On Verification of Parameterized Distributed Systems

Moscow State University
Computational Mathematics and Cybernetics
Igor V. Konnov

INTAS meeting

Verification by Model Checking

- Given a program P and its specification φ build a model M of P on some appropriate abstraction level.
- Check, whether M satisfies φ.
- Otherwise, generate a counter-example.

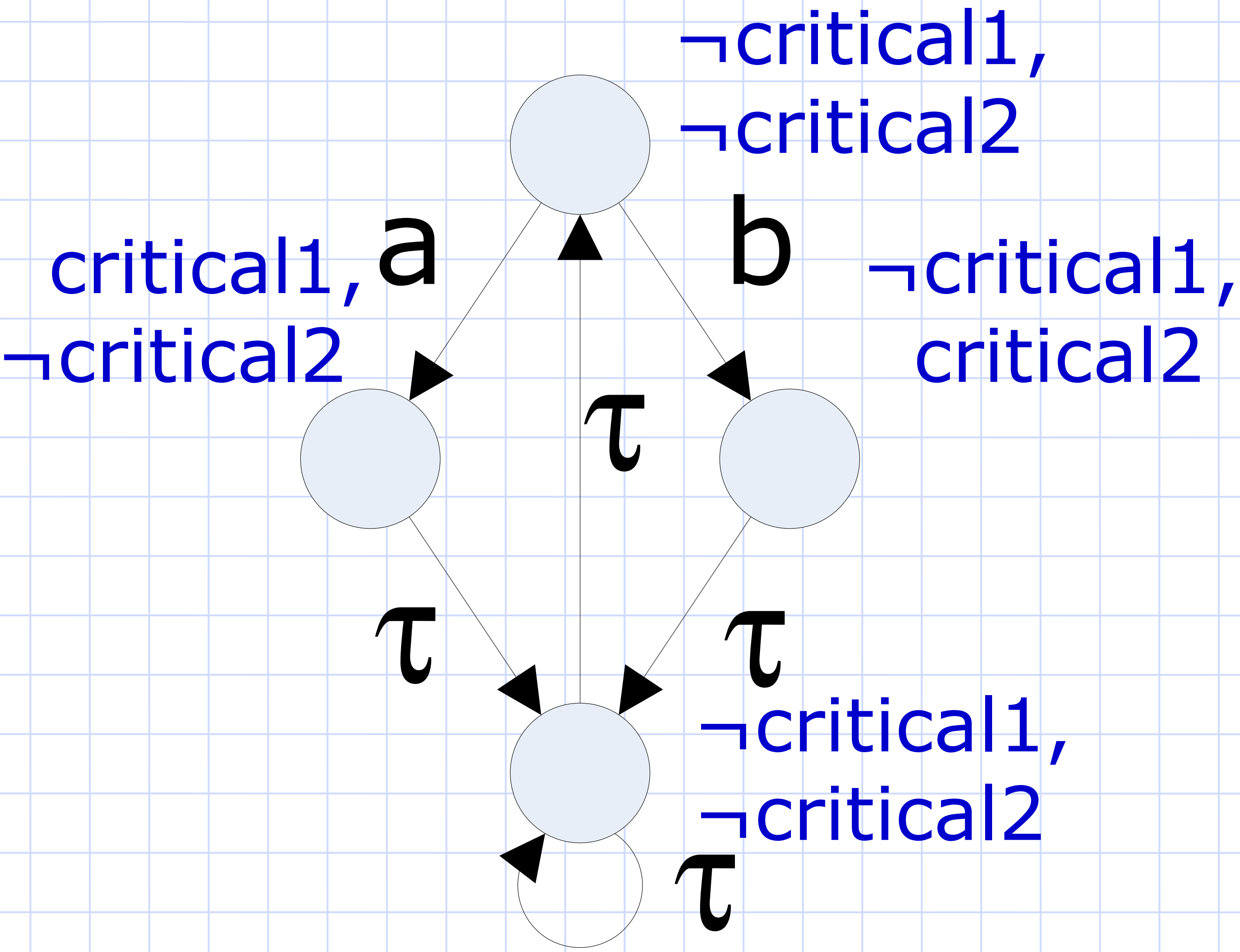
The Main Problems of MC

- To choose some suitable formalism for representing abstract models of programs.
- To choose some expressive formal language for representing specifications.
- To develop an efficient model-checking algorithm.

Modelling Distributed Systems

- Individual processes are modelled by Labelled Transition Systems.
- Model of distributed system is an asynchronous parallel composition of LTSes with rendezvous message passing (synchronous communication).

Example of Model



Specifications

- We specify program and model behavior by formulas of temporal logic ACTL*-X.
- Examples:
  - AF(critical1)
  - AG($\neg$critical1 $\wedge$ $\neg$critical2)
  - $\neg$receive2 AU send1.

