Introduction to Model Checking

Shiva Nejati
Simula Research Lab

March 21, 2011

Temporal Logic Model Checking

➜ Model checking is an automatic verification technique for finite state concurrent systems

➜ Developed independently by Clarke and Emerson and by Queille and Sifakis in early 1980's

➜ Specifications are written in propositional temporal logic (Pnueli 77)

➜ Verification is an intelligent exhaustive search of the state space of the design

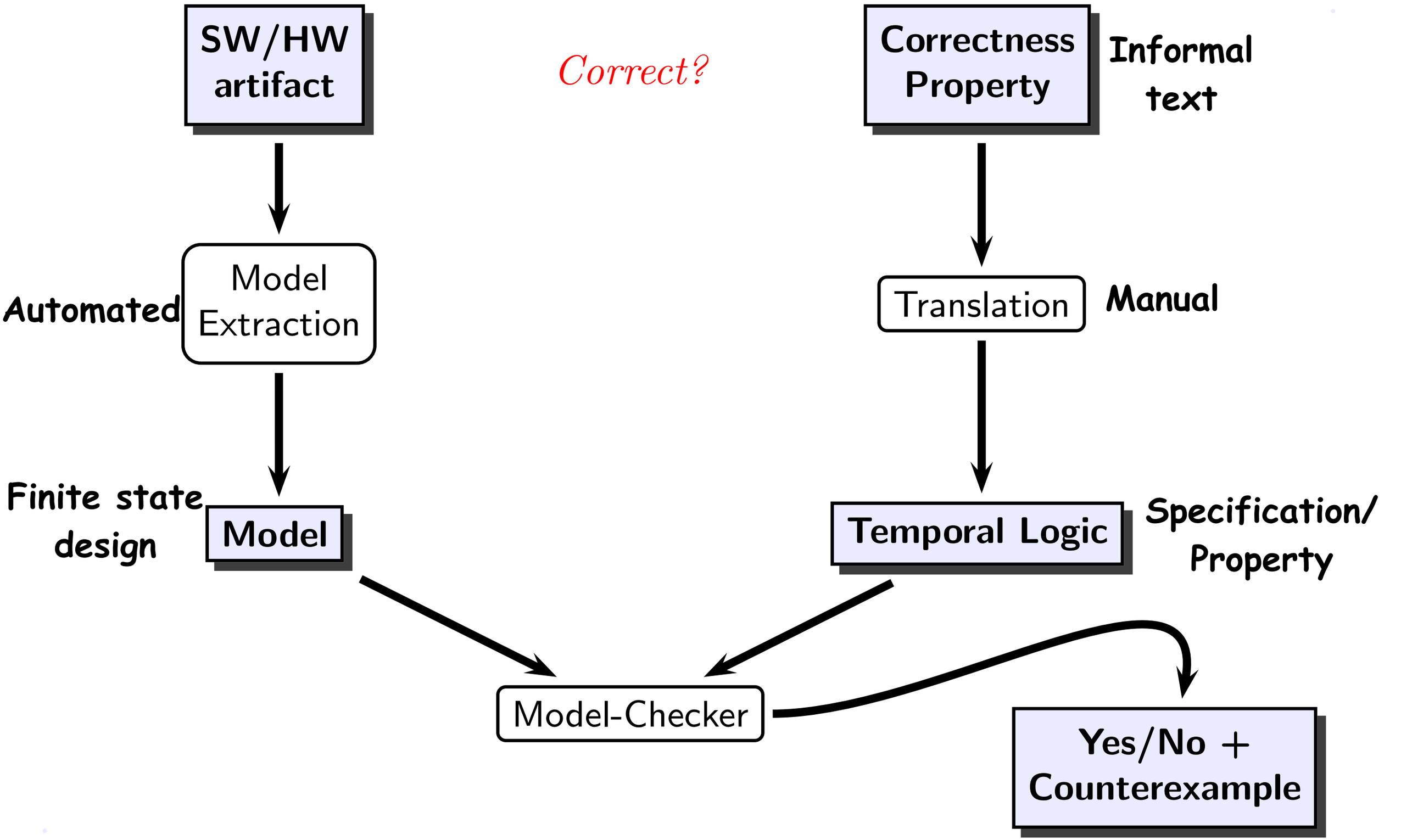
Advantages of Model Checking

- ➜ No proofs!
  - ➥ algorithmic rather than deductive
- ➜ Fast
  - ➥ Compared to other rigorous methods such as theorem proving
- ➜ Diagnostic counterexamples
- ➜ No problem with partial specification
- ➜ Logics can easily express many concurrency properties

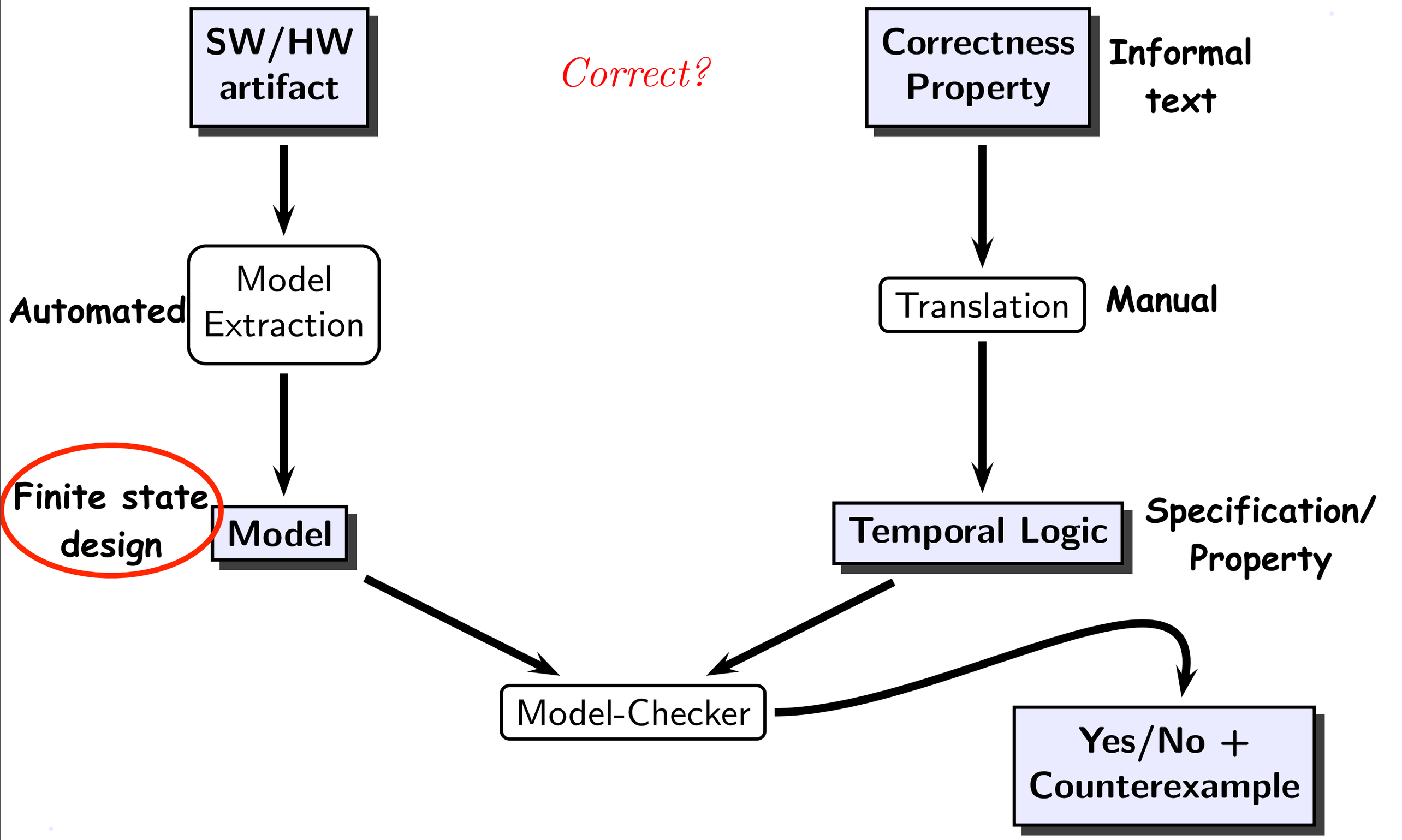
Model Checking vs Testing

- Testing
  - Checks only some of the system executions
  - May miss errors, but scales better to larger systems
- Model Checking
  - Exhaustively explores all executions in a systematic way
  - Reports diagnostic counterexamples
  - does not scale to large systems
    - state explosion problem

