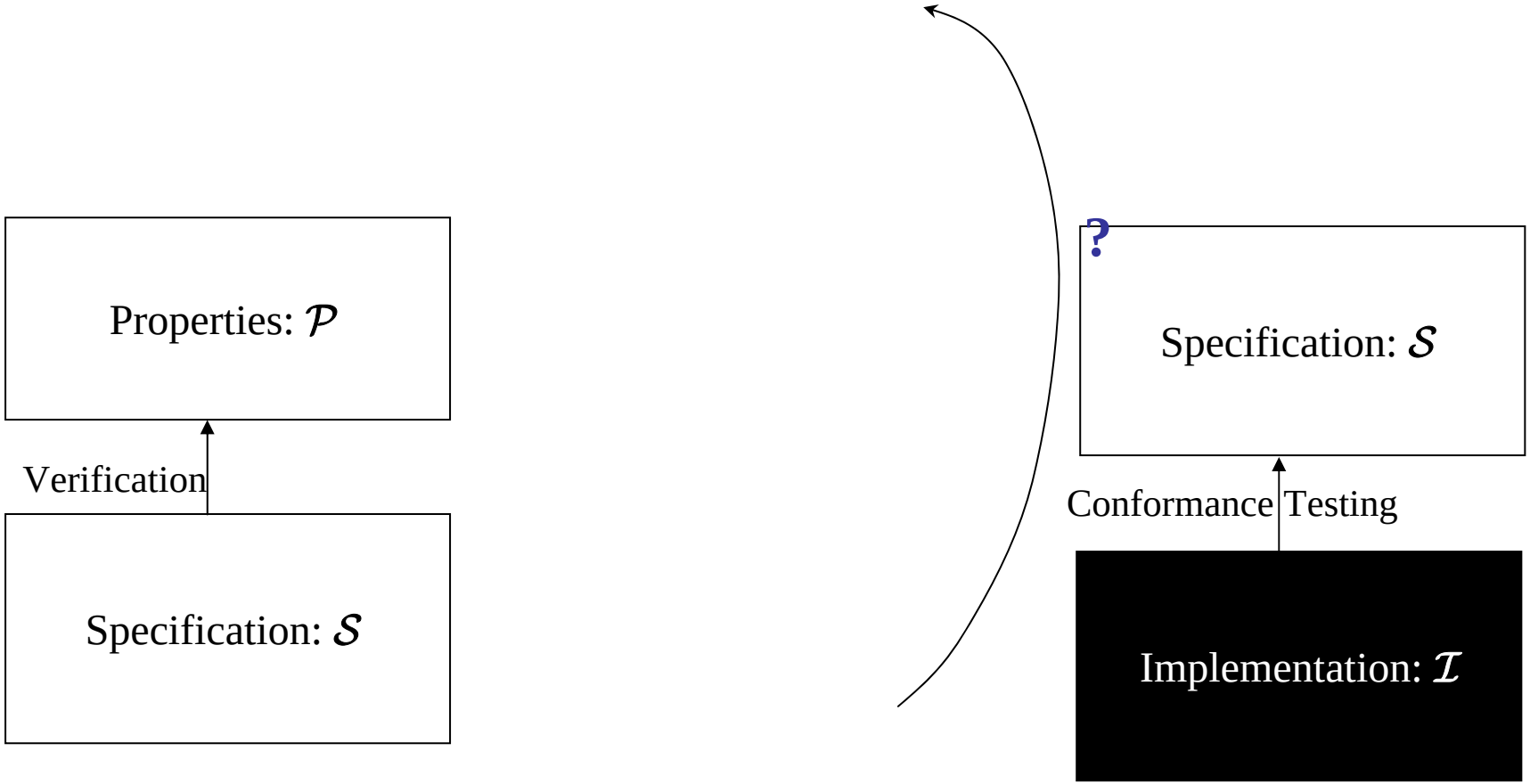
- Reformulated as $\mathcal{I} \| test(\mathcal{S}) \nrightarrow$ 
- Basic operations required :
  - Parallel composition $\|$ ($\equiv$ product $\times$)
  - Complementation (determinisation)
  - **Reachability.**



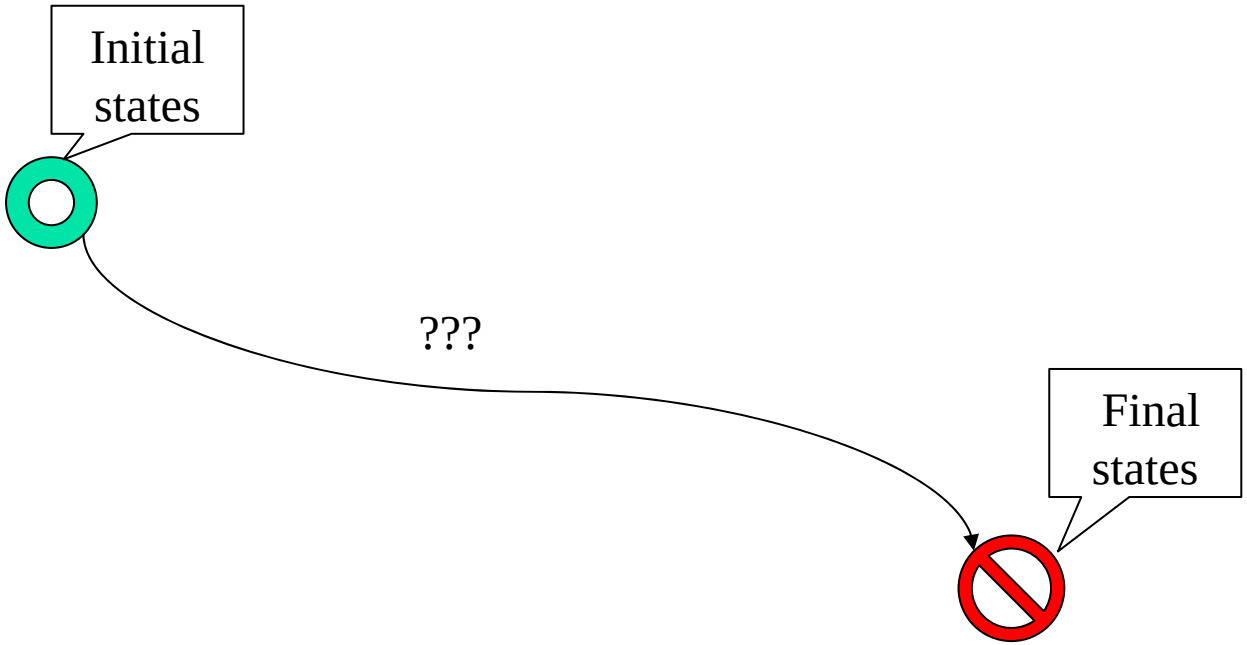
Verification vs. ConformanceTesting

- Same basic operations involved
- Verification: all formal models & reasoning
  - Can prove or disprove satisfaction relation
- Conformance testing: model of $\mathcal{I}$ unknown
  - Can only disprove conformance relation.