Parameterized Distributed Systems

- Many distributed algorithms are parameterized by:
  - <u>the number of similar processes</u>,
  - the size of data types,
  - the size of communication channels.
- Many distributed algorithms have unbounded data types.

Models Parameterized by Number of Processes

- We study the verification problem for families of distributed systems $\{M_n\}$, $n >= 1$
- Every system M_n is composed of some distinguished process Q and a number of isomorphic processes that are instances of the same prototype process P.
- $M_n = Q \,||\, P \,||\, P \,||\, \ldots \,||\, P$.

Specifications of Parameterized Systems

- To specify a behavior of parameterized distributed system $M_n = Q \,||\, P \,||\, P \,||\, \ldots \,||\, P$ we may:
  - either specify a desirable behaviour of the distinguished process Q; in this case we deal with the same specification for the whole family of systems $\{M_n\}$
  - or consider parameterized family of formulae φ_n ;
  - or use formulae over regular expressions.

Parameterized Model Checking

- For a family S_n of specifications and a family M_n of models we need to check, whether $M_n \models S_n$.
- The problem is undecidable [Apt, Kozen, 1986].
- The problem is undecidable even for ring networks that are composed of very simple processes.

PMC by Invariants

- Suppose that we are given some partial order $\leq$ on LTSes which complies with the following requirements:
  - It is **conservative** under a class of specifications Ψ. For any $\psi \in \Psi$ prop. $A \leq B$ *and* $B \models \psi$ *implies* $A \models \psi$
  - It is **monotonic**. Relation $A \leq B$ and $C \leq D$ implies $A \,||\, B \leq C \,||\, D$.
- Then to check that $M_n \models \psi$ holds for every n it is sufficient to find LTS I (invariant) such that $Q \,||\, P \;\; I$ and $I \,||\, P \;\; I$, hold, and check that $I \models \psi$.

Partial Orders on LTSes

- We should choose some order ≤
- Some partial orders on LTSes that may be used for the purpose of invariant-aided parameterized verification:
  - trace inclusion,
  - (strong) simulation,
  - weak simulation,
  - branching simulation,
  - block simulation (close to visible simulation),
  - quasi-block simulation.

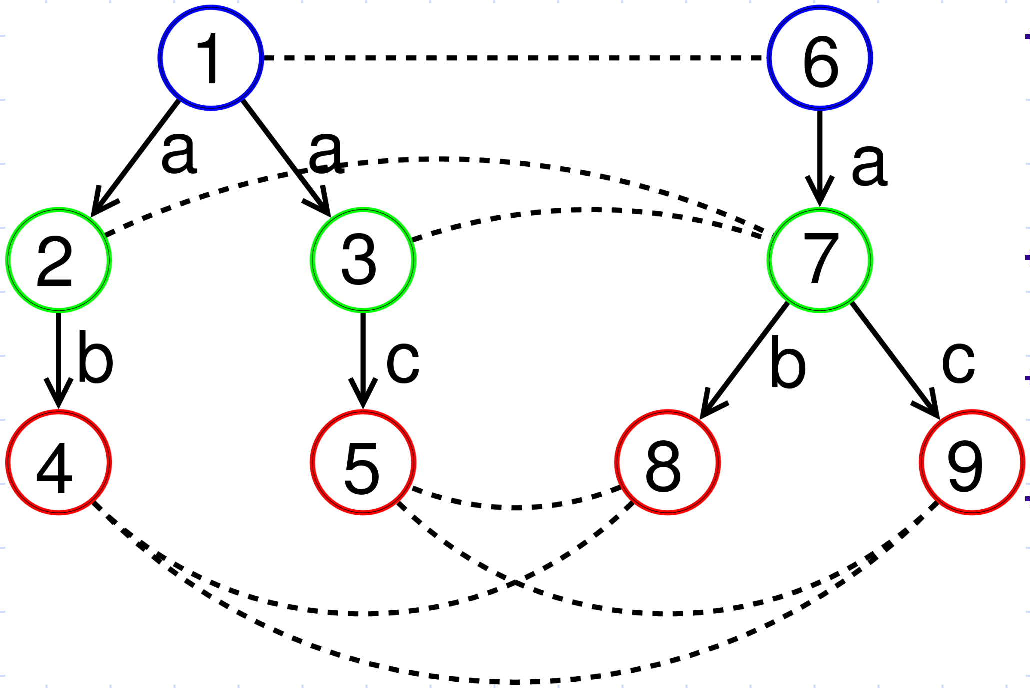
How to find an invariant?

- To guess it...
- To build another abstraction of *P* using heuristics and specification.
- To find N such that $M_{N+1} = M_N \parallel P \leq M_N$. In this we have $M_{N+2} = (M_N \parallel P) \parallel P \leq M_N \parallel P$, and for every n, n >= N + 1, $M_{n+1} \leq M_N$ holds. Thus it is sufficient to check models $M_1, \ldots, M_N$.