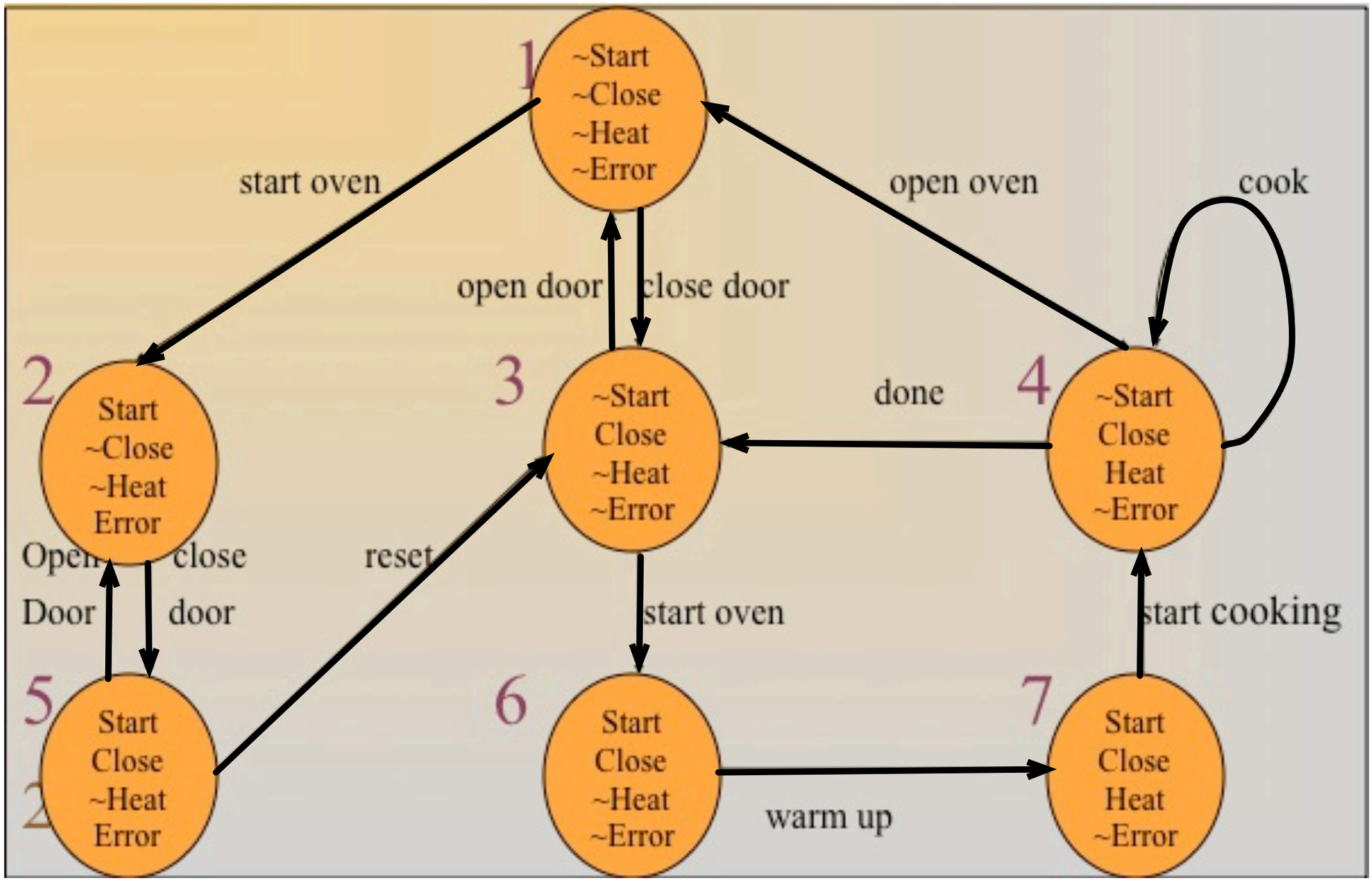
Overview of Model Checking



Overview of Model Checking

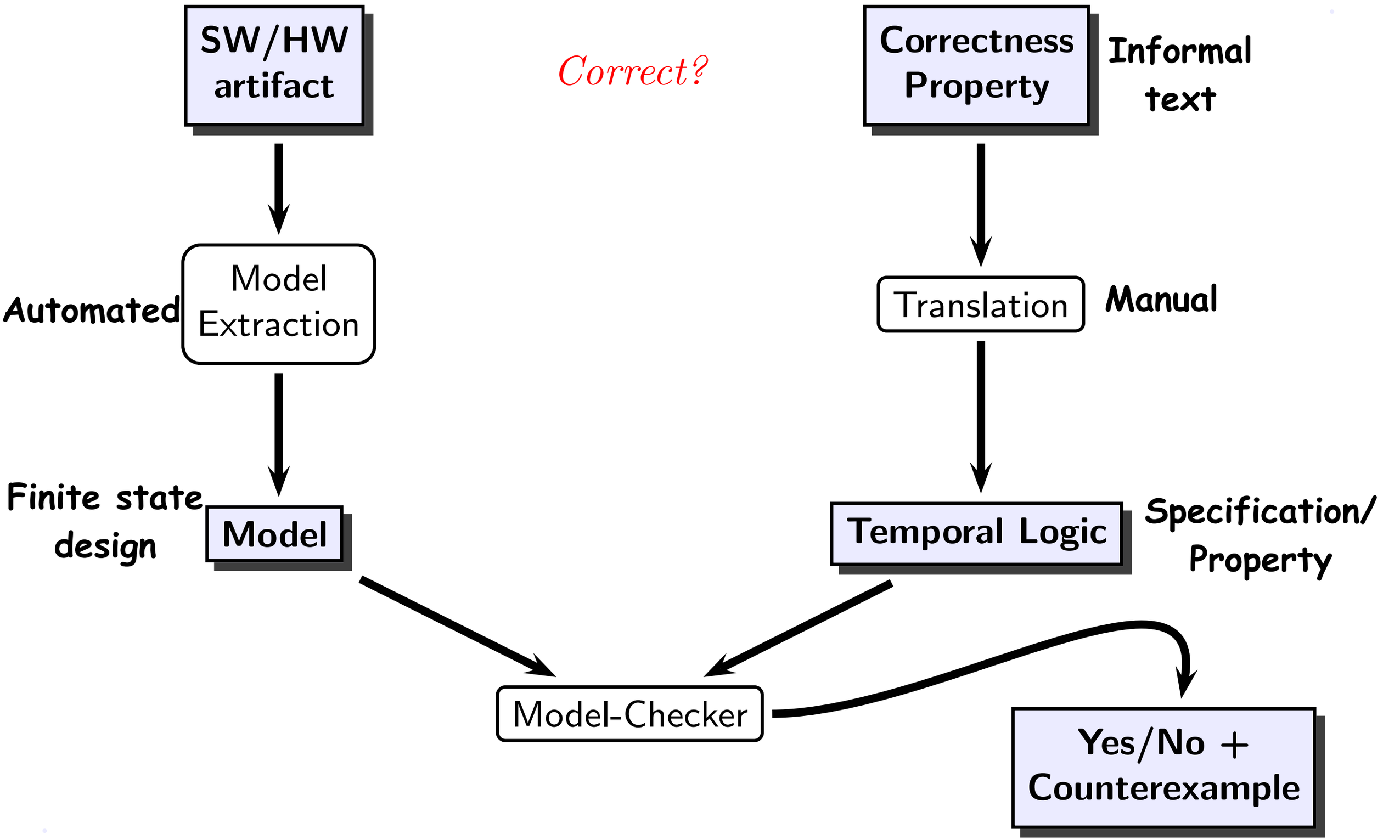


Finite State Design

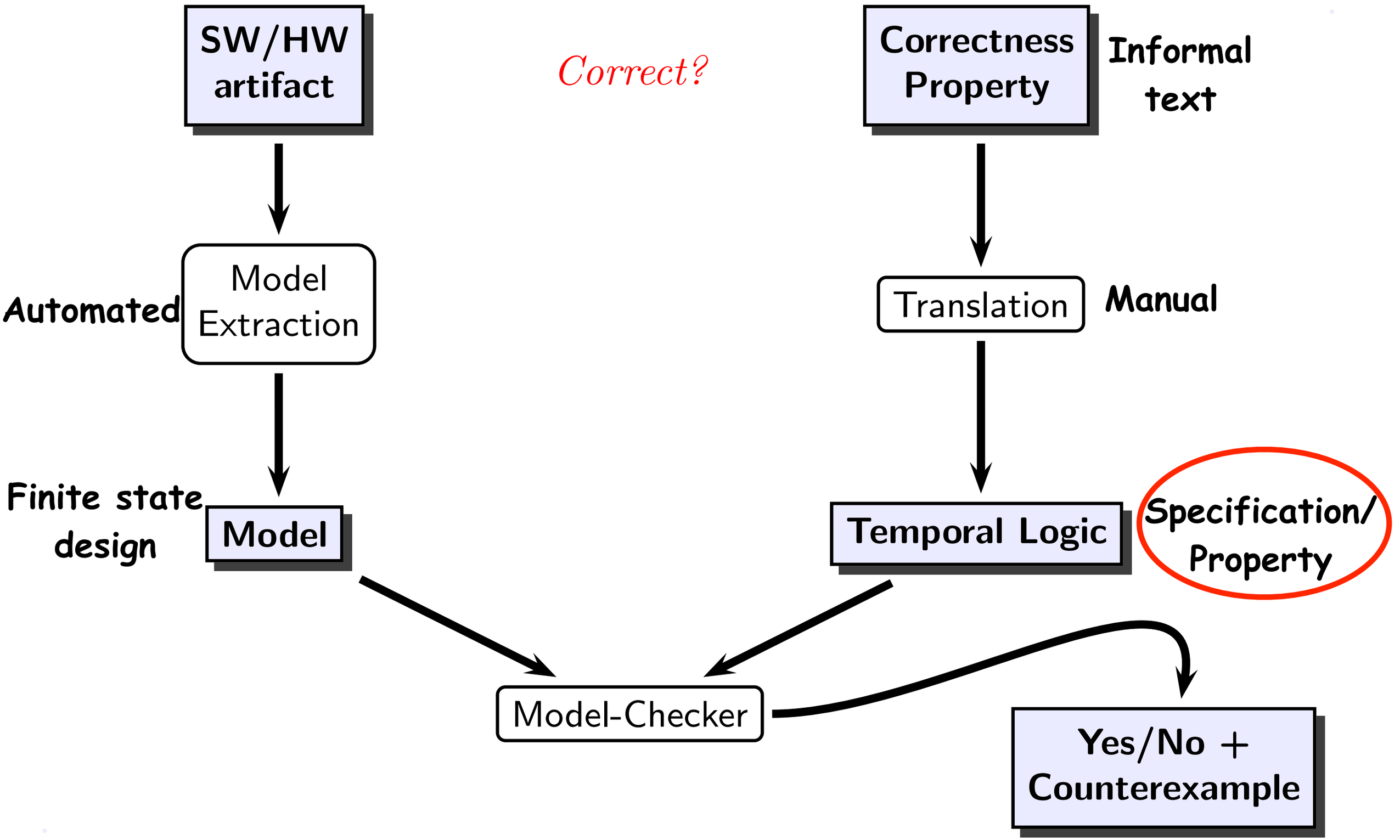


State transition graph describes the system evolving over time

Overview of Model Checking



Overview of Model Checking



System Properties

➜ The oven does not heat up until the door is closed

➜ Not heat_up holds until door_closed

➜ (¬ heat_up) U door_closed

Properties in Temporal Logic

➜ Express properties of event orderings in time

➥ Mutual exclusion

$$\text{always } \neg(\text{proc1} \wedge \text{proc2})$$

➥ Non starvation

$$\text{always } (\text{request} \Rightarrow \text{eventually granted})$$

➥ Sanity check

$$\text{eventually request}$$

➥ Communication protocols