Verification *and* Conformance Testing

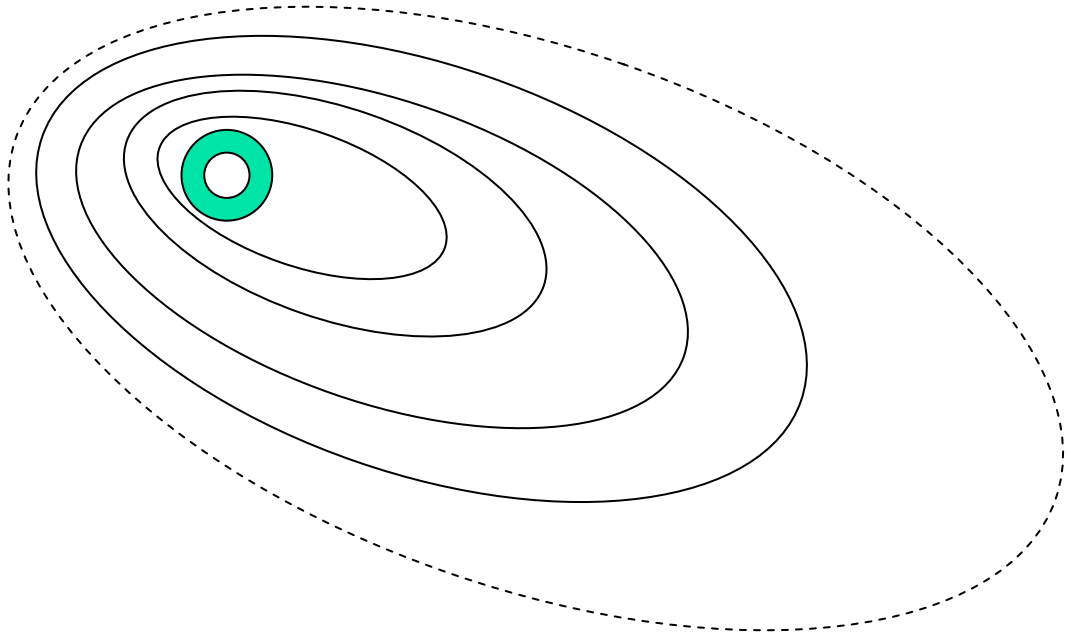


Verification: Reachability

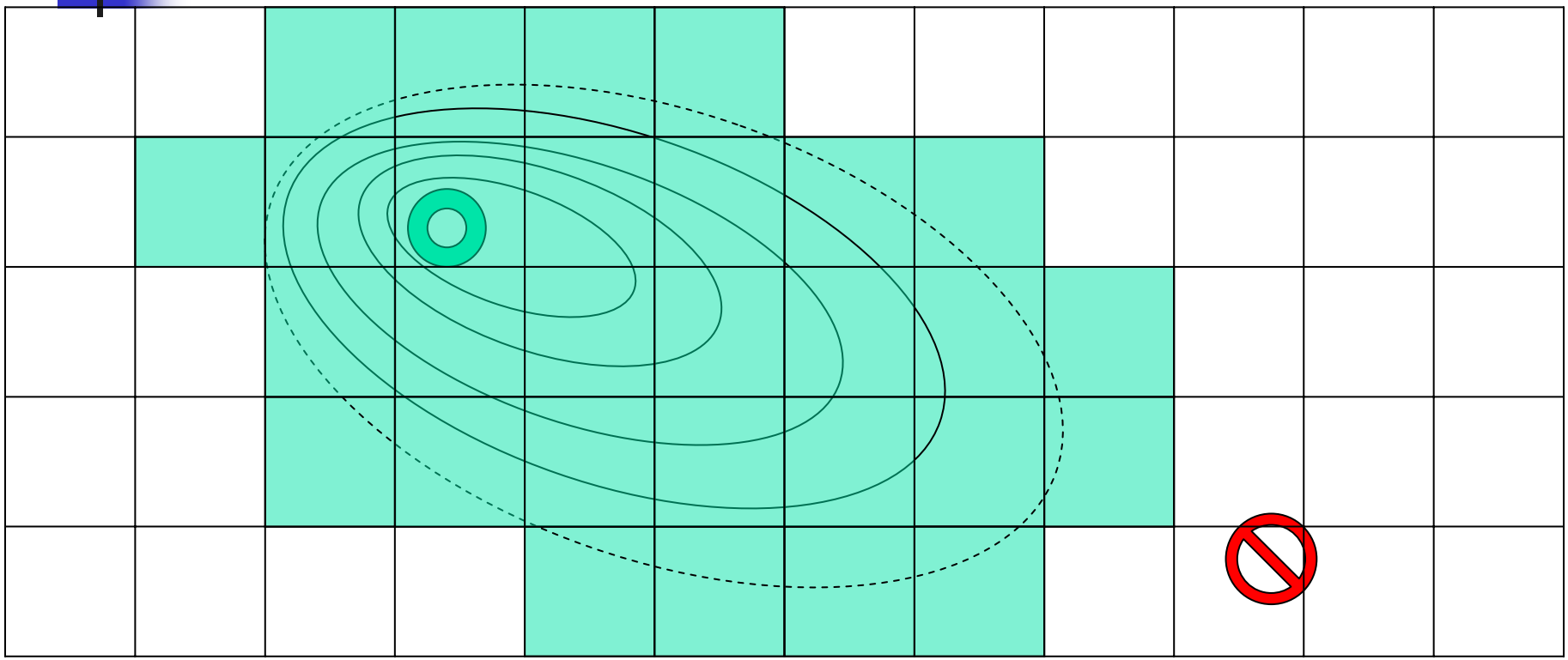


Computing sets of reachable states

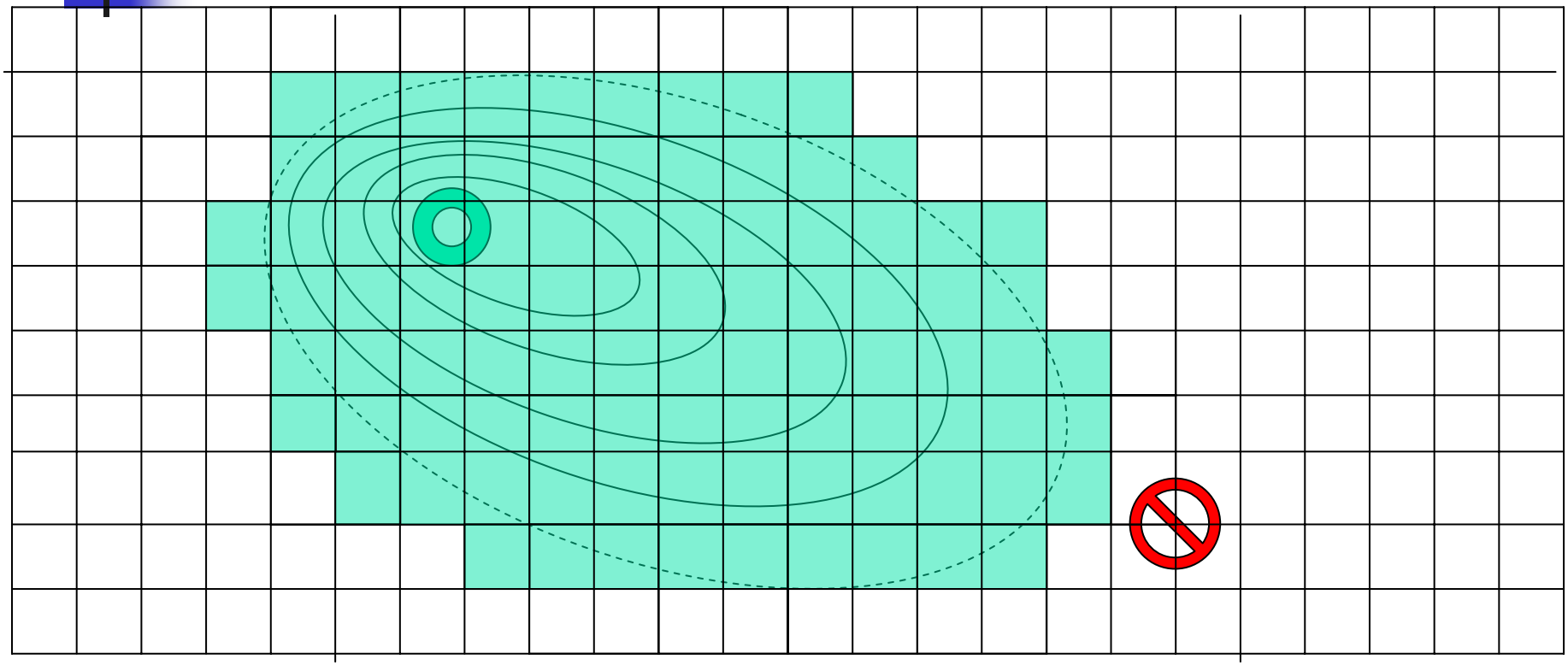
For classes of hybrid automata
with O. Roux,
F. Cassez, A. Burgueno
[SAS96]
[Hart97]
[EurPar98]
[Formal Aspects of Computing99]
with T. Henzinger
[HSCC98]

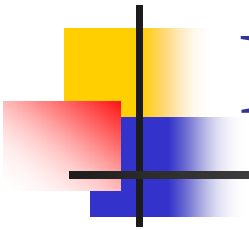


If exploration does not terminate..



Refine approximation





Predicate abstraction/refinement

- Success story in formal verification
  - SLAM (Microsoft)
- Still an active research domain
- In [Tacas99] with E. Singerman
  - Refining abstractions using information obtained from attempted (non-reachability) *proof*
  - Equivalence of abstraction and invariant strengthening.

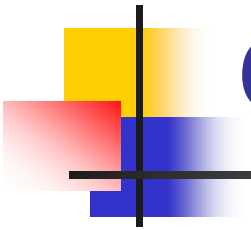
Verification by Theorem Proving: Invariant Strengthening

- Goal: find predicate Θ
  - *Invariant* (closed) under $\rightarrow$
  - Includes 
  - Does not intersect 
- Start with $\Theta =$ 
  - Failed invariance proofs: *auxiliary predicates* $\mathcal{A}$
  - Continue with $\Theta := \mathcal{A}$ until proof (or… too tired).

Helping invariant strengthening