If we can't find an invariant

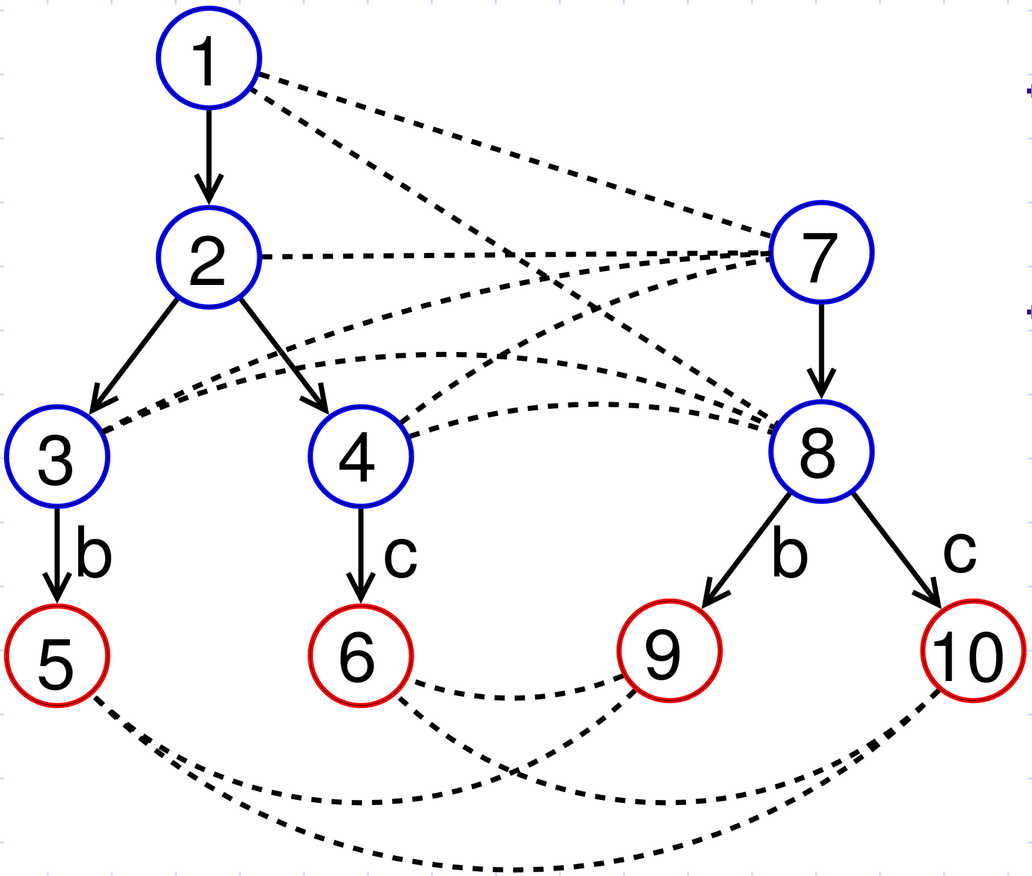
- Think more.
- Change the level of abstraction.
- Choose a more suitable partial order relation.
  - Strong simulation is applicable to synchronous systems, but it is poorely suited for finding an invariant of asynchronous systems (though it is possible with combination of abstraction [Clarke, Grumberg, Jha, 1997]).
  - To extend invariant based technique on asynchronous systems we introduce block and quasi-block simulations.

(Strong) Simulation



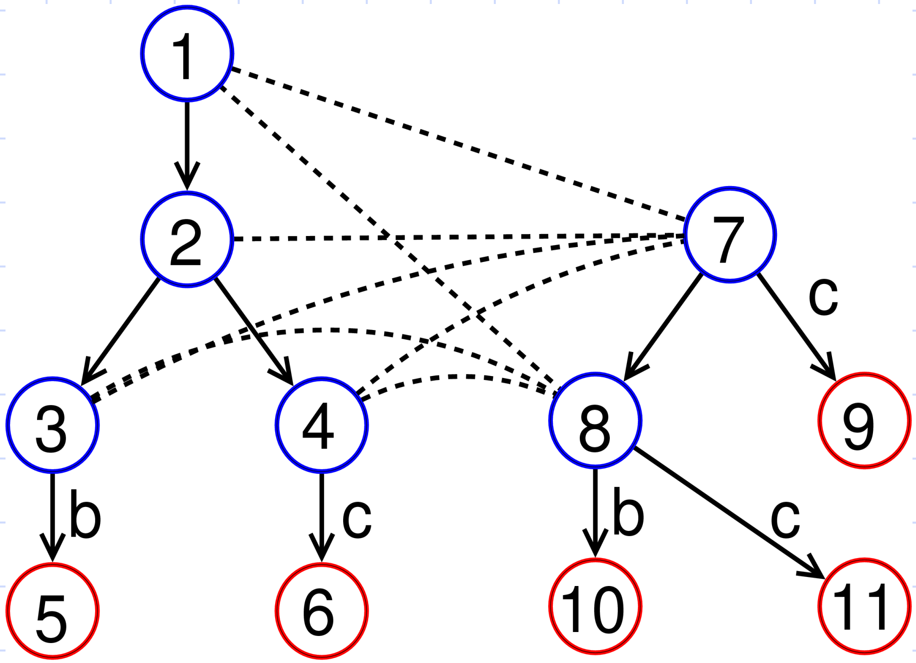
- Conservative under ACTL*
- Monotonic
- Easy to check!
- Too strong to us