$$\neg\text{get-message until send-message}$$

➥ Fairness

$$\text{always eventually control-granted}$$

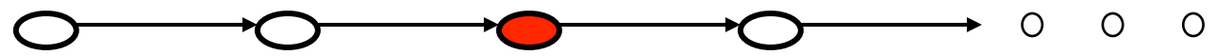
LTL - Linear Time Logic (Pn 77)

➜ Determines patterns of infinite traces

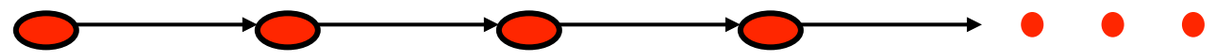
➥ Every moment has a unique successor

➢ The symbol 'p' is an atomic proposition, e.g., "Device Enabled"

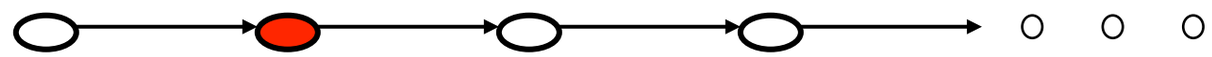
➢ F p - p holds sometime in the future



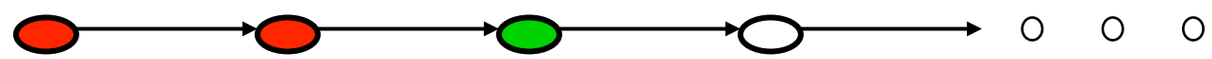
➢ G p - p holds globally in the future



➢ X p - p holds next time



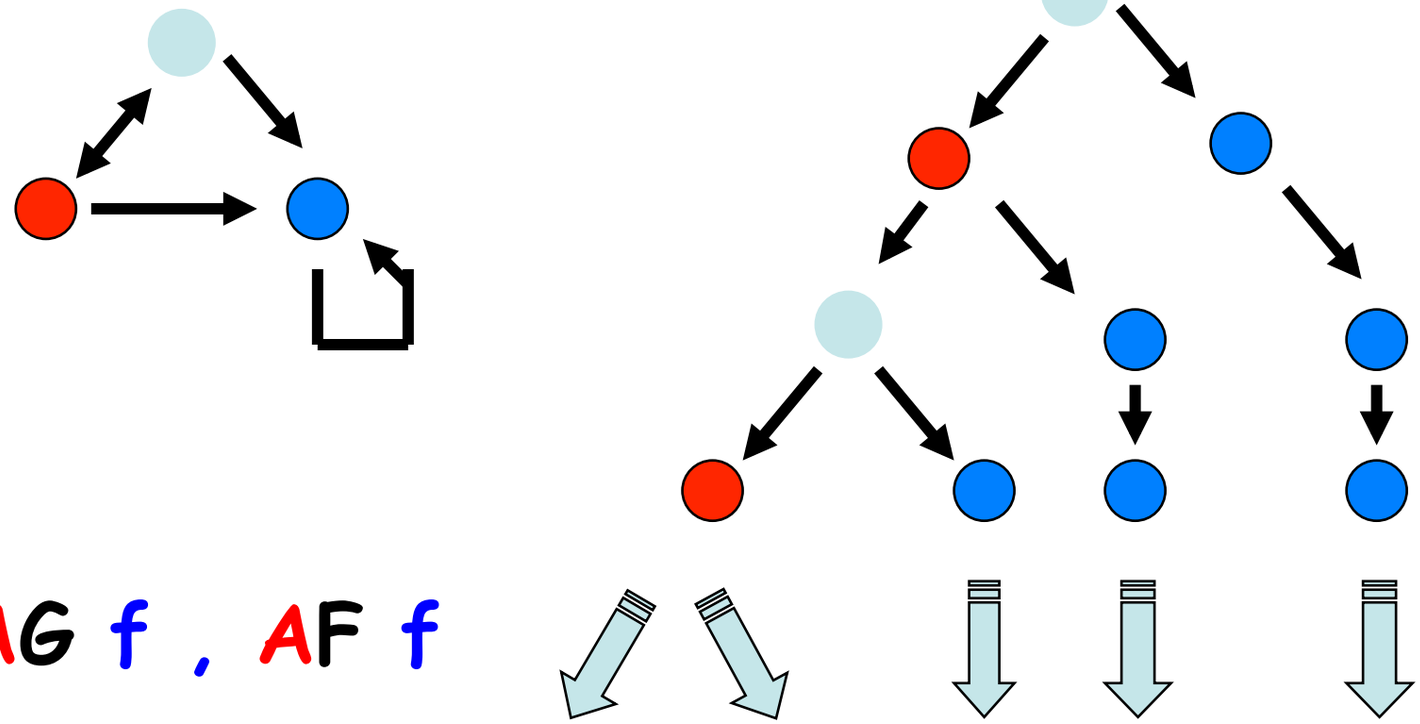
➢ p U q - p holds until q holds



CTL - Computational Tree Logic (CES 83)

➜ Determines patterns of infinite trees

➥ Every moment has several successors



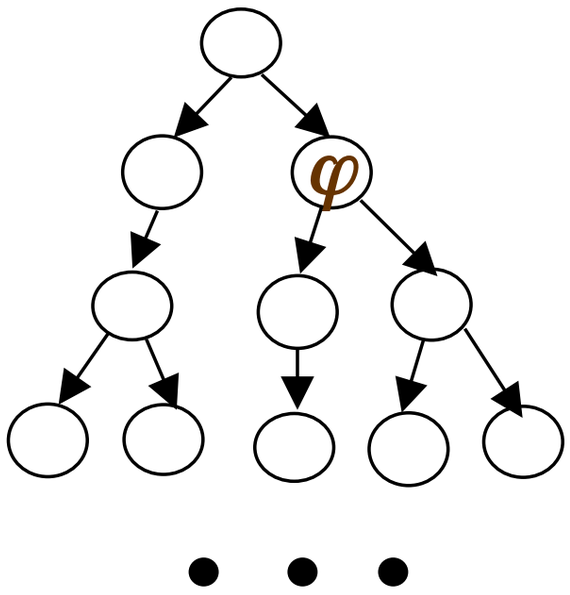
Universal formulas:

- AX f, A(f U g), AG f , AF f

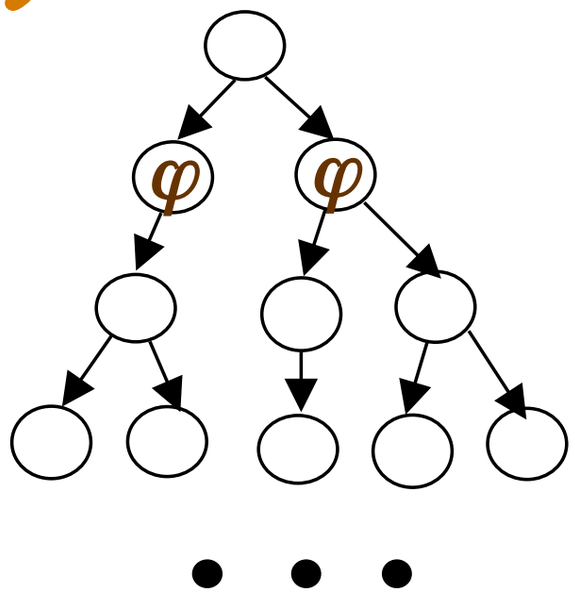
Existential formulas:

- EX f, E(fUg), EG f , EF f

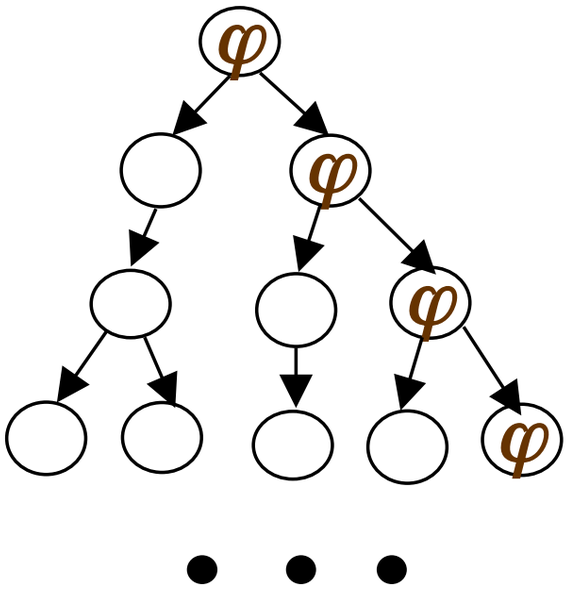
CTL (Cnt'd)



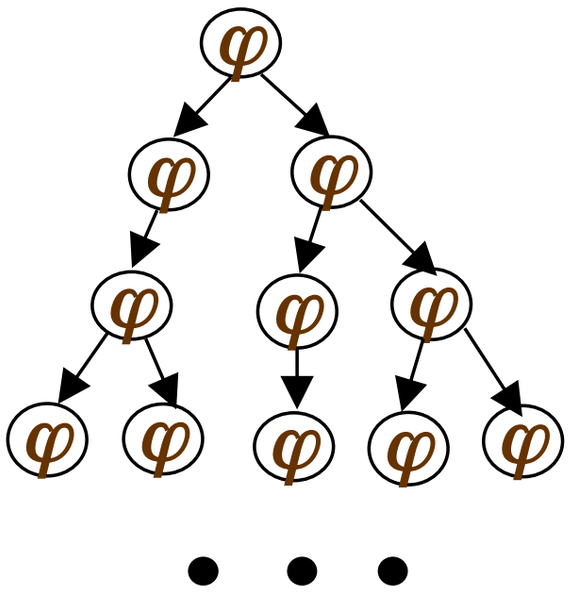
EX (exists next)



AX (all next)

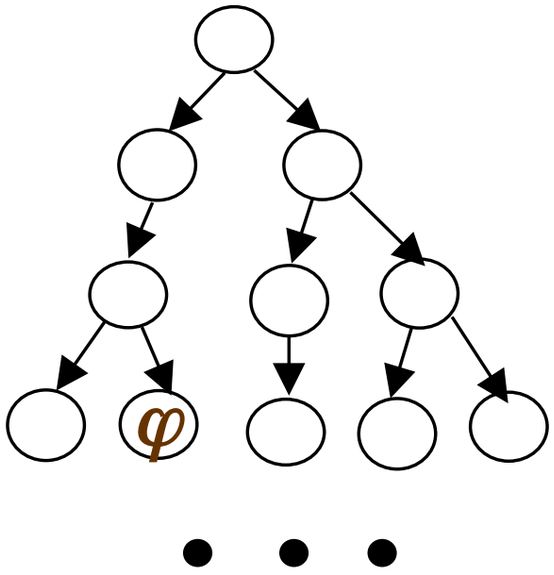


EG (exists global)

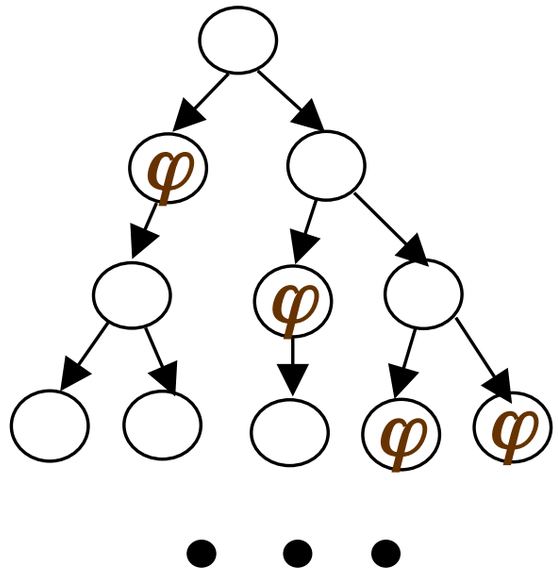


AG (all global)

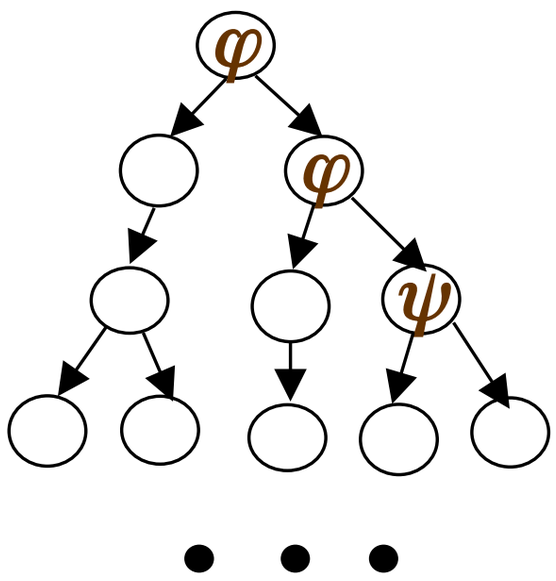
CTL (Cnt'd)



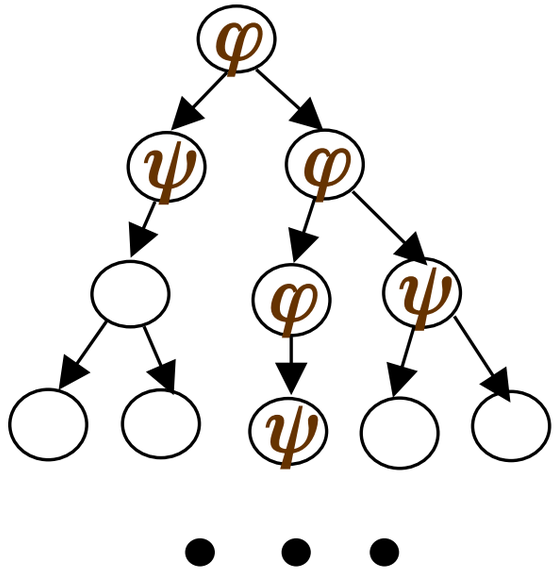
EF (exists future)



AF (all future)