- For properties/specs in expressive formalism (Presburger arithmetic + function symbols):
  - Checking invariance: *true* or *don't know*
    - All but one auxiliary invariants in Electronic Purse case study [VCL'02], with E. Zinovieva, D. Clarke
  - Decidable for a subclass
    - All auxiliary invariants for Sliding-Window Protocol [Forte'01, TPHOLS'01]
- Also, bounded symbolic state-space exploration [ENTCS'01] with E. Zinovieva.

Case study: the SSCOP protocol

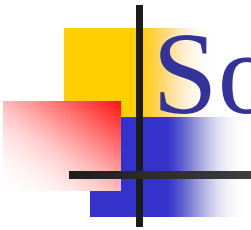


- ATM protocol from adaptation layer
- Infinite-state variables (counters, FIFOs, arrays, FIFOs of arrays of arrays…)
- With theorem proving (PVS), partial-order reduction & compositionality : 3 months

[FME03, Computer Journal06]

Other works in theorem proving

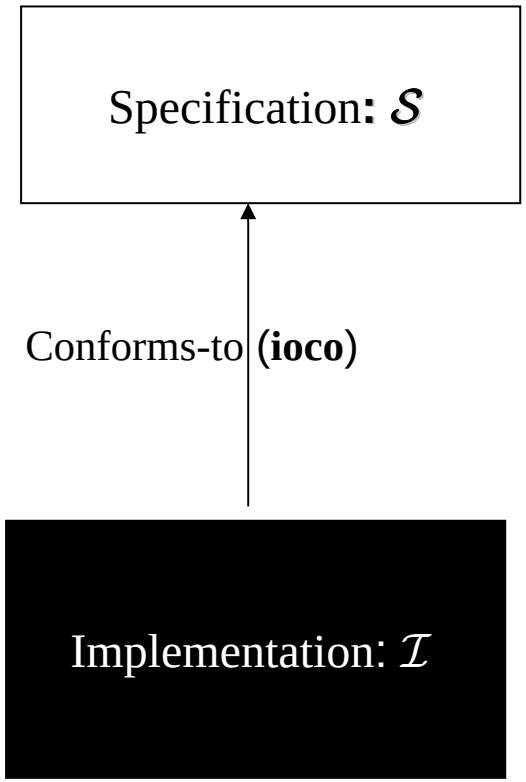
- Verifying the symbolic slice of a specification relevant to given property; case study: Electronic purse [FME02, Soft. Test. Verif. Jnal'04]
- Extracting a static analyser for Java byte code in constructive logic - Coq [Esop04,TCS'05] with D. Cachera, T. Jensen, D. Pichardie
- A method for defining & reasoning about general recursive functions in Coq [Flops'06] with D. Pichardie, G. Barthe, J. Forest
  - ⊂ Coq8.1