Weak Simulation



- Neither conservative
- Nor monotonic

Block Simulation

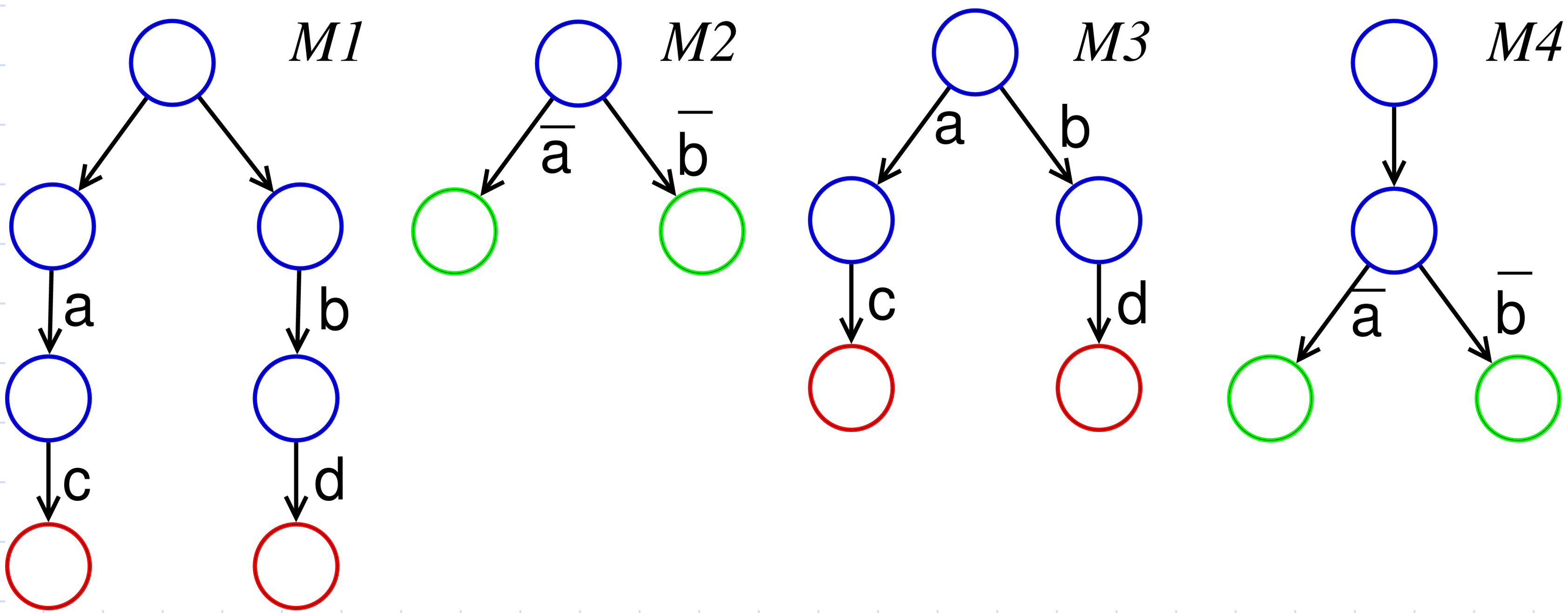


- Conservative under ACTL*-X
- Still not monotonic (but it is in some limited cases)