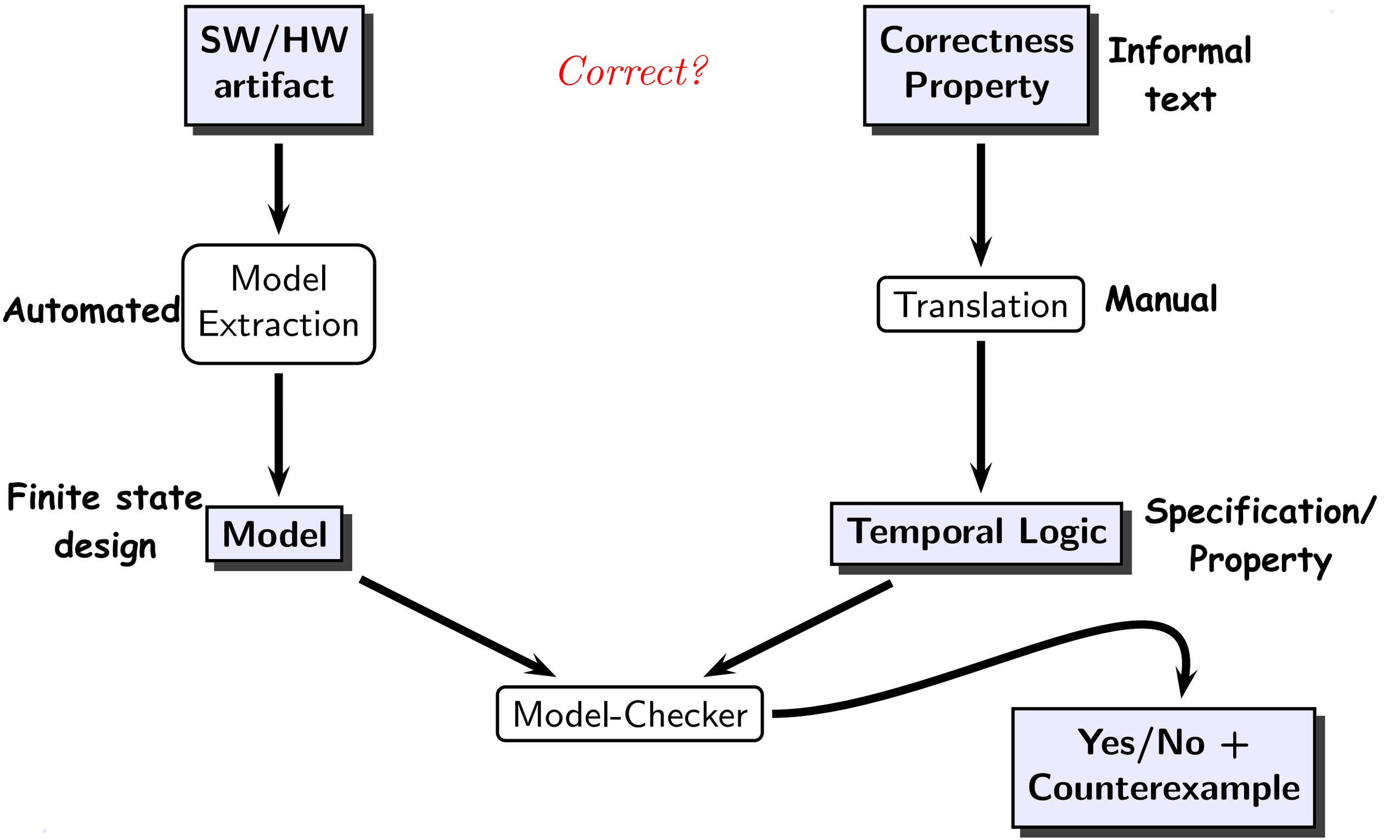


EU (exists until)

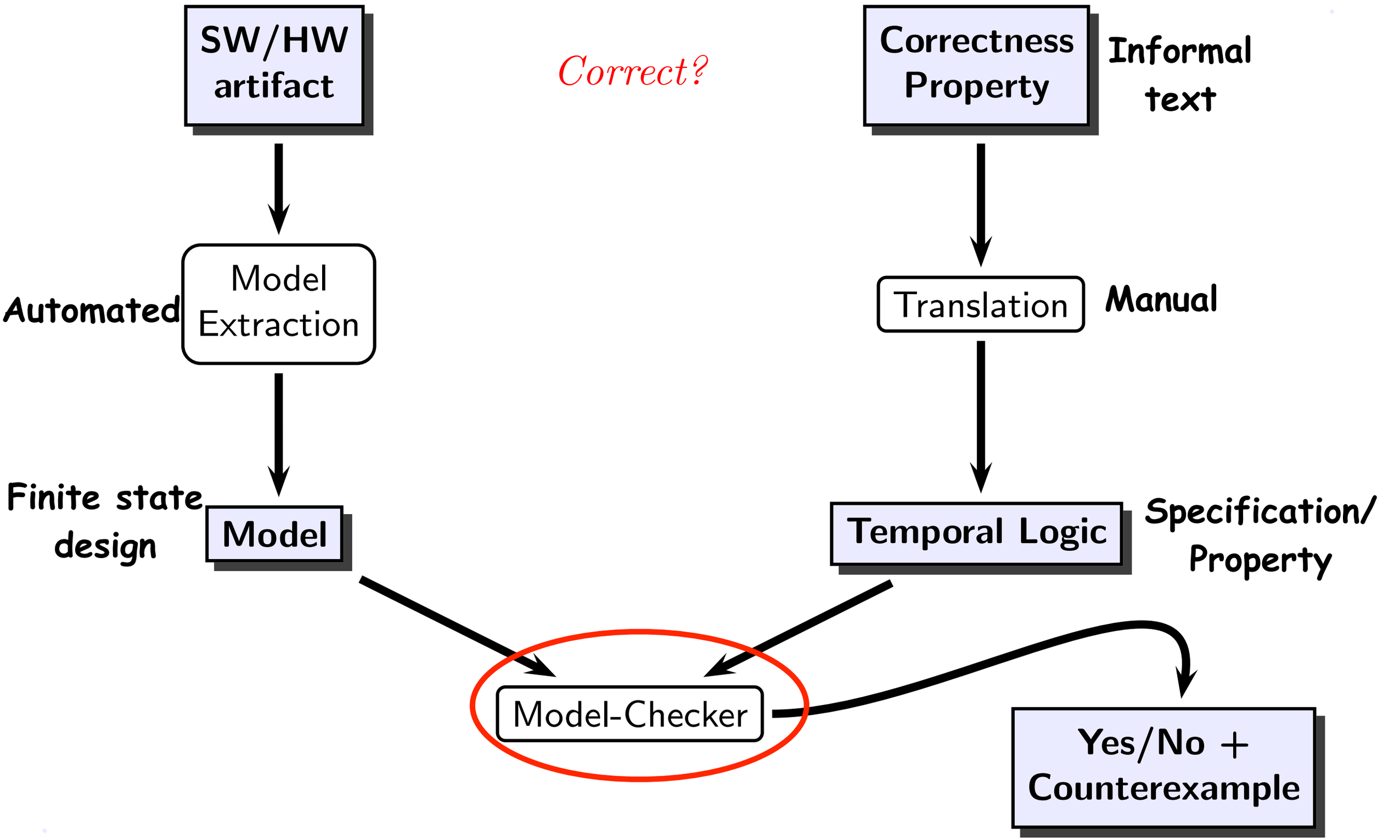


AU (all until)