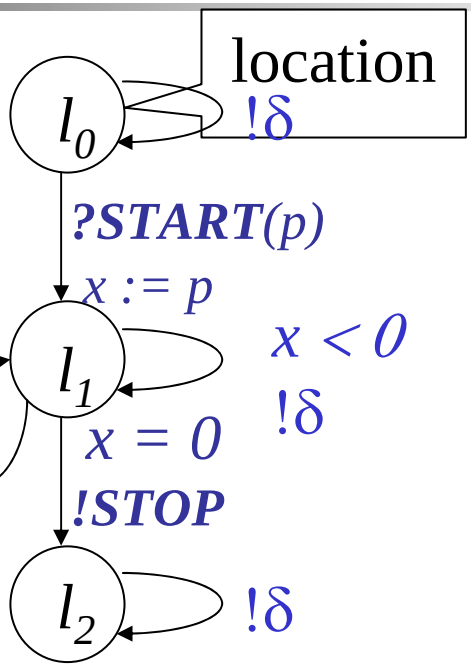
Some Conclusions on Verification

- Automatic methods, however advanced and optimised, have inherent limits
- Theorem proving alone is too tedious
- Integrating several automated techniques in theorem proving seems the best choice.

Conformance Testing



Example: counting downwards



Transition:

Guard

Action(parameters)

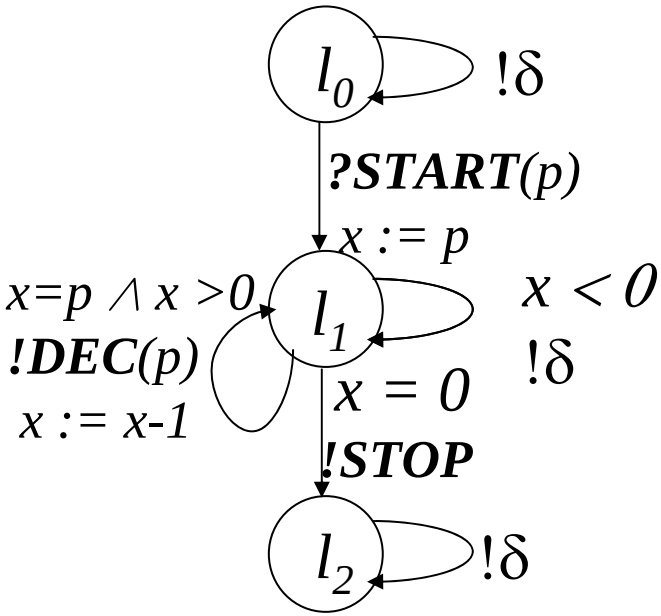
Assignments

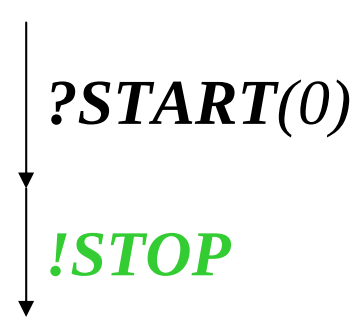
Making blockings explicit: $!\delta$

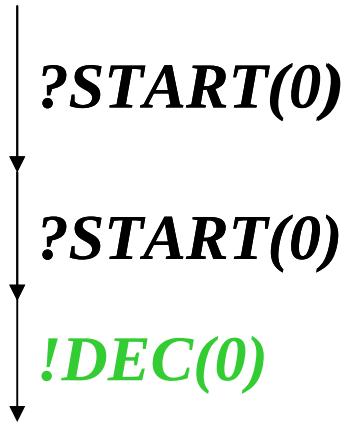
Traces:

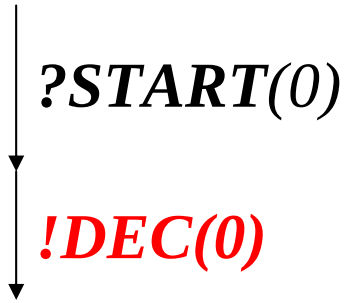
- $!\delta^*?START(p)\,!\delta^*, p < 0$
- $!\delta^*?START(0)!STOP\,!\delta^*$
- $!\delta^*?START(p)!DEC(p)!DEC(p-1)\ldots!DEC(1)!STOP!\delta^*(p > 0)$

$\mathcal{I}$ ioco $\mathcal{S}$: after all traces of $\delta(\mathcal{S})$, outputs of $\delta(\mathcal{I}) \subseteq$ outputs of $\delta(\mathcal{S})$

