Verifying lossy channel systems

Ph. Schnoebelen

http://www.lsv-ens-cachan.fr/~phs

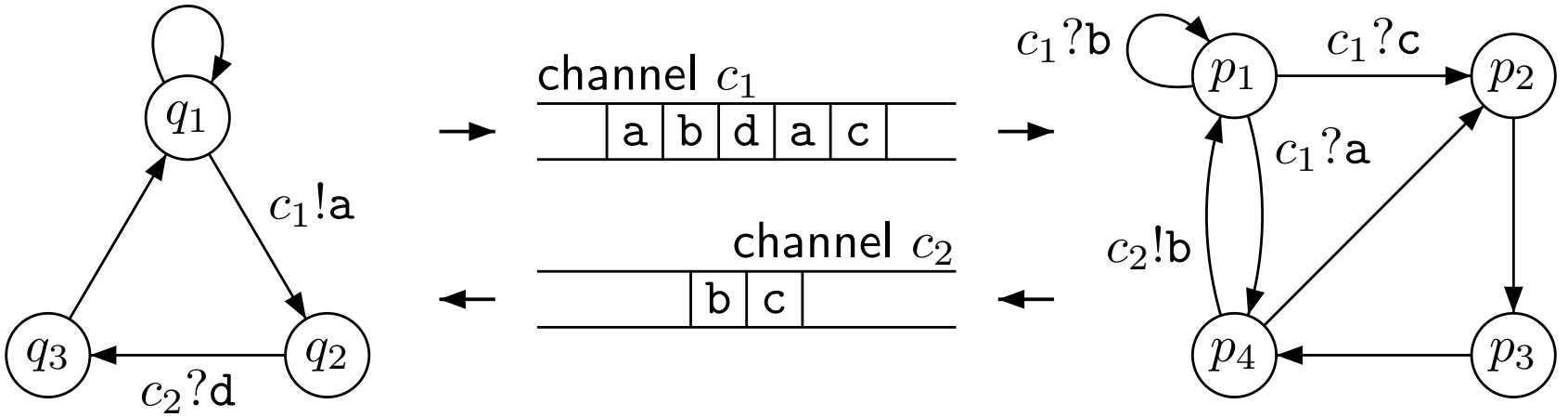
Lab. Specification & Verification (LSV)

ENS de Cachan & CNRS

Cachan, France

Channel Systems

A.k.a. "communicating finite-state machines"

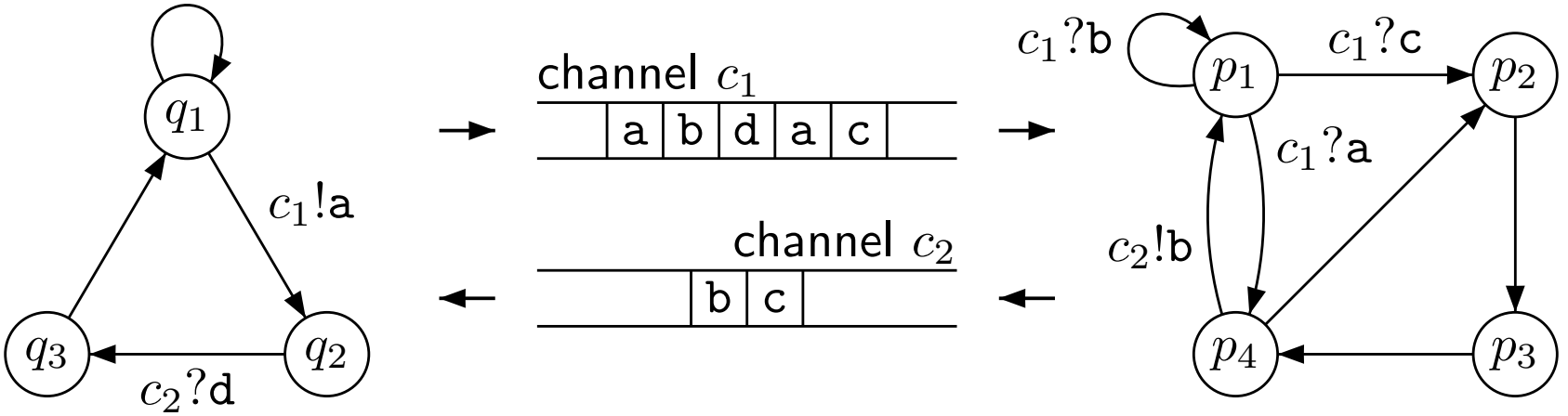


Natural model for asynchronous communication protocols: SDL, Estelle

[von Bochmann 1978; Brand & Zafiropulo 1983]

Channel Systems

A.k.a. "communicating finite-state machines"

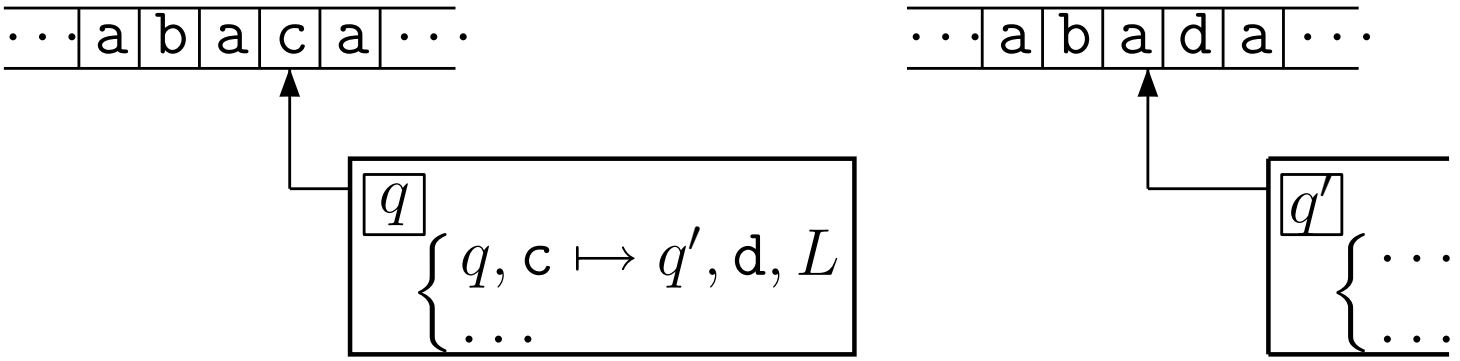


Natural model for asynchronous communication protocols: SDL, Estelle

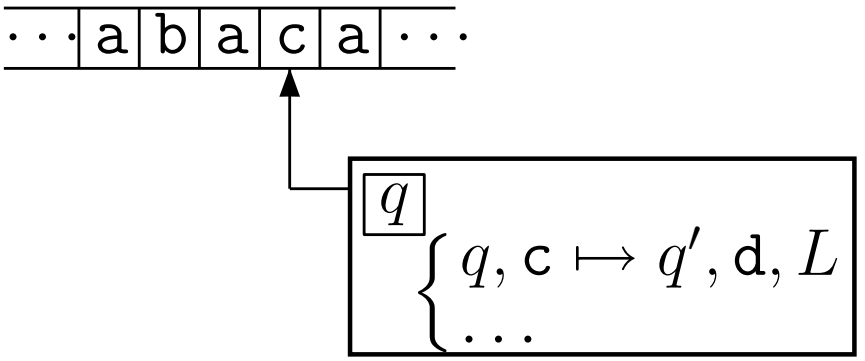
[von Bochmann 1978; Brand & Zafiropulo 1983]