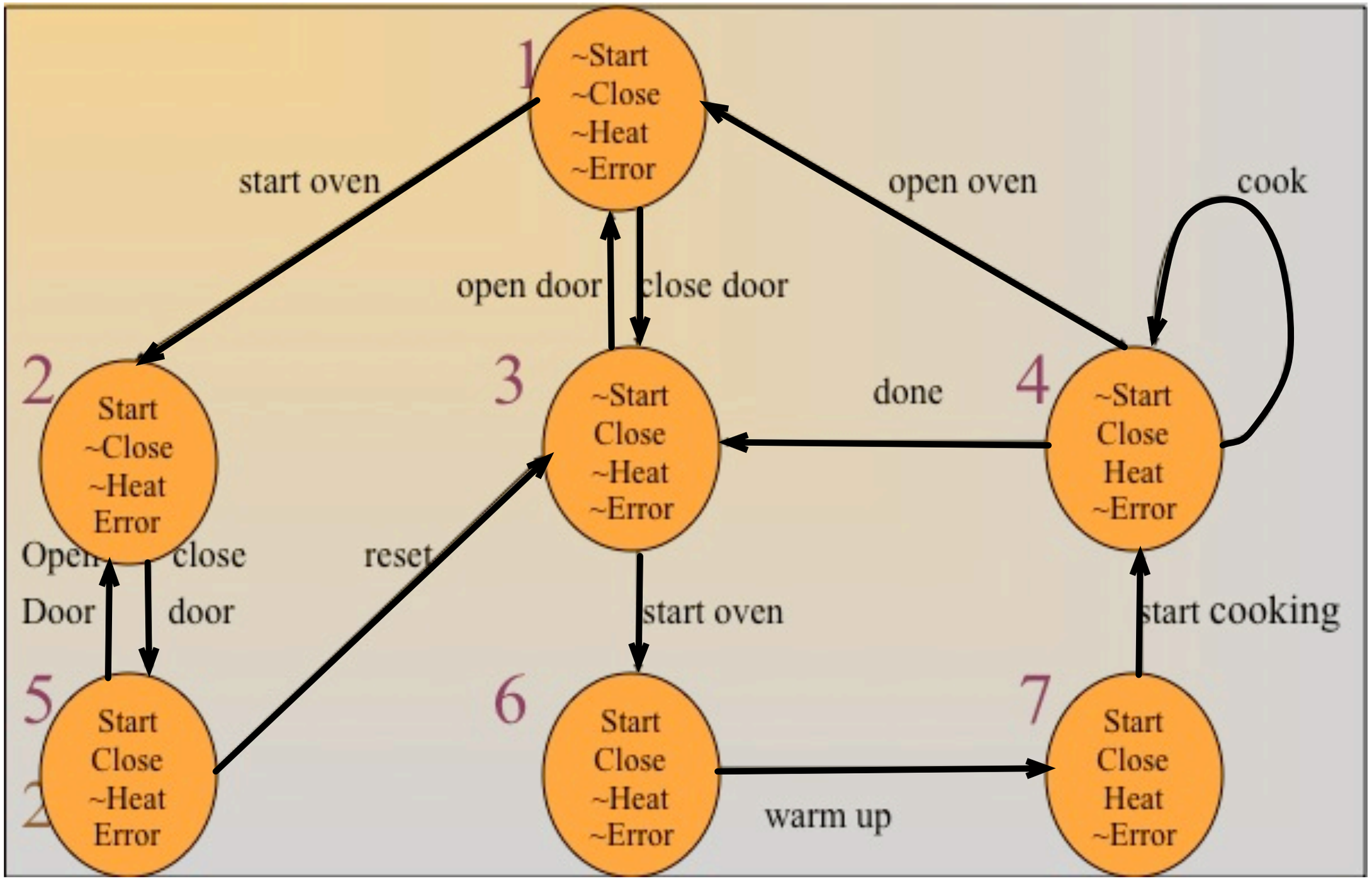
Overview of Model Checking



Overview of Model Checking



Model Checking Algorithm



(¬ heat_up) U door_closed

Model Checking Algorithm (CTL)

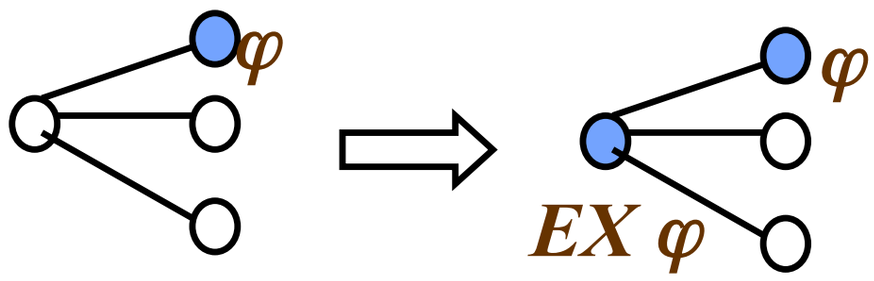
- **Receive:**
  - **Finite state Design**
  - **Temporal logic formula φ**
- **Assumptions:**
  - **Finite number of processes**
    - **Each having a finite number of finite-valued variables**
  - **Finite length of a CTL formula**
- **Algorithm:**
  - **Label states of *K* with subformulas of that φ are satisfied there and working outwards towards φ.**
  - **Output states labeled with φ**

Example: EX AG ($p \rightarrow$ E[p U q])

MC Algorithm (CTL) (Cnt'd)

EX φ

- **Label any state with EX φ if any of its successors are labeled with φ**



AF φ

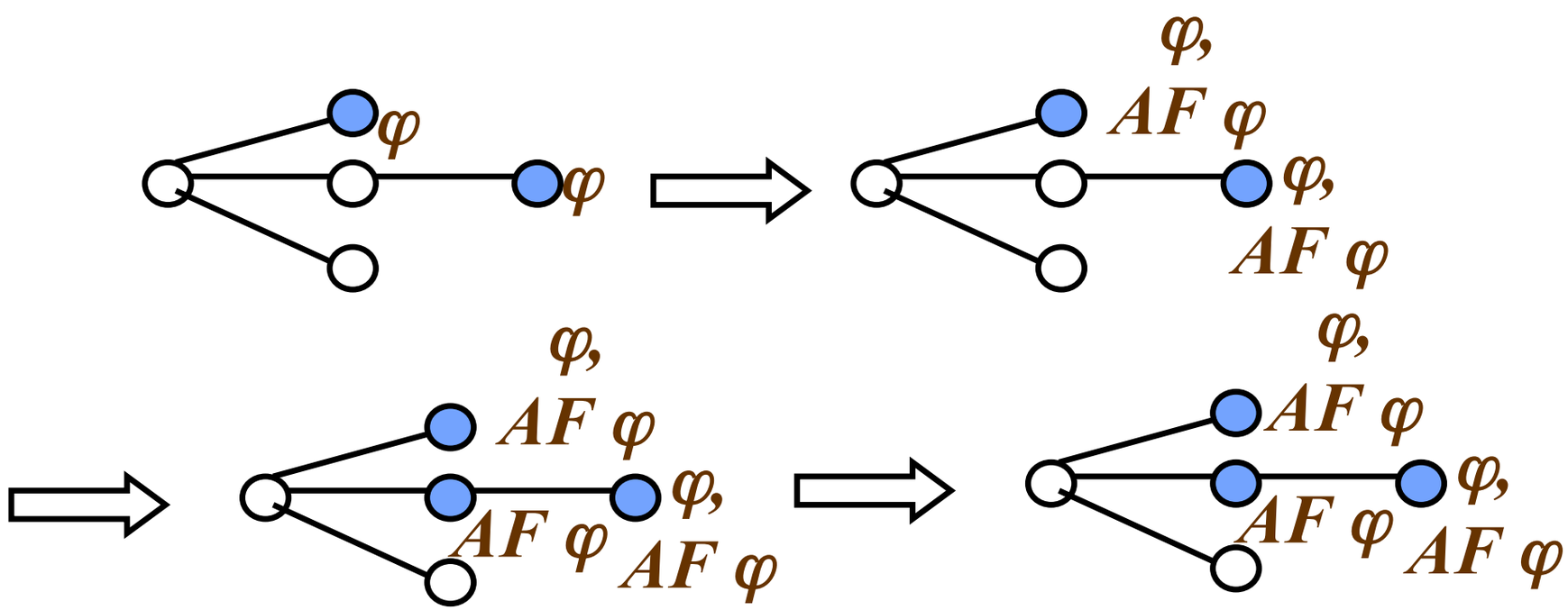
- **If any state s is labeled with φ, label it with AFφ**
- **Repeat:**