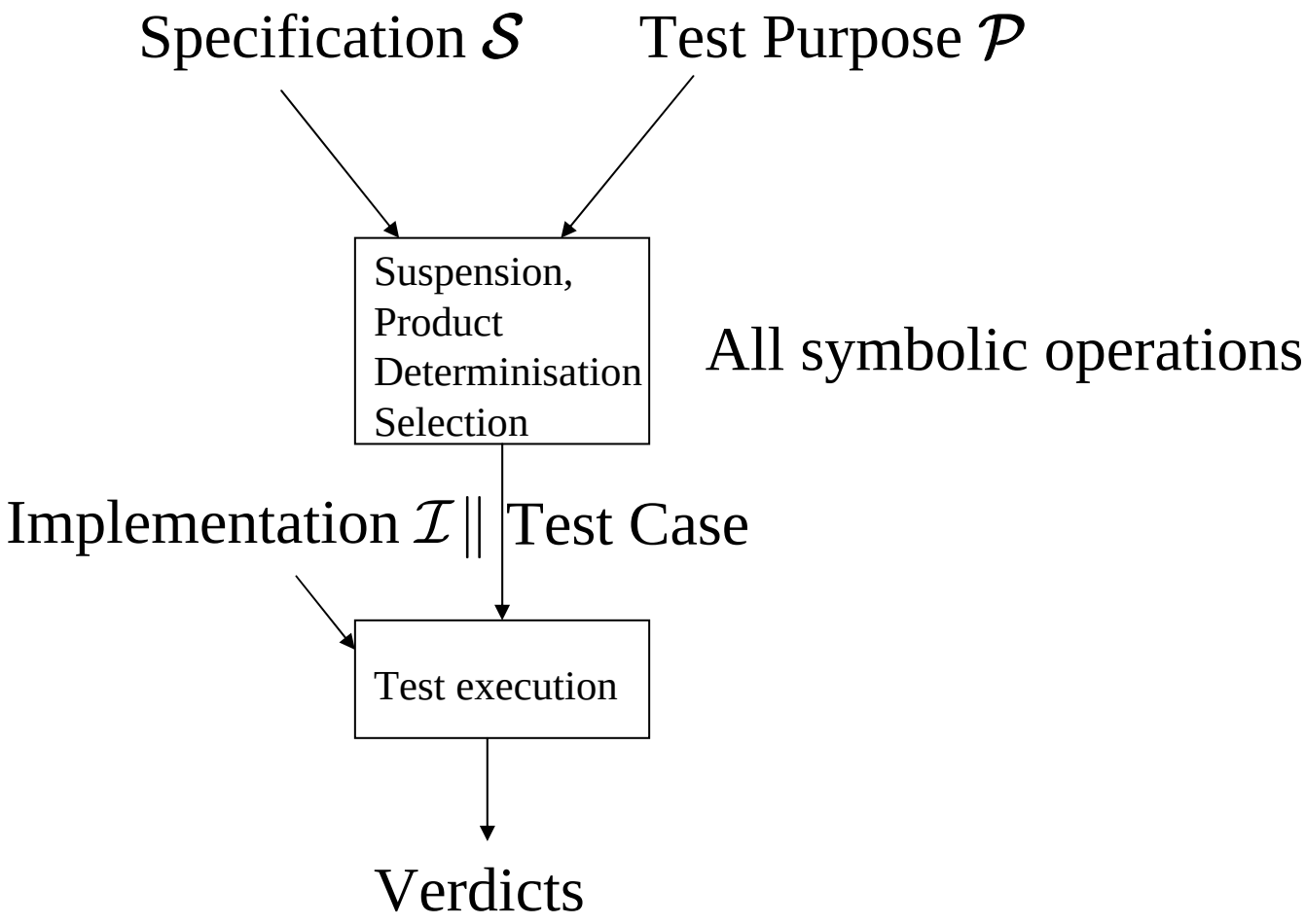




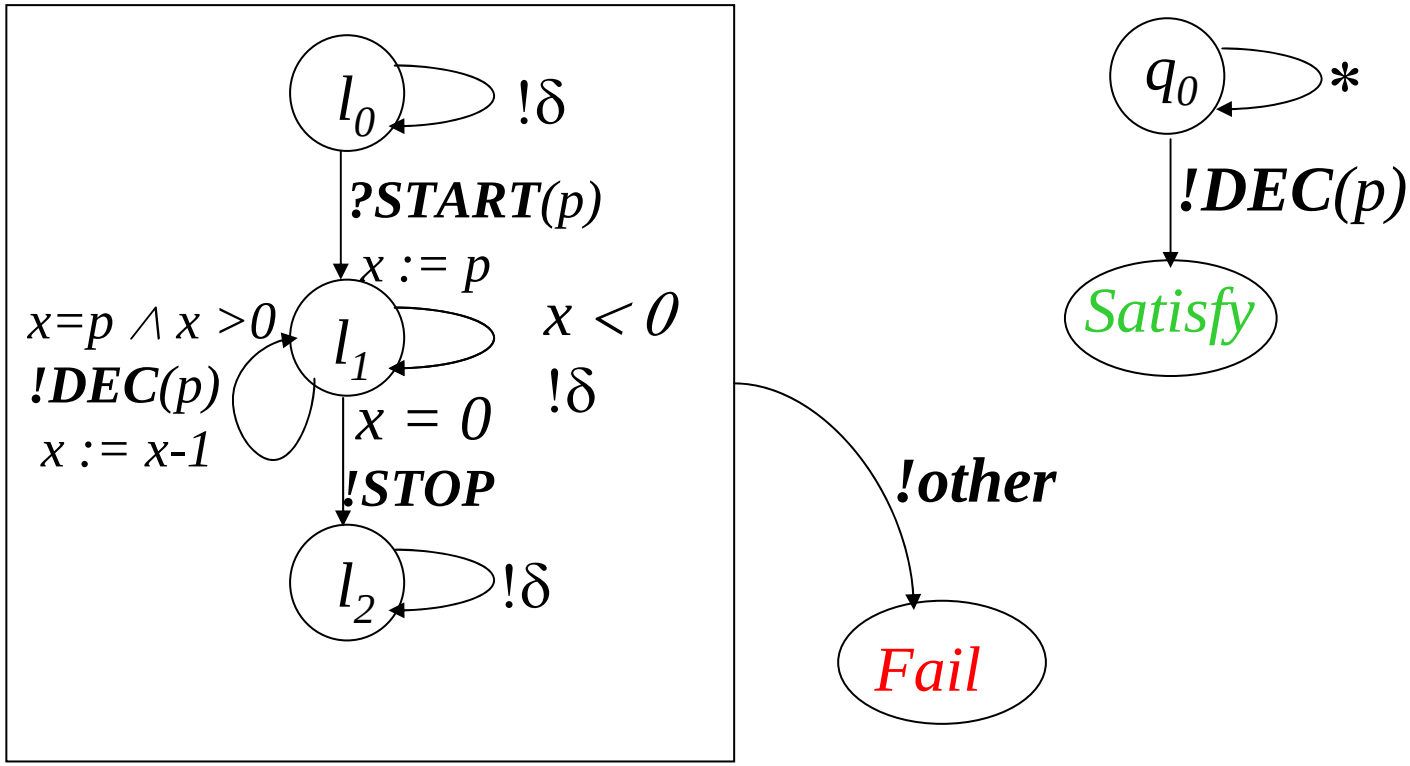




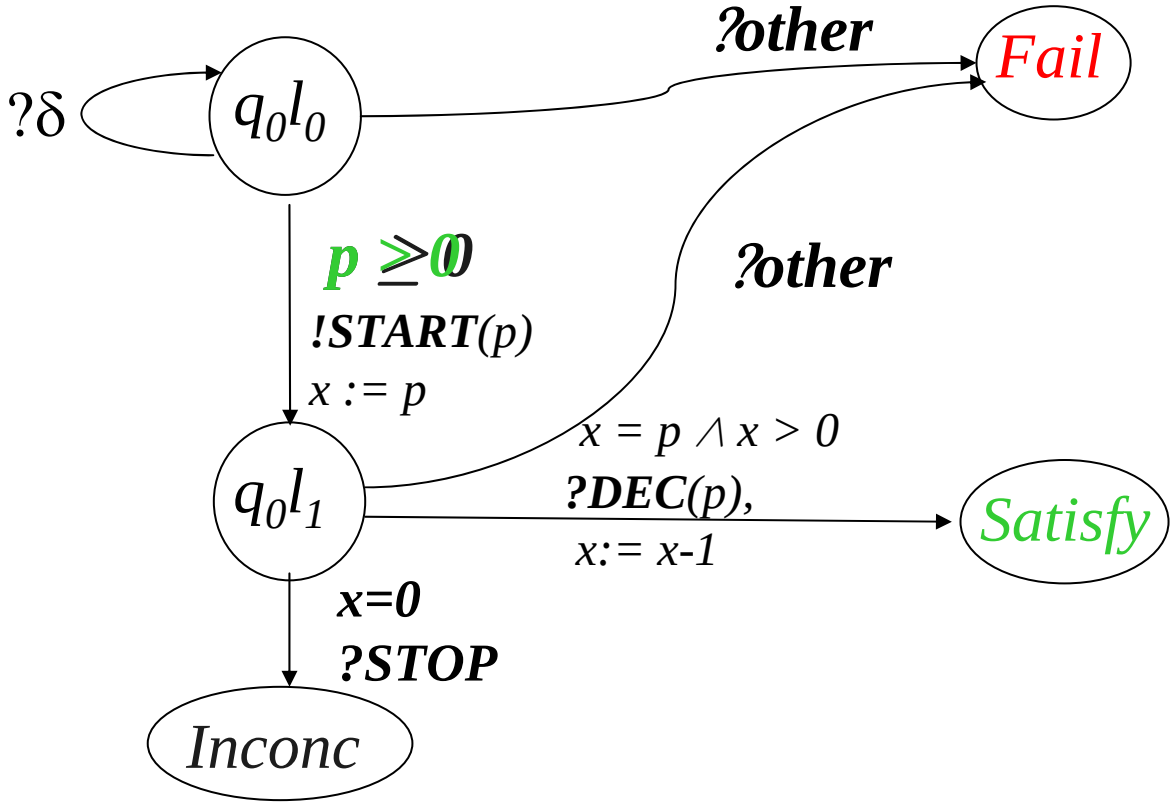
Symbolic Test Generation



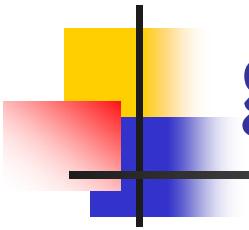
Example: specification, test purpose



Resulting Symbolic Test Case

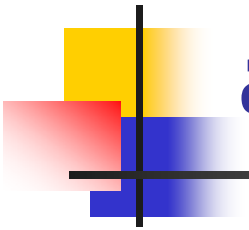


Contributions to symbolic test generation



- Papers:
  - [IFM'00, esmart'01, Tacas'02,Tacas'05] with L. du Bousquet, D. Clarke, T. Jéron, B. Jeannet, E. Zinovieva [PhD, 2004]