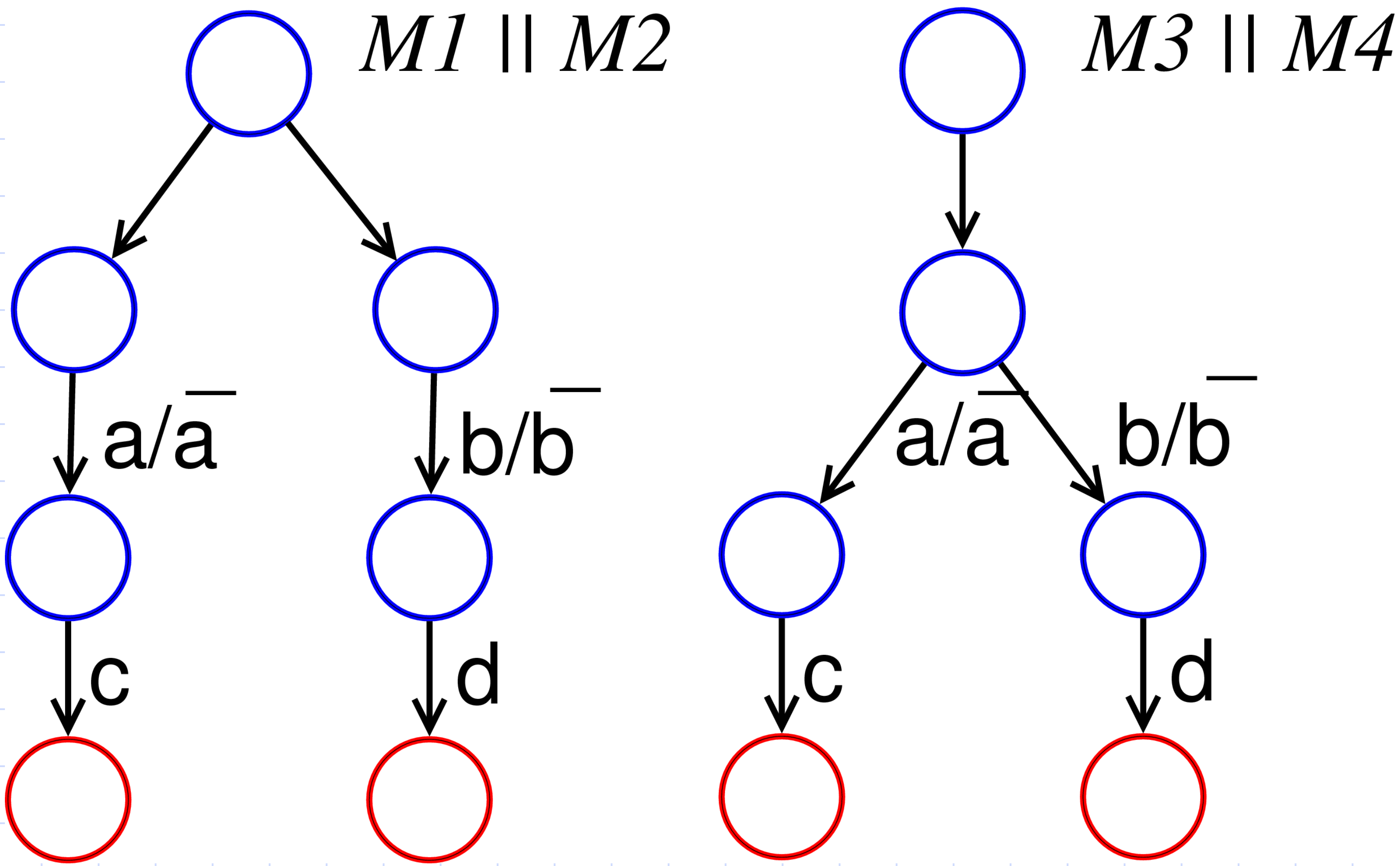
Definition of Block Simulation

- Let Mi = (Si, Si_0, Ai, Ri, Σi, Li), i = 1,2, be LTSes. Let Σ0 in Σ1 ∩ Σ2. H ∈ S1 × S2 is a block simulation iff for each (s_1, t_1) ∈ H:
  - L1(s_1) ∩ E0 = L2(t_1) ∩ E0,
  - For every finite block $s_1 -\tau\rightarrow s_2 -\tau\rightarrow \ldots -\tau\rightarrow s_m -a\rightarrow s_{m+1}$ there is a block $t_1 -\tau\rightarrow t_2 -\tau\rightarrow \ldots -\tau\rightarrow t_n -a\rightarrow t_{n+1}$ such that (s_{m+1}, t_{n+1}) ∈ H and (s_i, t_j) ∈ H
  - For any infinite block $s_1 -\tau\rightarrow s_2 -\tau\rightarrow \ldots$ from s_1 there is an infinite block $t_1 -\tau\rightarrow t_2 -\tau\rightarrow \ldots$ such that (s_i, t_j) in H.