  label any state with AFφ

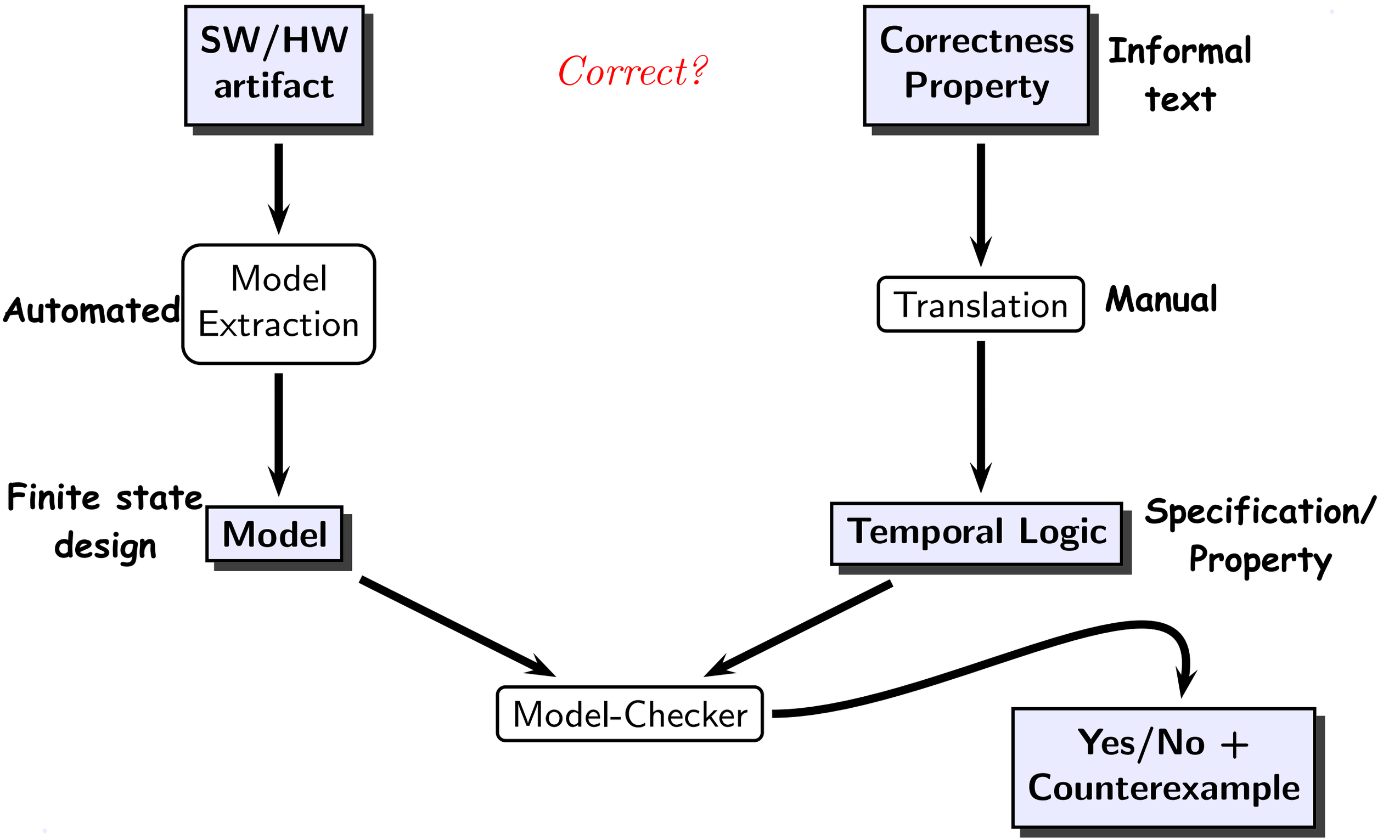
  if all of its successors

  are labeled with AFφ

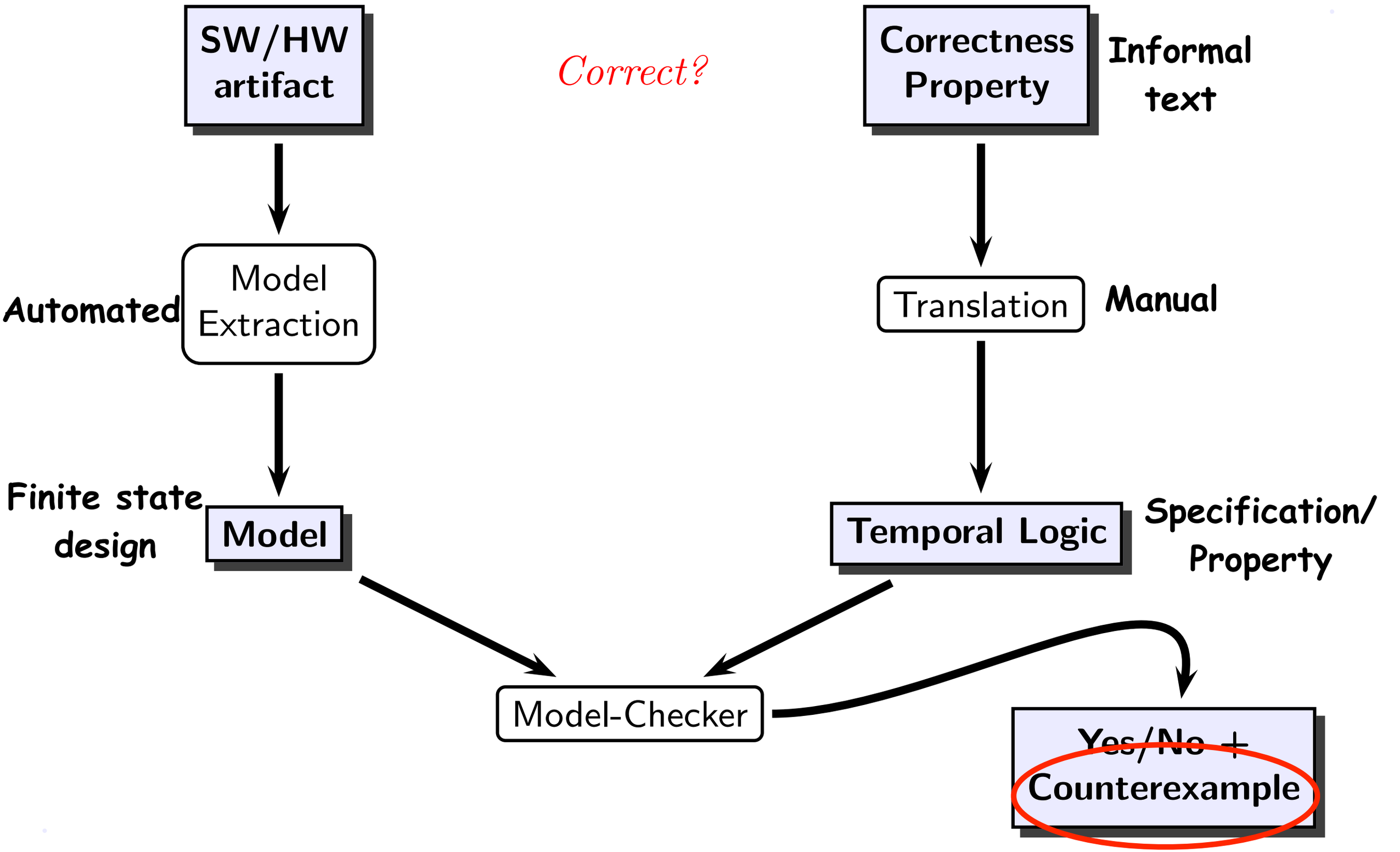
until there is no change



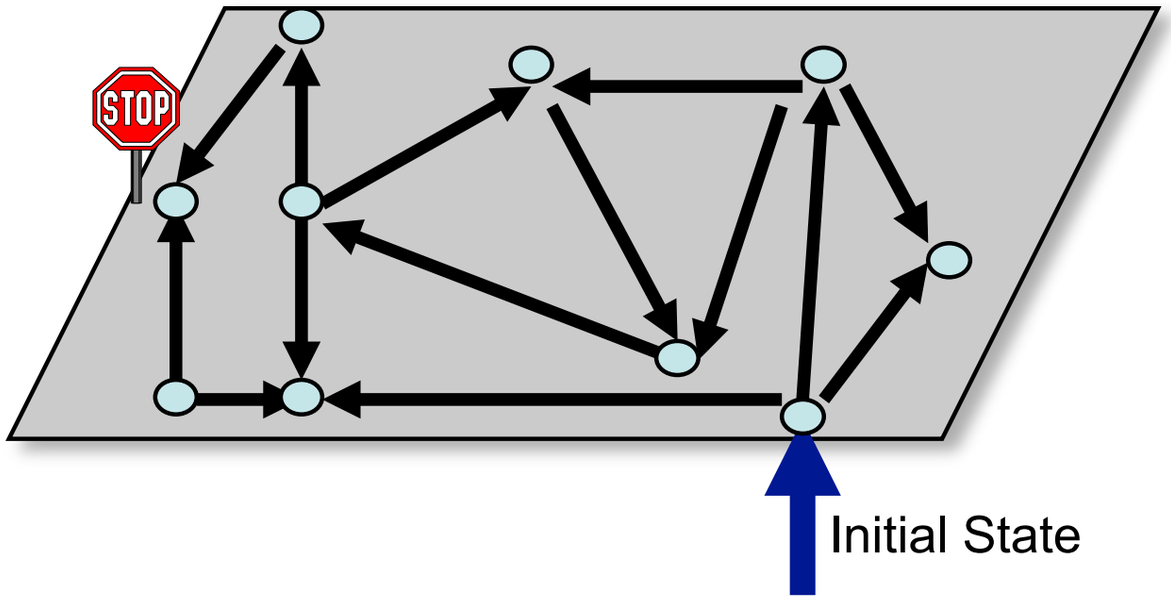
Overview of Model Checking



Overview of Model Checking



Counterexamples

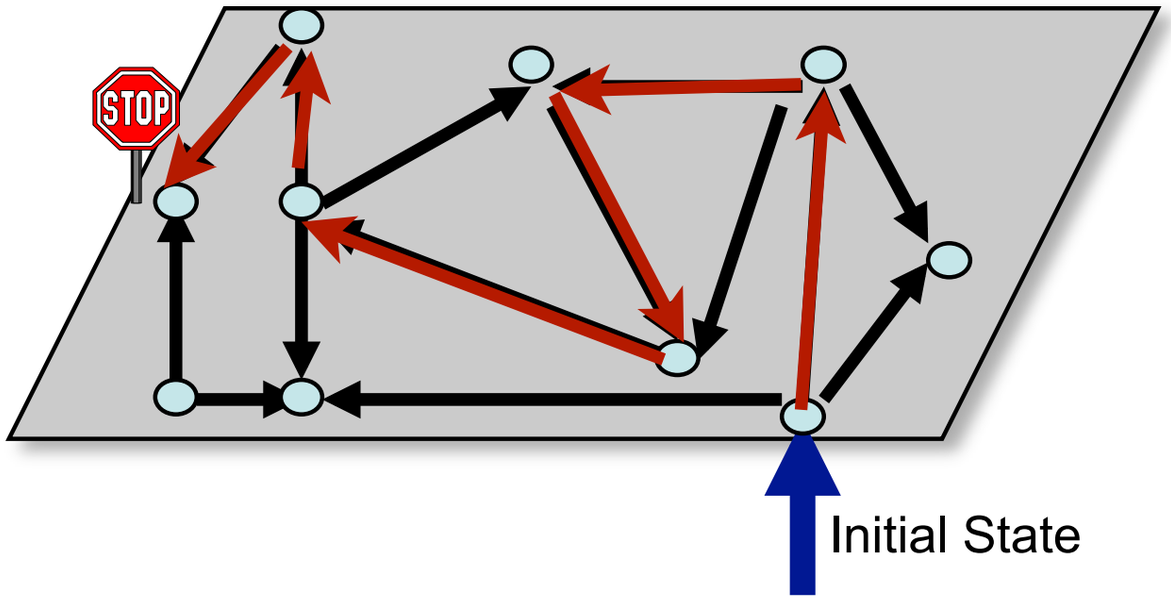


Safety Property:

bad state  unreachable

Counterexample

Counterexamples



Safety Property:

bad state  unreachable

Counterexample

Main Disadvantages of Model Checking

➜ Proving a program does not help you understand it!

➜ Temporal logic specifications are ugly

➥ Depends on who is to write them

➜ Writing specification is hard

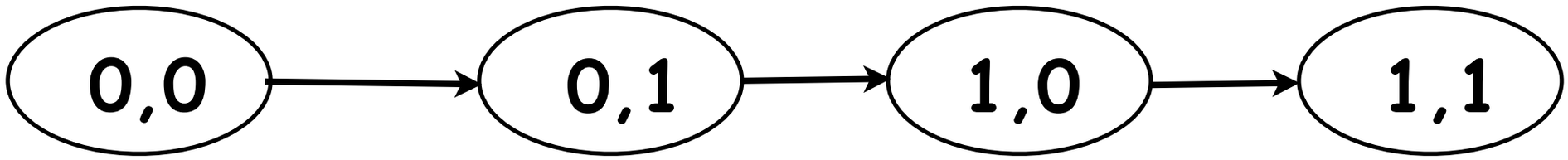
➥ True, but perhaps partially a matter of education

➜ State explosion is a major problem

➥ Model checking may not scale!

State Explosion problem

→ 2-bit counter



→ n-bit counter

➥ 2^n states

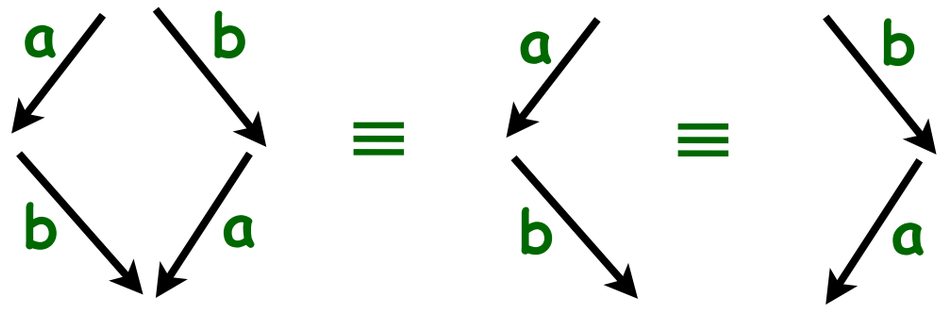
→ Parallel composition of processes

➥ m processes, each having n states

➢ #states of the composition = n^m

Some Remedies

- Symbolic Model Checking (McMillan 92)
  - Symbolic representations for state transition systems
    - using BDDs (Binary Decision Diagrams)
    - scales to 10^20 states
- Partial State Reduction (Godefroid 90)
  - Avoid checking different inter-leavings of independent actions
    - Asynchronous systems, 'a' and 'b' are independent actions



    - Implemented in Gerard Holzmann's SPIN

Some Remedies (Cnt'd)

- Bounded Model Checking (Biere et. al. 99)
  - Translate the model and the specification into a propositional formula
    - Use fast SAT solvers to check satisfiability
  - Unbounded MC
    - Can a given property fail over time?
  - Bounded MC
    - Can a given property fail in k-steps?
      - where k is bounded
  - Suitable for falsification, not verification

Dealing with Very Complex Systems

→ Special techniques are needed when Symbolic methods, the partial order reduction, and bounded MC do not work

➥ Compositional Reseaning

➢ Jamieson M. Cobleigh, Dimitra Giannakopoulou, Corina S. Pasareanu: Learning Assumptions for Compositional Verification. TACAS 2003: 331-346

➥ Abstraction

➢ Patrick Cousot, Radhia Cousot: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. POPL 1977: 238-252

➥ Symmetry Reduction

➢ Edmund M. Clarke, E. Allen Emerson, Somesh Jha, A. Prasad Sistla: Symmetry Reductions inModel Checking. CAV 1998: 147-158

➥ Induction and Parameterized Verification

➢ E. Allen Emerson, Kedar S. Namjoshi: Verification of Parameterized Bus Arbitration Protocol. CAV 1998: 452-463

Future Challenges

- Model Checking Software Programs
  - Why software is so difficult?