- STG tool by F. Ployette with contribs from D. Clarke, F.-X. Ponscarme, E. Zinovieva
- Main case study: Electronic purse

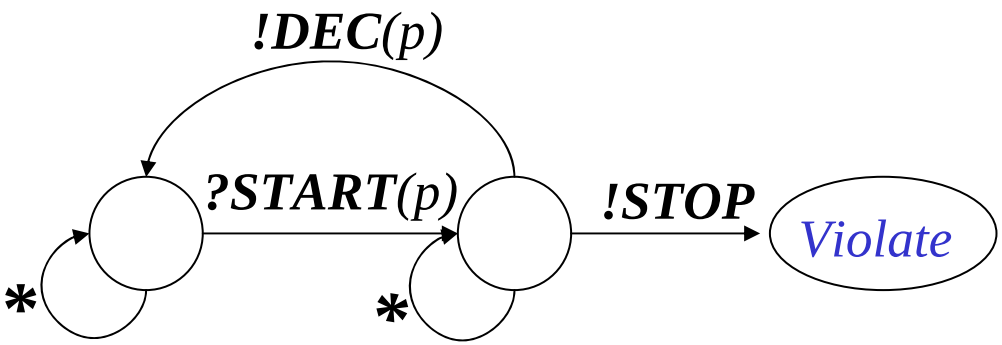
Towards integrating verification and conformance testing



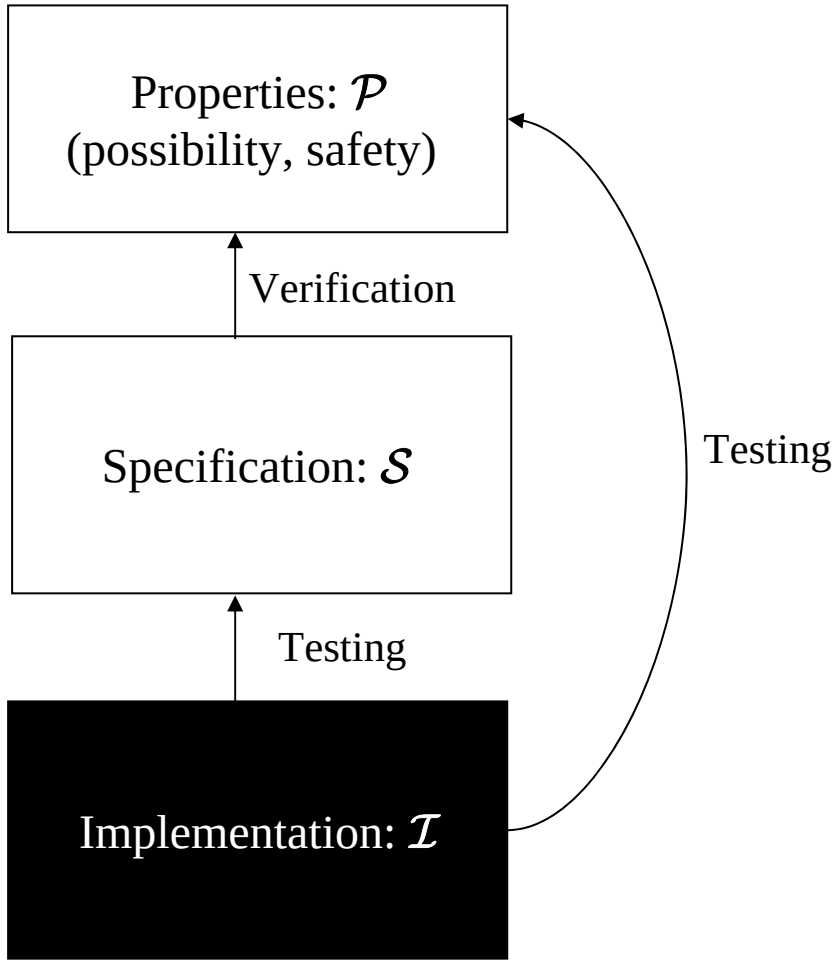
- "Test purpose" $\equiv$ a *possibility* property of the specification: certain traces are *possible*
- More (most?) interesting properties: *safety*
- Different interpretation of *final locations*
- Observers: standard approach in verification.