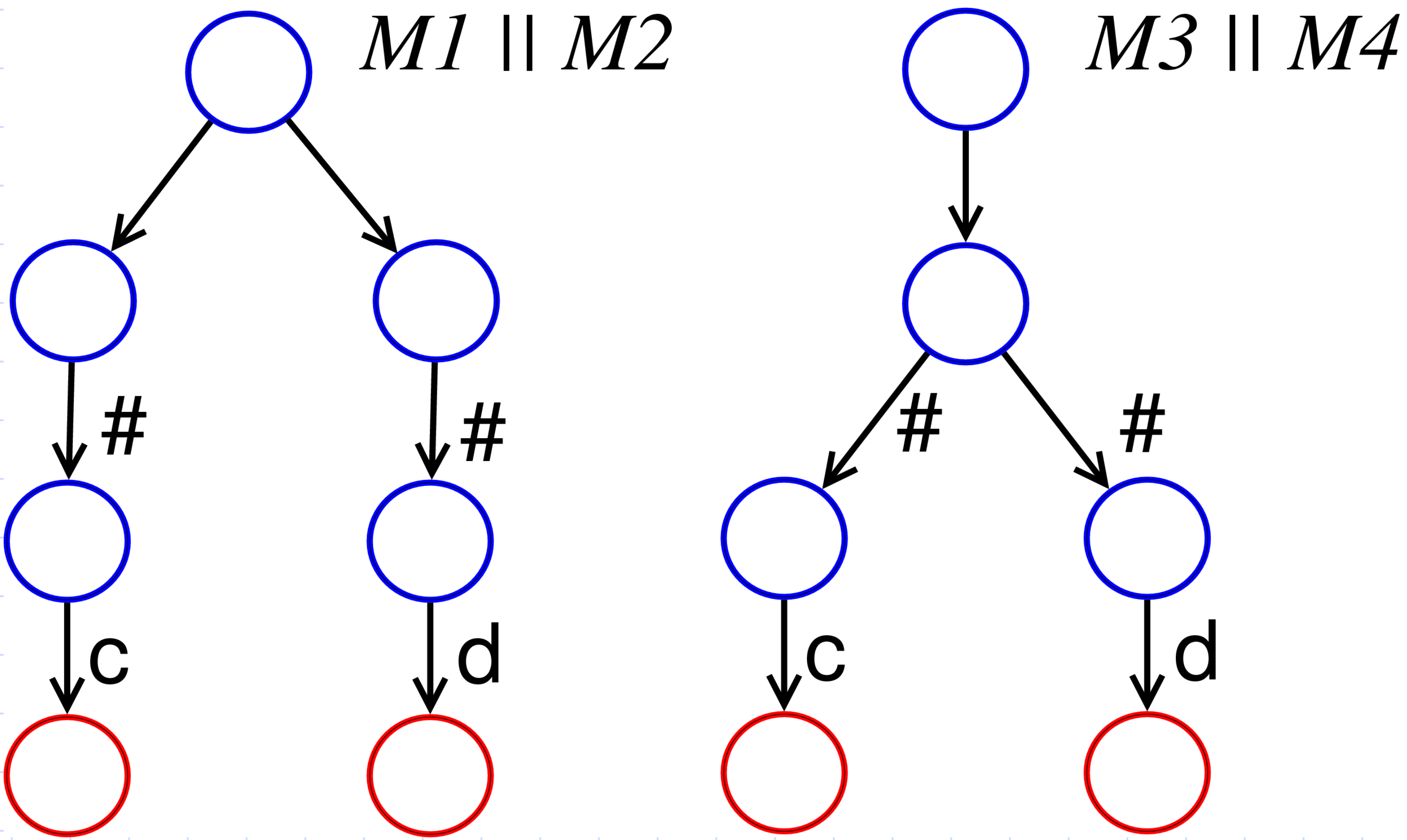
M3 block simulates M1, M4 block simulates M2



But the composition does not preserve block simulation



Quasi-block Simulation



Properties of Quasi-block Simulation

- Block simulation is a quasi-block simulation.
- As a consequence, quasi-block simulation is conservative under ACTL*-X.
- It is monotonic (if synchronization is performed in the same way in the both pairs of models).