    - large/unbounded base types: int, float, string
    - User-defined types/classes
    - Pointer/aliasing + unbounded number of heap allocated cells
    - Procedure calls/recursion/overloading
    - Concurrency + unbounded number of threads
    - etc.
  - Some existing tools
    - SLAM
      - Developed at Microsoft Research early 2000 for model checking Windows device drivers
      - Uses static analysis to extract a finite model from device drivers code written in C
    - Other tools
      - Bandera (Kansas State), MAGIC (CMU), SATABS (CMU), BLAST (Berkley), F-Soft (NEC)

Future Challenges

- ➜ Exploiting the Power of SAT, Satisfiability Modulo Theories (SMT)
- ➜ Compositional Model Checking of both Hardware and Software
- ➜ Verification of Embedded Systems (Timed and Hybrid Automata)
- ➜ Model Checking and Theorem Proving (PVS, STEP, SyMP, Maude)
- ➜ Probabilistic and Statistical Model Checking
- ➜ Interpreting Counterexamples
- ➜ Scaling up even more!!

Some Model Checking Tools

- SPIN
  - http://spinroot.com/spin/whatispin.html
- NuSMV
  - http://nusmv.fbk.eu/
- Java Path Finder
  - http://babelfish.arc.nasa.gov/trac/jpf
- UPPAAL
  - http://www.uppaal.org/
- PRISM
  - http://www.prismmodelchecker.org/
- Contact me for more info!

Acknowledgments

➜ These slides are based on the lecture notes from

- ➥ Ed Clarke @ CMU, http://www.cs.cmu.edu/~emc/
- ➥ Orna Grumberg @ Technion, http://www.cs.technion.ac.il/~orna/
- ➥ Marsha Chechik @ UofT, http://www.cs.toronto.edu/~chechik/

References

- Model Checking, Edmund M. Clarke, Orna Grumberg and Doron A. Peled, MIT Press, 1999
- Logic in Computer Science: Modelling and Reasoning About Systems, Michael Huth and Mark Ryan, Cambridge University Press, 2004
- Symbolic Model Checking, Kenneth L. McMillan, Kluwer, 1993
- Clarke, E. M.; Emerson, E. A.; Sistla, A. P. (1986), "Automatic verification of finite-state concurrent systems using temporal logic specifications", *ACM Transactions on Programming Languages and Systems* 8: 244
- Queille, J. P.; Sifakis, J. (1982), "Specification and verification of concurrent systems in CESAR", International Symposium on Programming
- The Spin Model Checker: Primer and Reference Manual, Gerard J. Holzmann, Addison-Wesley
- Patrice Godefroid, Pierre Wolper: Using Partial Orders for the Efficient Verification of Deadlock Freedom and Safety Properties. *CAV* 1991: 332-342
- Edmund M. Clarke, Armin Biere, Richard Raimi, Yunshan Zhu: Bounded Model Checking Using Satisfiability Solving. Formal Methods in System Design 19(1): 7-34 (2001)
- Thomas Ball, Sriram K. Rajamani: The SLAM project: debugging system software via static analysis. *POPL* 2002: 1-3

Thank You!

Questions?