Example: observer for a safety property

No ***!STOP*** between ***?START*** and ***!DEC***



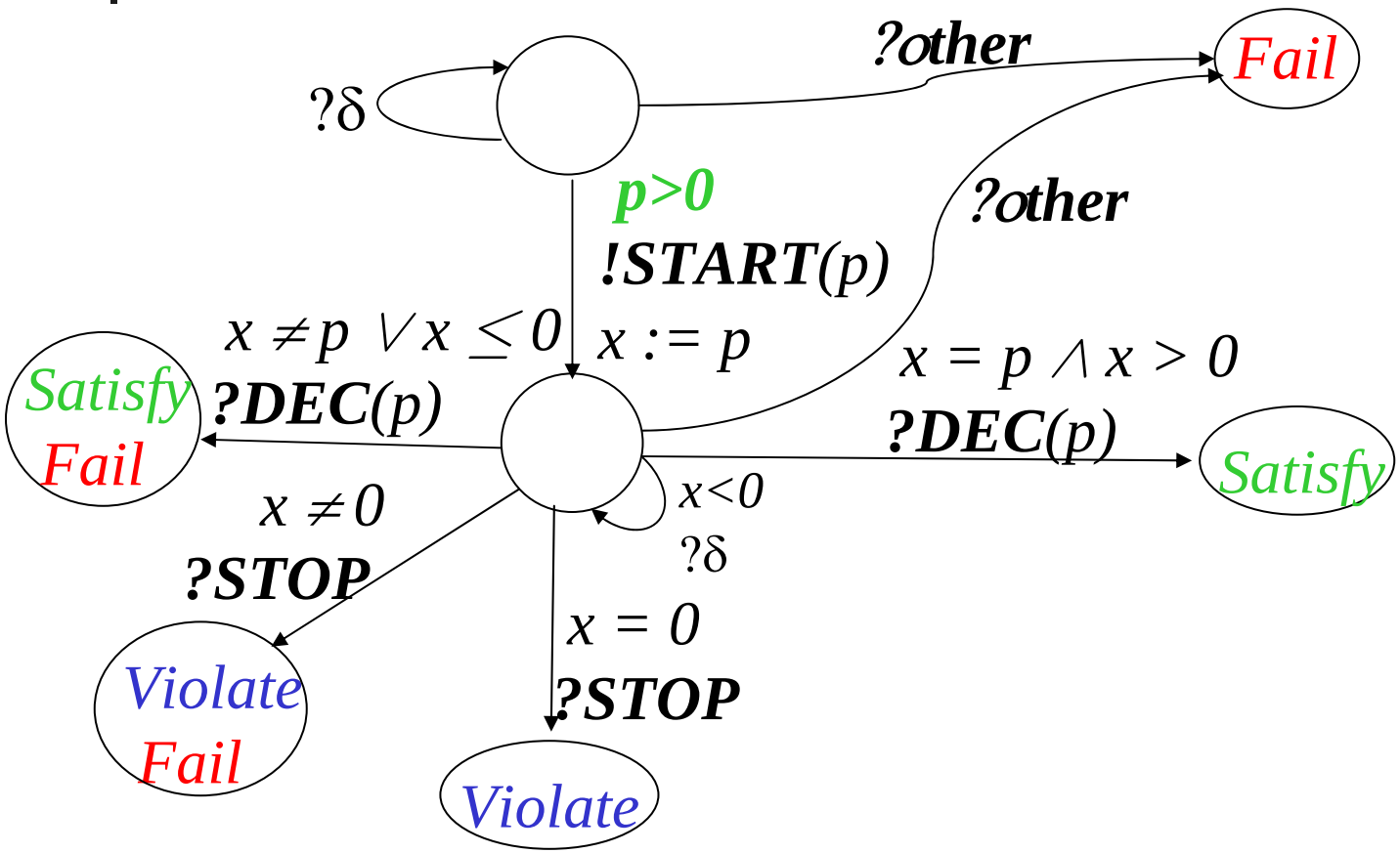
Verification *and* Conformance Testing



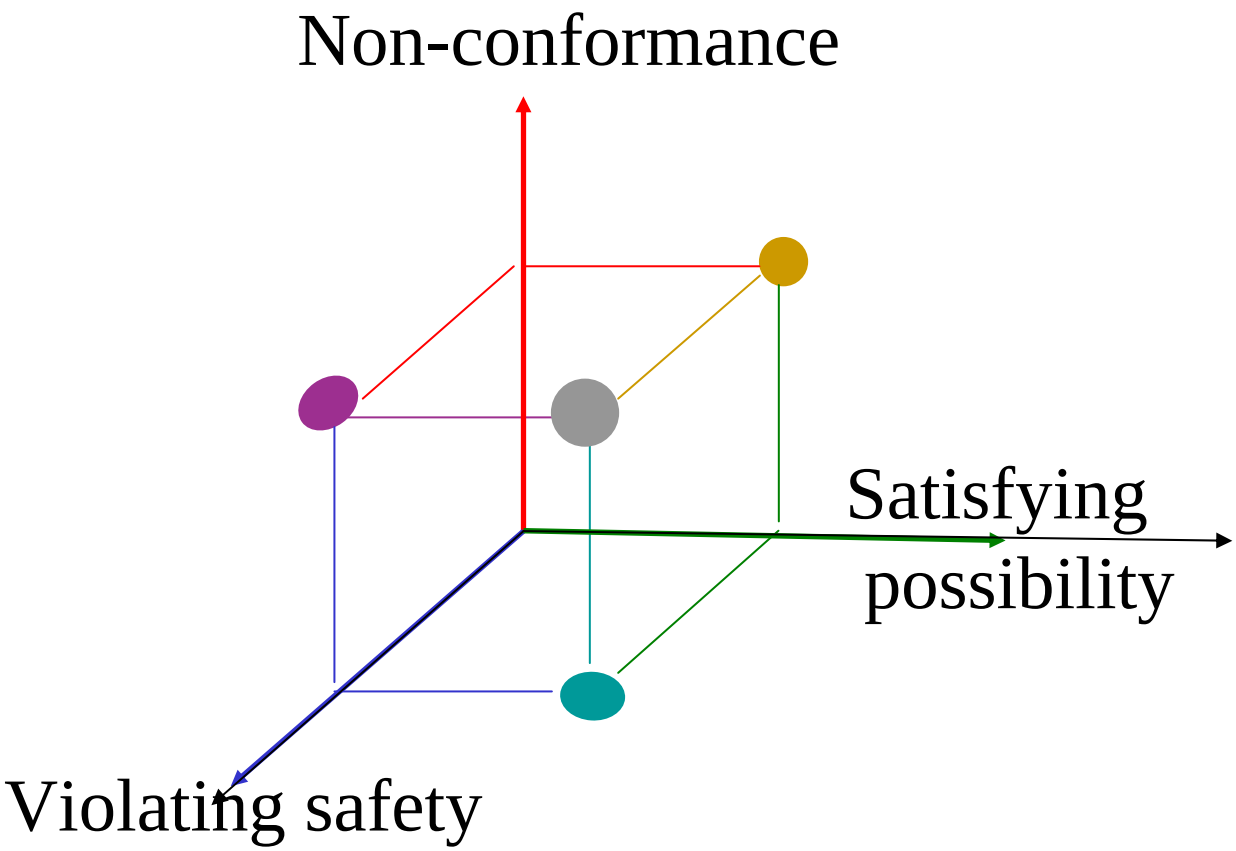
Methodology

- Verify $\mathcal{S}$ against (observers for) properties $\mathcal{P}$
  - Build their product $\times$, check reachability of final location(s)
  - Under-approximation (e.g. model checking) to prove reachability
  - Over-approximation (e.g. abstract interpretation) to disprove it
- Whether verification conclusive or not$\rightarrow$ test generation
  - Transform $\mathcal{S}$ into observer for nonconformance: “canonical tester”
    - Suspension, Determinisation, Output-completion
  - Product with observers for properties $\mathcal{P}$**:** lots of verdicts!
- Test selection: choose among verdicts, compute co-reachability (abstraction interpretation again)
- Test execution: may complete verification.

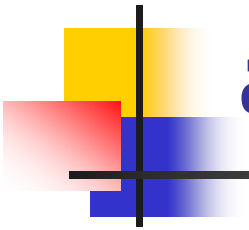
Test generation: product, selection



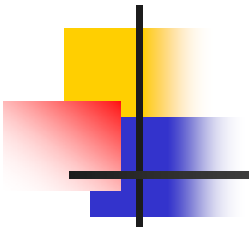
Interpretation of verdicts



Summary: integrating verification and conformance testing…

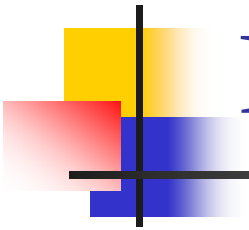