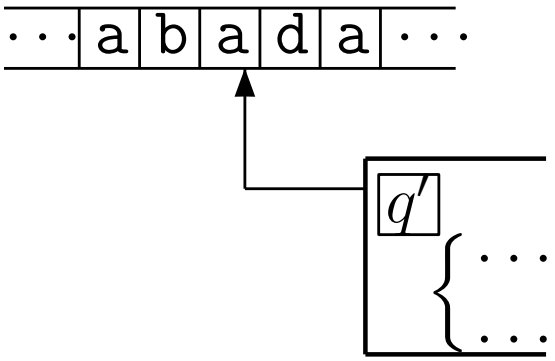
Turing powerful!

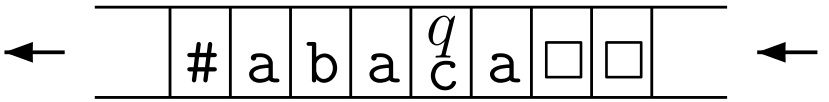
Simulating Turing Machines



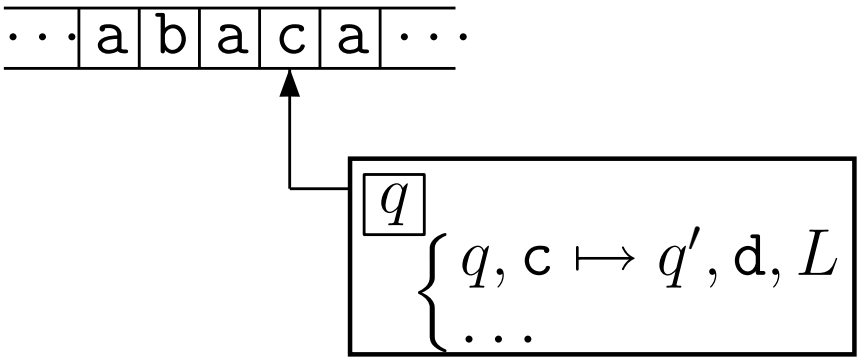
Simulating Turing Machines

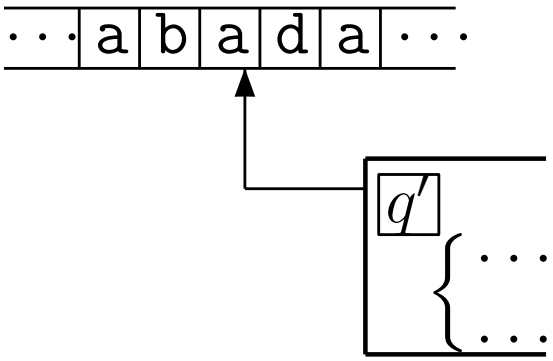


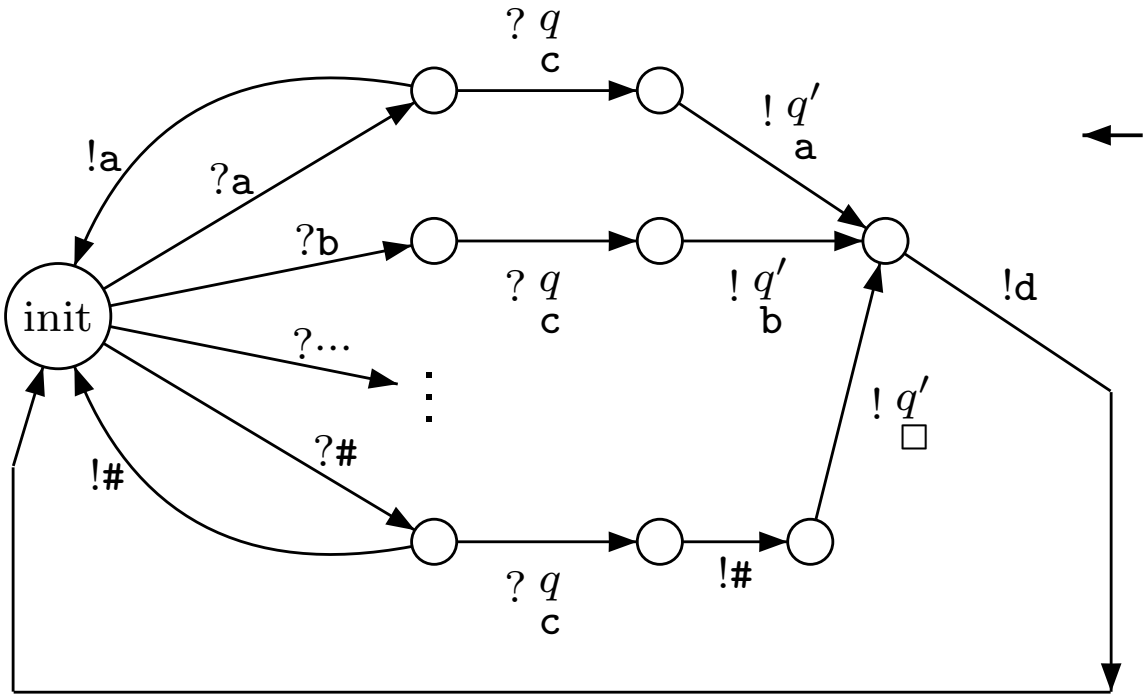


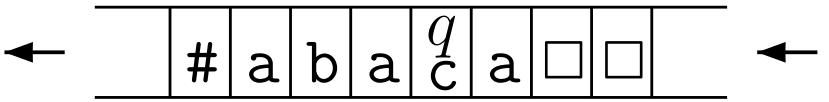


Simulating Turing Machines









Model Checking Channel Systems

Rice Theorem for Channel Systems: All nontrivial behavioural properties of channel systems are undecidable.