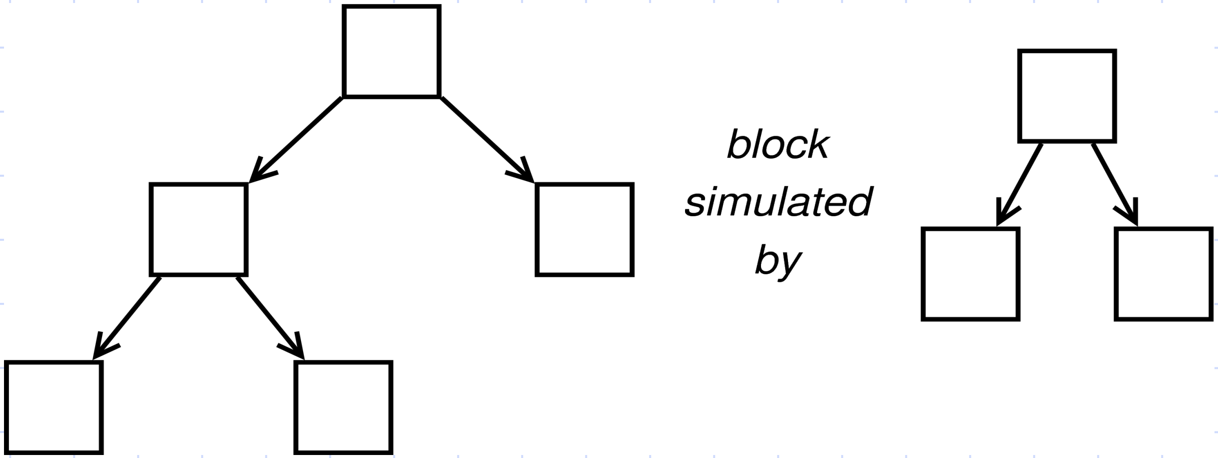
Our Verification Framework

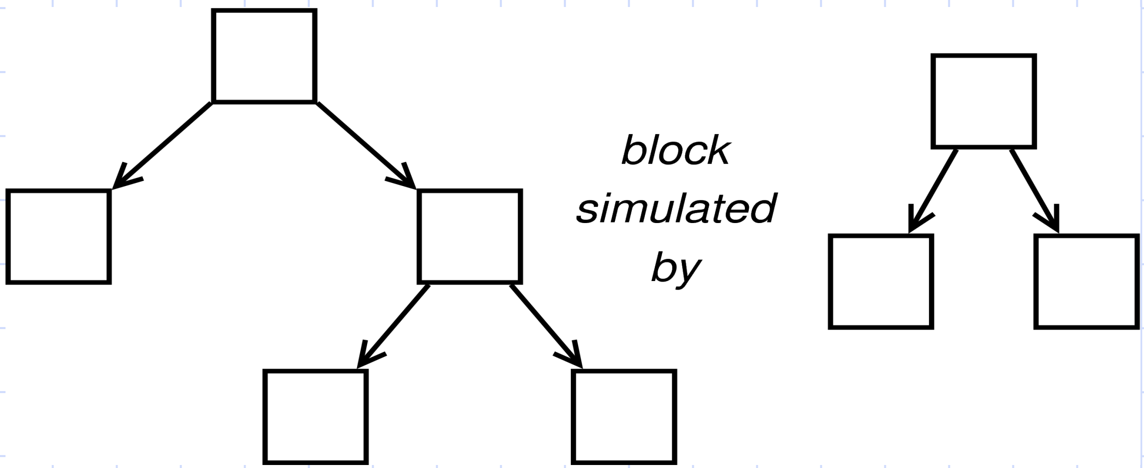
- Family of parameterized models is described by network grammars (as in [Clarke, Grumberg, Jha, 1995]).
- Fragments derived from the same non-terminal are checked against block simulation.
- If for some M it holds $M \| P \| \ldots \| P \leq M$, then invariant of non-terminal is found.

Example:
Tree Wave Algorithm

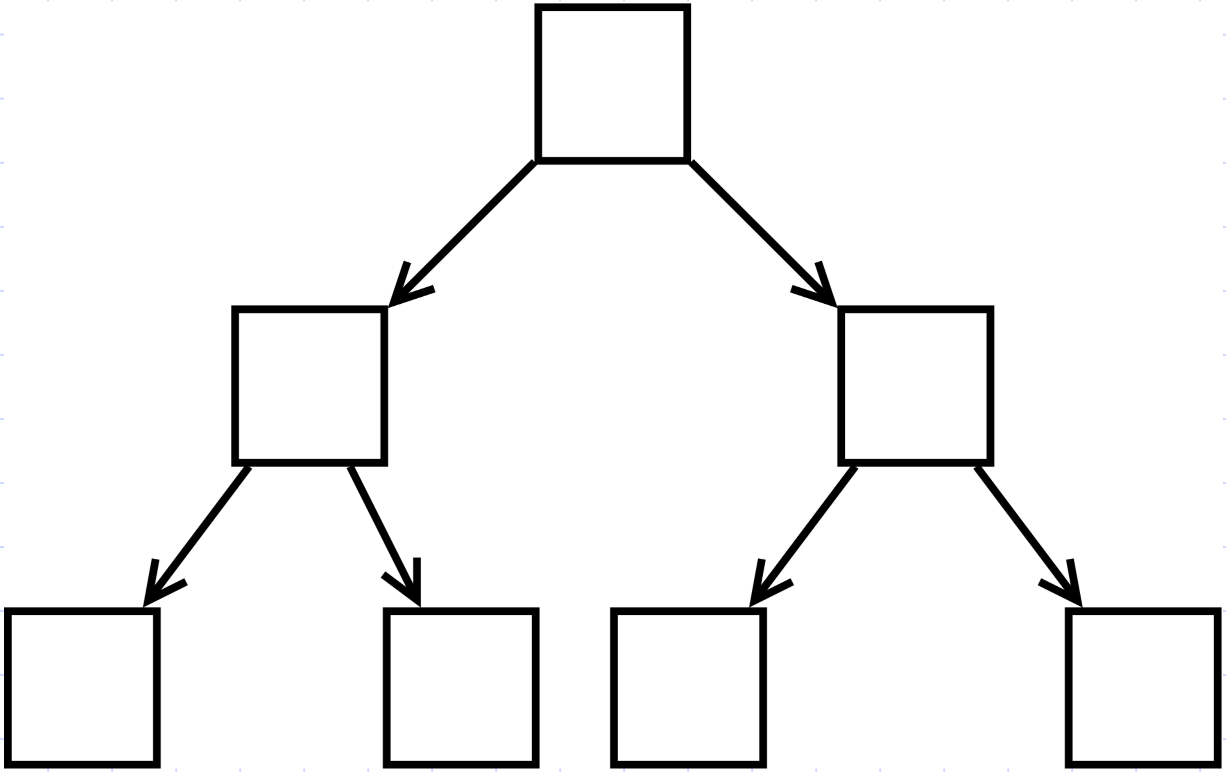
- The root node sends message to its successors and waits for response.
- An intermediate node waits for a message from its parent, sends message to its successors, waits for responses, and relays these replies to the parent.
- A leaf node waits for a message from its parent and sends a response back.