- Establishes relative consistency between implementation, specification, properties
- Testing step does not depend on success of verification step
  - Can even be done all at the same time
- [FM'05,BookChapter] with C. Constant, T. Jéron, H. Marchand.



Some General Conclusions

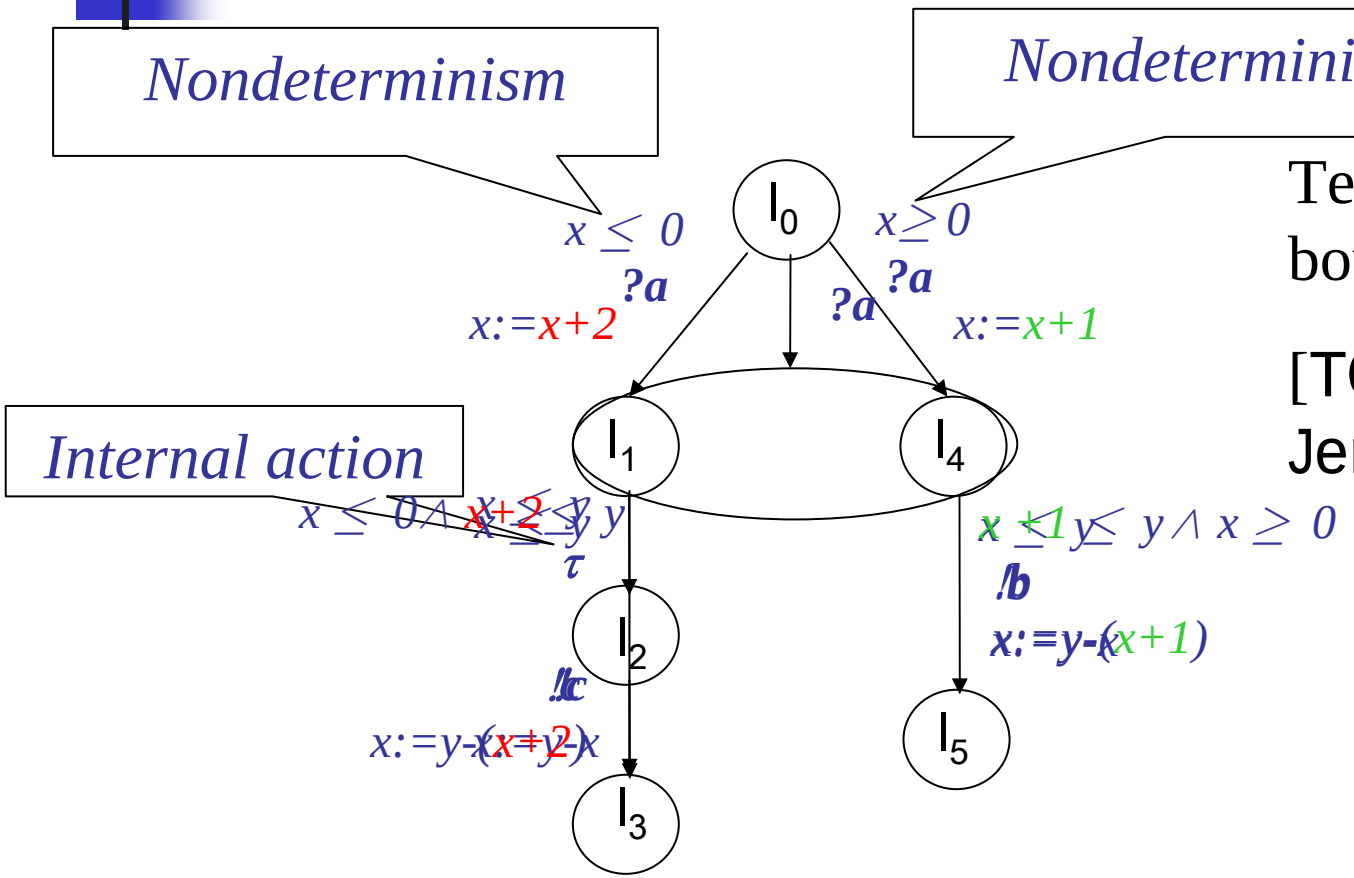
- Verification and testing are complementary
  - Operations, methodology
- Integration of methods is still the future
  - Also with control synthesis, fault diagnosis…
- Main issues to wider application
  - Complexity/limits of tools
  - Lack/incompleteness of formal specifications
  - But promising start in certain areas/industries.



Perspectives

- In conformance testing:
  - Coverage
  - More expressive models (time, recursion, …)
  - Compositionality
  - Testing and games
  - Target application: security
- In verification:
  - Build links with semi-formal methods, notations
  - “Invisible formal methods” [Rushby]
- Even more integration
  - Example: abstraction/refinement in rewriting
  - To deal with incomplete/missing specifications: *learning*.

Symbolic Determinisation



Terminates iff bounded looakead

[TCS'06] with T. Jeron, H. Marchand