Hence model checking is impossible on principle grounds.

BUT

Model Checking Channel Systems

Rice Theorem for Channel Systems: All nontrivial behavioural properties of channel systems are undecidable.

Hence model checking is impossible on principle grounds.

BUT

Paradoxical finding [Finkel 94; Abdulla & Jonsson 96b]: Model checking becomes possible when channels are unreliable (can lose messages).

These lossy channel systems are well-suited to the analysis of asynchronous protocols that are designed to cope with message losses.

Model Checking Channel Systems

Rice Theorem for Channel Systems: All nontrivial behavioural properties of channel systems are undecidable.

Hence model checking is impossible on principle grounds.

BUT

Paradoxical finding [Finkel 94; Abdulla & Jonsson 96b]: Model checking becomes possible when channels are unreliable (can lose messages).

These lossy channel systems are well-suited to the analysis of asynchronous protocols that are designed to cope with message losses.

Side question: Such protocols are everywhere!! Why weren't lossy channel systems identified earlier???

Outline of the Talk

Basics and Definitions

Algorithm for Inevitability

Algorithm for Reachability

Hard Problems With a Recurring idea

Why Study Probabilistic Systems?

Channel Systems: Perfect

W.l.o.g., we only consider systems made of one component and several channels.

$S = \langle Q, \Sigma, C, \Delta \rangle$ with
– $Q = \{q, \ldots\}$, the *control states*
– $\Sigma = \{a, b, \ldots\}$, the *messages*
– $C = \{c_1, c_2, \ldots, c_n\}$, the *channels*
– $\Delta \subseteq Q \times C \times \{?, !\} \times \Sigma \times Q$, the *rules*

Rules in Δ written e.g. as "$(q, c!a, q')$"

A *configuration* of S: $\sigma = \langle q, u_1, \ldots, u_n \rangle$

Perfect steps: $\langle q, u_1, \ldots, u_n \rangle \rightarrow \langle r, v_1, \ldots, v_n \rangle$ if
— $(q, c_i?a, r)$ is a rule, $u_i = a.v_i$ and $v_j = u_j$ for $j \neq i$, or
— $(q, c_i!a, r)$ is a rule, $v_i = u_i.a$ and $v_j = u_j$ for $j \neq i$.
NB: no test for emptiness

Channel Systems: Lossy

Subword ordering: $abba \sqsubseteq abracadabra$

Subword relation for configurations: $\sigma \sqsubseteq \sigma'$

Lossy steps: $\sigma \rightarrow_{\text{lossy}} \sigma' \stackrel{\text{def}}{\Leftrightarrow} \sigma \sqsupseteq \delta \rightarrow_{\text{perf}} \delta' \sqsupseteq \sigma'$

Corollary: If $\sigma_1 \rightarrow \sigma_2$ then $\sigma_1' \rightarrow \sigma_2'$ for any $\sigma_1' \sqsupseteq \sigma_1$ and $\sigma_2' \sqsubseteq \sigma_2$.

Channel Systems: Lossy

Subword ordering: $abba \sqsubseteq abracadabra$

Subword relation for configurations: $\sigma \sqsubseteq \sigma'$

Lossy steps: $\sigma \rightarrow_{\text{lossy}} \sigma' \overset{\text{def}}{\Leftrightarrow} \sigma \sqsupseteq \delta \rightarrow_{\text{perf}} \delta' \sqsupseteq \sigma'$

Corollary: If $\sigma_1 \rightarrow \sigma_2$ then $\sigma_1' \rightarrow \sigma_2'$ for any $\sigma_1' \sqsupseteq \sigma_1$ and $\sigma_2' \sqsubseteq \sigma_2$.

Lemma [Higman 1952]: the subword ordering is a well-quasi-order (wqo), i.e. any infinite sequence $u_0, u_1, u_2, \ldots$ of words has an infinite increasing subsequence $u_{i_0} \sqsubseteq u_{i_1} \sqsubseteq u_{i_2} \sqsubseteq \cdots$

$\Rightarrow$ Applies equivalently to configurations of S ordered by $\sqsubseteq$.

Corollary. Any set of configurations has a finite number of minimal elements.

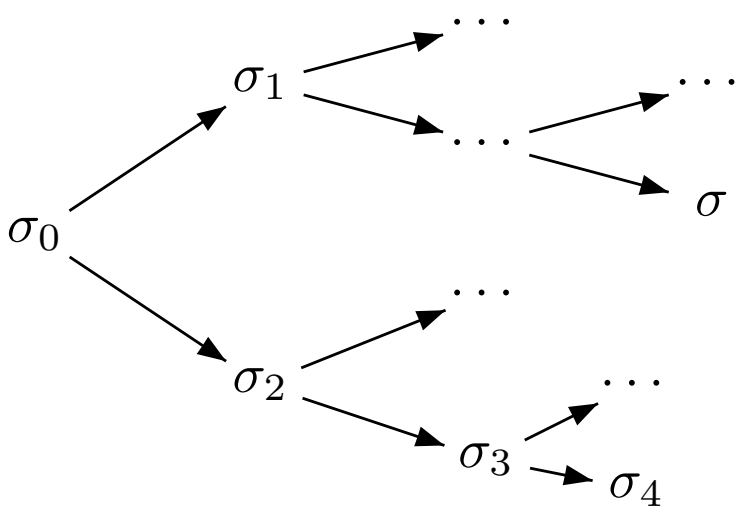
Termination Is Decidable [Finkel 1994]

Problem is to decide whether all runs (starting from a given initial configuration σ_0) are finite.

Termination Is Decidable [Finkel 1994]

Problem is to decide whether all runs (starting from a given initial configuration σ_0) are finite.

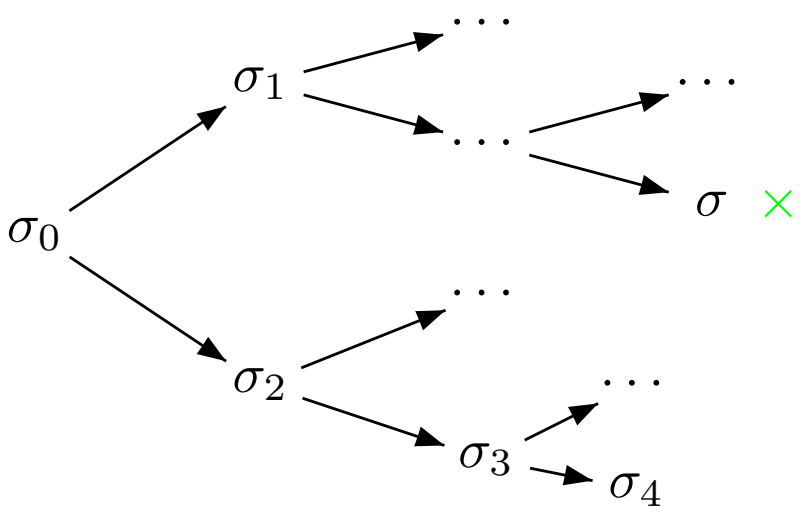
Algorithm is to explore all possible runs!