Checking Invariant





It is enough to check the model



Another models

- We are looking for interesting (and practical) models as case study for running experiments
- Now we are trying to build an abstraction of Resource ReserVation Protocol (RSVP) and check its properties.

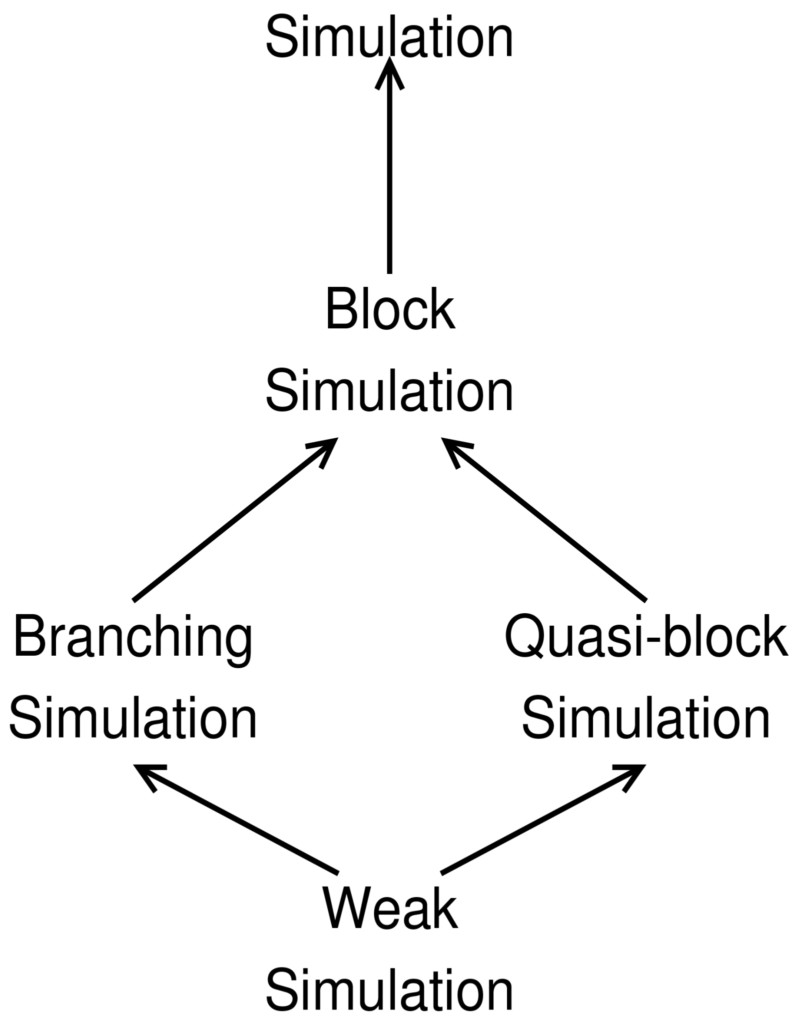
Computing Block Simulation, straightforward approach

- To check M' ≤ M'' one may:
  - begin with including all pairs (s', s''): s' in S', s'' in S'' of nodes having the same labels
  - refine the set by removing one by one those pairs that do not fit the definition
  - until only those pairs that agree the definition remain.
- Pairs may be added on demand.
- Models may be built on-the-fly.

Computing Block Simulation, game-theoretic approach

- Simulation-like relations may be interpreted as a parity game of two players: Spoiler and Duplicator [T. Henzinger, O. Kupferman, S. Rajamani, 2002].
- Spoiler tries to find a move which testifies against the simulation while Duplicator should find an adequate response to certify the simulation.
- If Duplicator provides a winning strategy, then the simulation do exists.

Hierarchy of Simulations



References

- K.R. Apt, D. Kozen. Limits for automatic program verification of finite-state concurrent systems. Information Processing Letters, 22(6), 1986, pp. 307-309.
- E.M. Clarke, O. Grumberg, and S. Jha. Verifying parameterized networks using abstraction and regular languages. Proceedings of the 6-th International Conference on Concurrency Theory, 1995.
- E.M. Clarke, O. Grumberg, and S. Jha. Verifying parameterized networks. ACM Transactions on Programming Languages and Systems, vol. 19, N 5, 1997, pp. 726—750.
- Thomas A. Henzinger, Orna Kupferman, and Sriram K. Rajamani. Fair Simulation. Information and Computation 173:64-81, 2002.