Termination Is Decidable [Finkel 1994]

Problem is to decide whether all runs (starting from a given initial configuration σ_0) are finite.

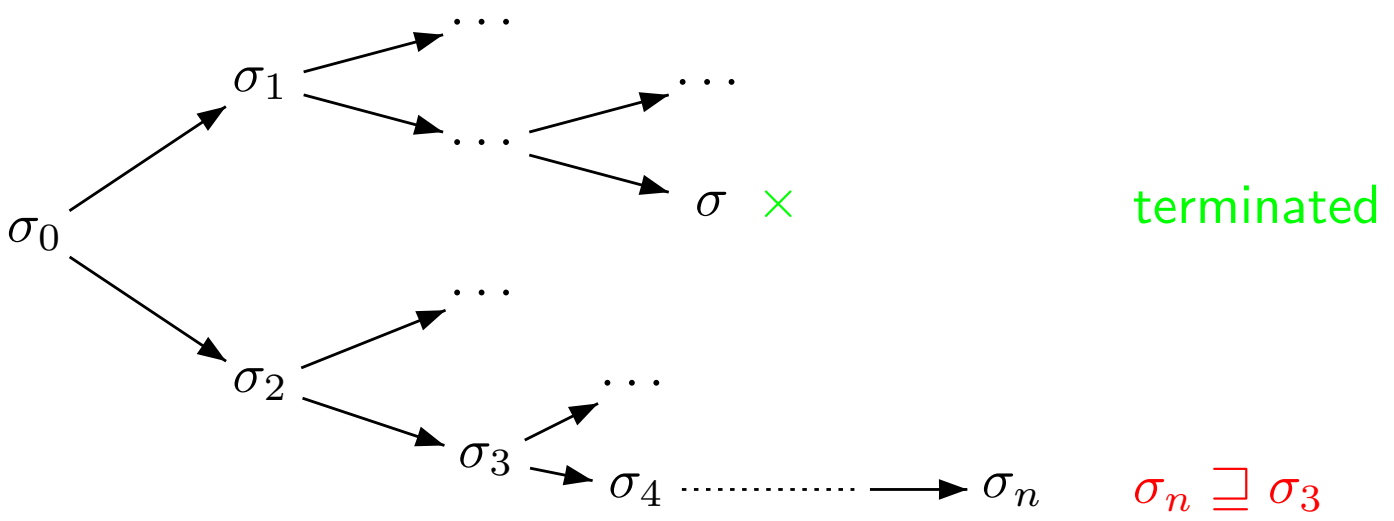
Algorithm is to explore all possible runs!



Termination Is Decidable [Finkel 1994]

Problem is to decide whether all runs (starting from a given initial configuration σ_0) are finite.

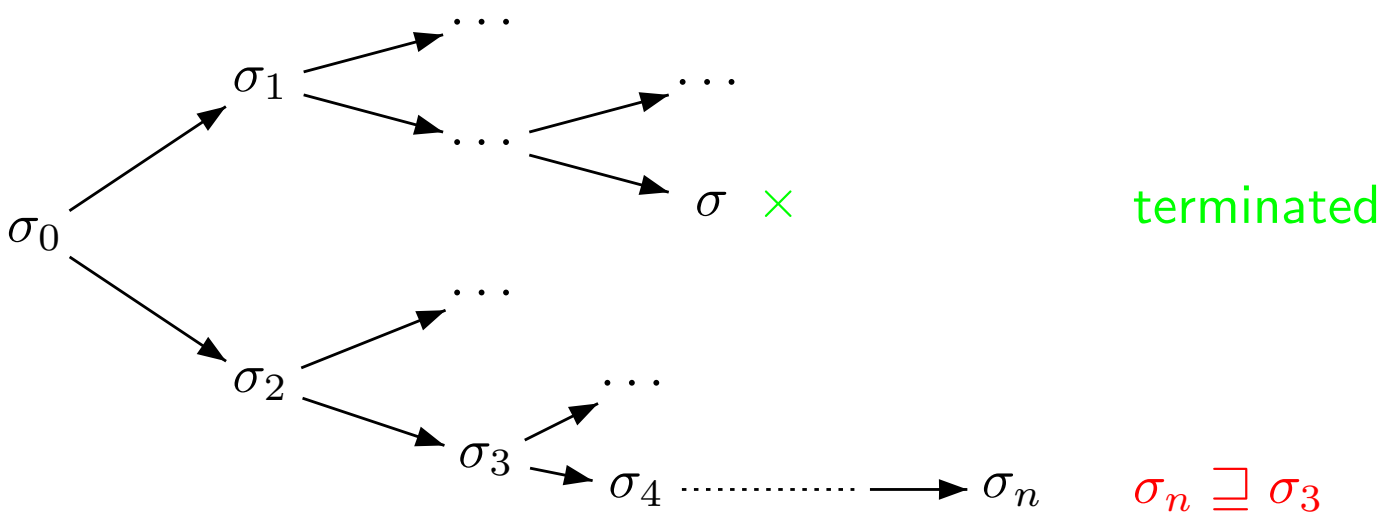
Algorithm is to explore all possible runs!



Termination Is Decidable [Finkel 1994]

Problem is to decide whether all runs (starting from a given initial configuration σ_0) are finite.

Algorithm is to explore all possible runs!



$\Rightarrow$ $\sigma_n \rightarrow \sigma_4$

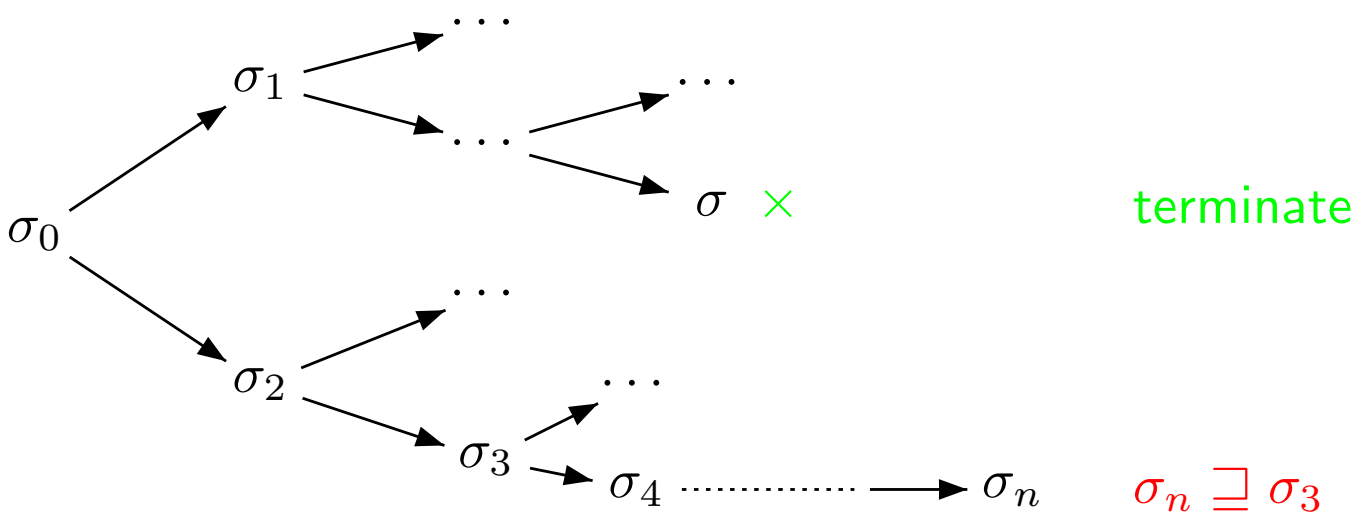
$\Rightarrow$ looping is possible

$\Rightarrow$ infinite runs from σ_0 exist

Termination Is Decidable [Finkel 1994]

Problem is to decide whether all runs (starting from a given initial configuration σ_0) are finite.

Algorithm is to explore all possible runs!



$\Rightarrow$ $\sigma_n \rightarrow \sigma_4$
$\Rightarrow$ looping is possible
$\Rightarrow$ infinite runs from σ_0 exist

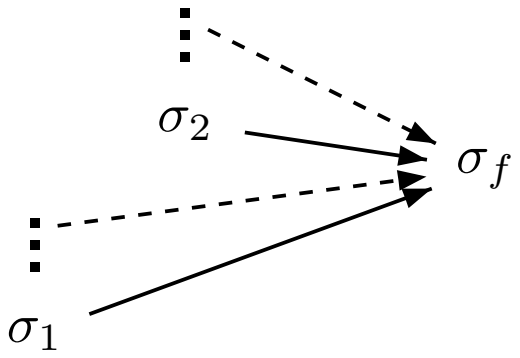
By Higman's & König's Lemmas, the algorithm must eventually conclude.
(NB: works more generally for $\mathsf{A}\varphi\mathsf{U}\psi$ properties.)

Reachability Is Decidable [AJ96b]

Problem is to decide whether $\sigma_0 \xrightarrow{*} \sigma_f$.
Algorithm is backward search + pruning:

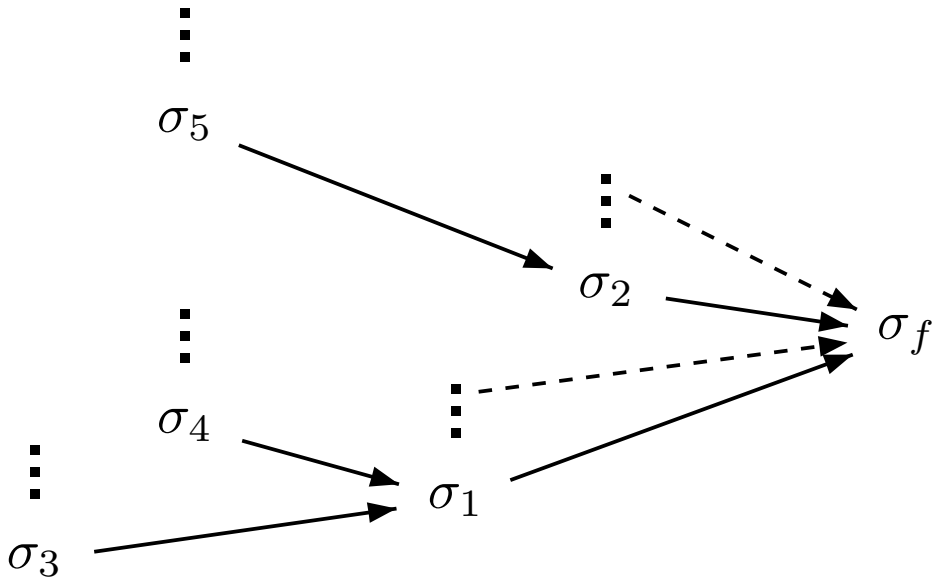
Reachability Is Decidable [AJ96b]

Problem is to decide whether $\sigma_0 \xrightarrow{*} \sigma_f$.
Algorithm is backward search + pruning:



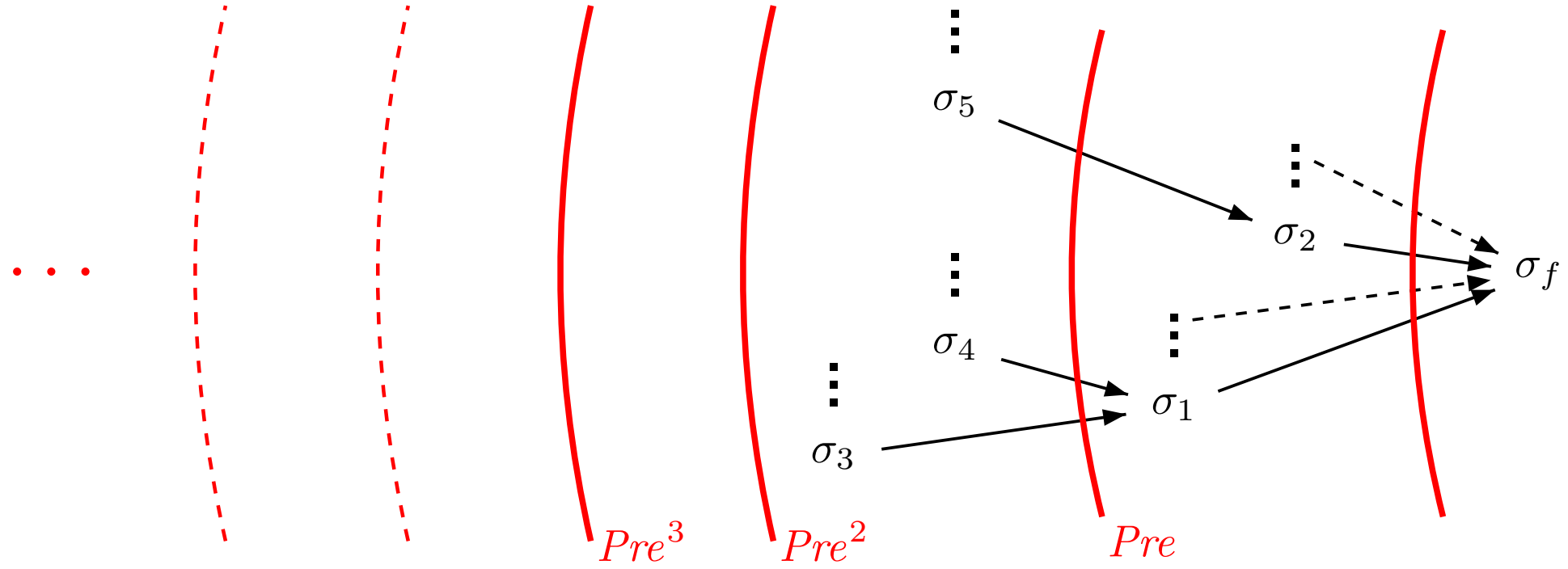
Reachability Is Decidable [AJ96b]

Problem is to decide whether $\sigma_0 \xrightarrow{*} \sigma_f$.
Algorithm is backward search + pruning:



Reachability Is Decidable [AJ96b]

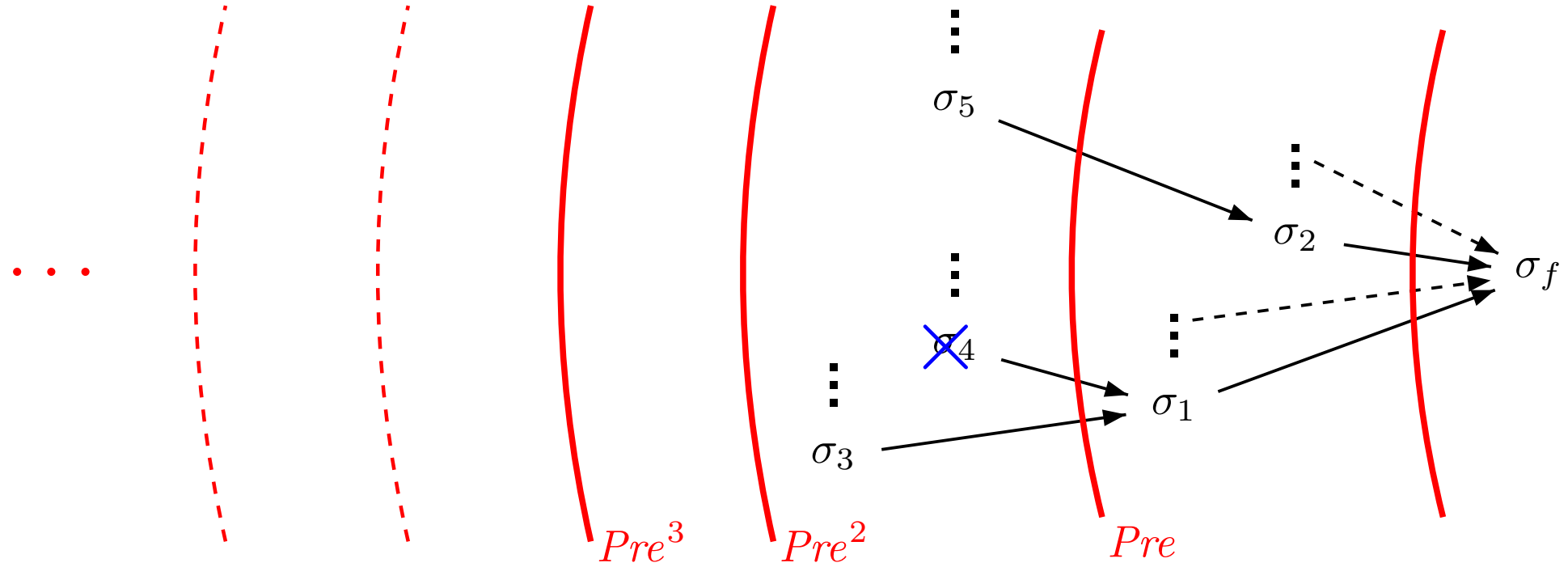
Problem is to decide whether $\sigma_0 \xrightarrow{*} \sigma_f$.
Algorithm is backward search + pruning:



$\lim_i Pre^i = Pre^*$

Reachability Is Decidable [AJ96b]

Problem is to decide whether $\sigma_0 \xrightarrow{*} \sigma_f$.
Algorithm is backward search + pruning:

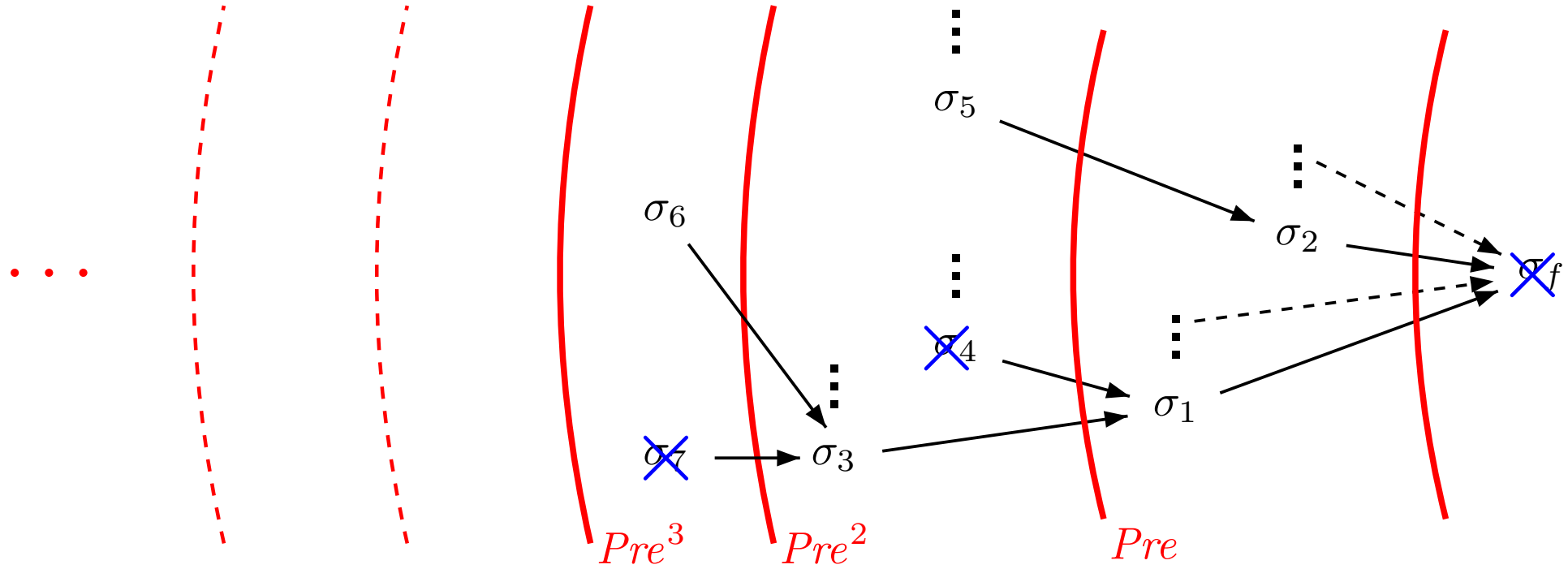


$\lim_i Pre^i = Pre^*$

assuming $\sigma_2 \sqsubseteq \sigma_4$ (say)

Reachability Is Decidable [AJ96b]

Problem is to decide whether $\sigma_0 \xrightarrow{*} \sigma_f$.
Algorithm is backward search + pruning:

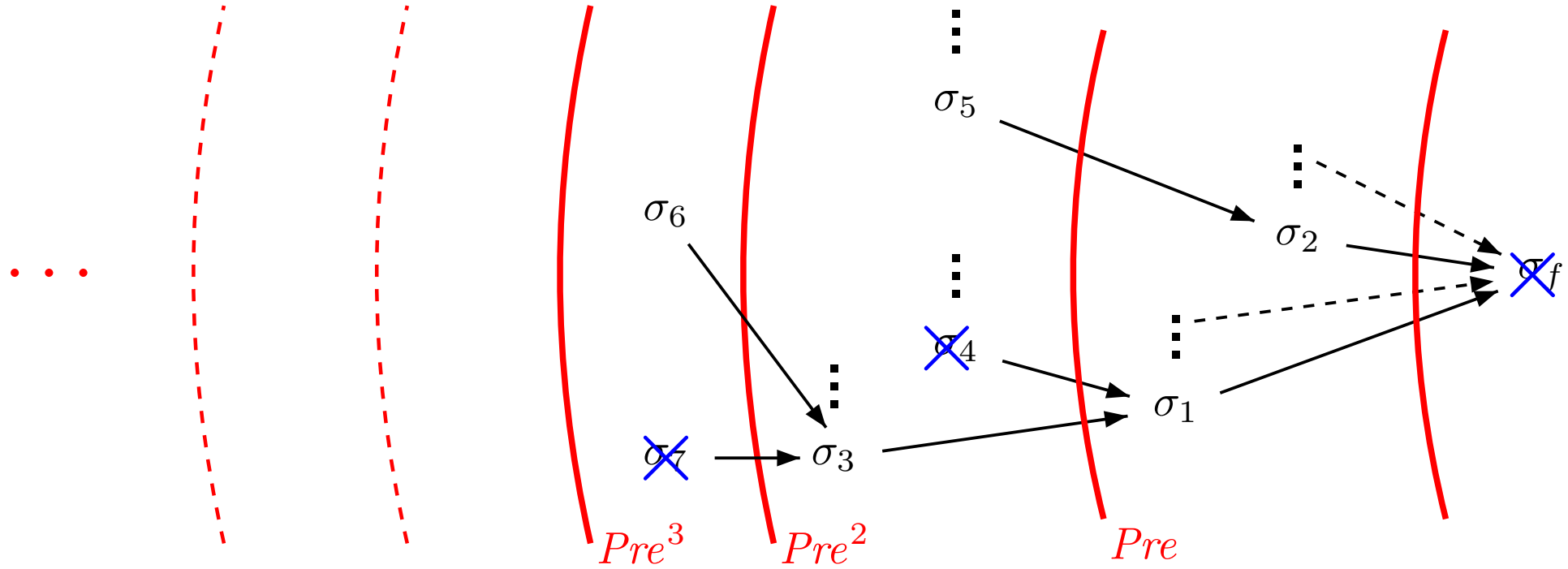


$\lim_i Pre^i = Pre^*$

assuming $\sigma_2 \sqsubseteq \sigma_4$ (say)

Reachability Is Decidable [AJ96b]

Problem is to decide whether $\sigma_0 \xrightarrow{*} \sigma_f$.
Algorithm is backward search + pruning:

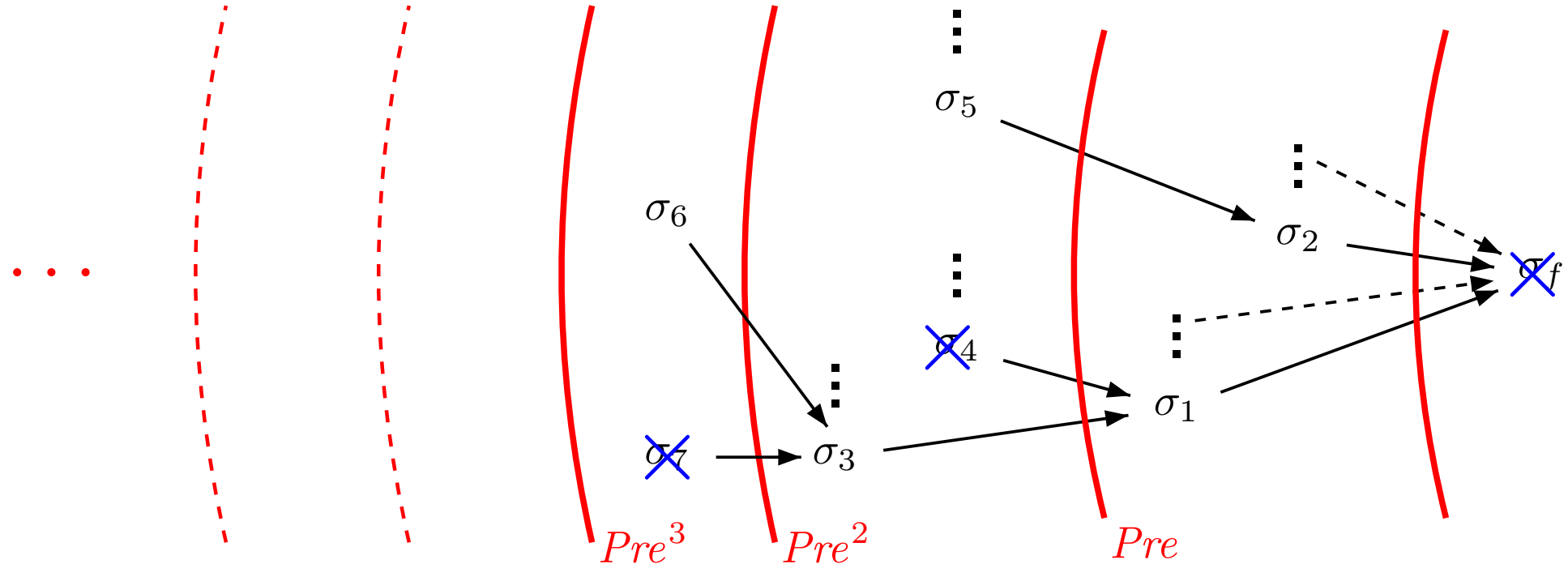


$\lim_i Pre^i = Pre^*$ assuming $\sigma_2 \sqsubseteq \sigma_4$ (say)

At step k we know the minimal elements of Pre^k.

Reachability Is Decidable [AJ96b]

Problem is to decide whether $\sigma_0 \xrightarrow{*} \sigma_f$.
Algorithm is backward search + pruning:



$\lim_i Pre^i = Pre^*$ assuming $\sigma_2 \sqsubseteq \sigma_4$ (say)

At step k we know the minimal elements of Pre^k.
Higman's Lemma ensures $Pre^k = Pre^{k+1}$ for some k.
$\Rightarrow Pre^*$ is effectively computable.

Other Decidable Problems?

Safety properties are decidable.

Simulation and bisimulation with a finite state system are decidable.

Other Decidable Problems?

Safety properties are decidable.

Simulation and bisimulation with a finite state system are decidable.

Question: Are lossy channel systems a *trivial* model?

Other Decidable Problems?

Safety properties are decidable.

Simulation and bisimulation with a finite state system are decidable.

Question: Are lossy channel systems a *trivial* model?

Answer: No! Many problems are undecidable, all other are hard.

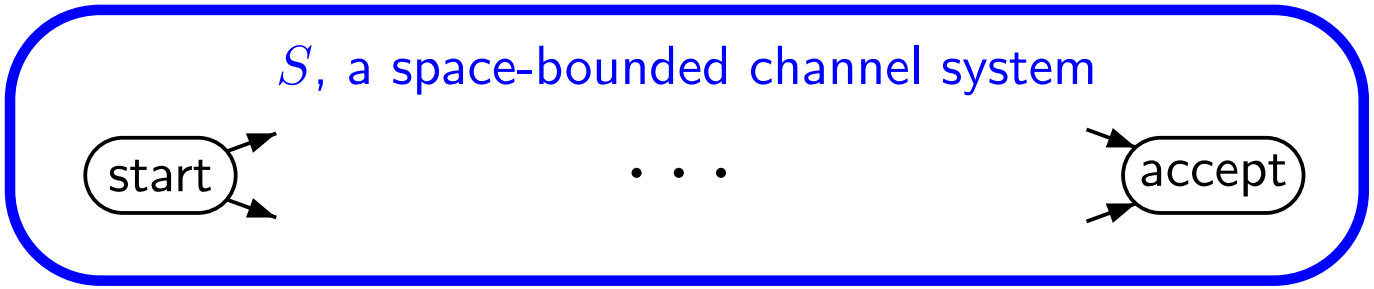
We illustrate this in a uniform way...

Recurrent Reachability Is Undecidable [AJ96a]

Problem is to decide whether there is a run from some σ_0 that visits some control state r infinitely often.

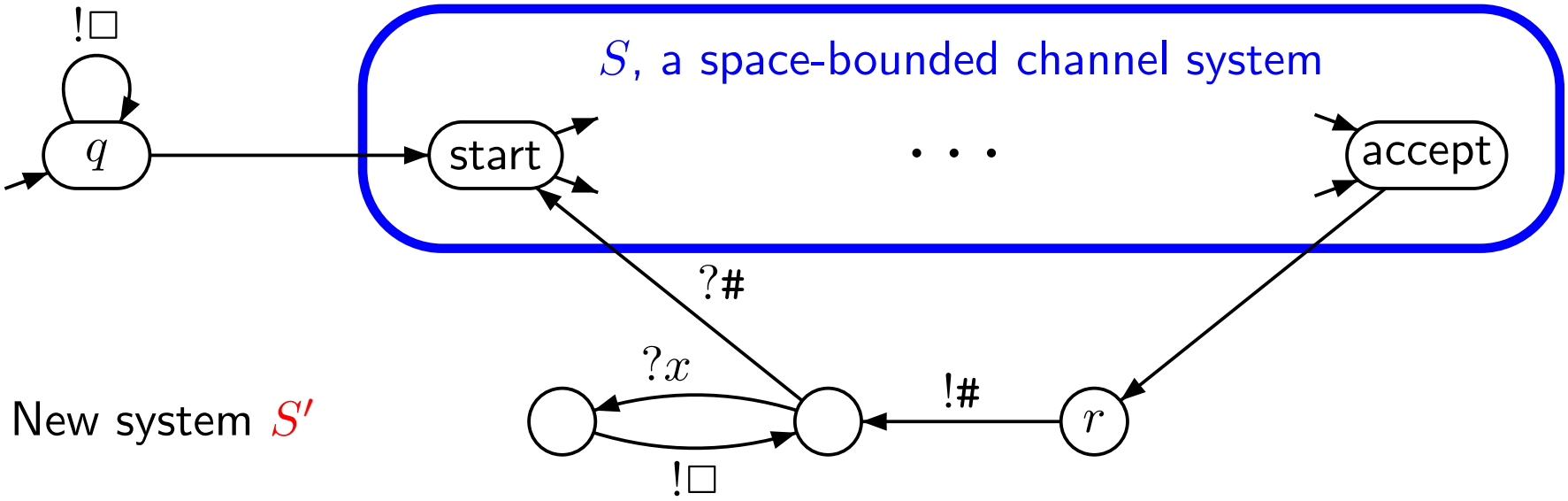
Recurrent Reachability Is Undecidable [AJ96a]

Problem is to decide whether there is a run from some σ_0 that visits some control state r infinitely often.



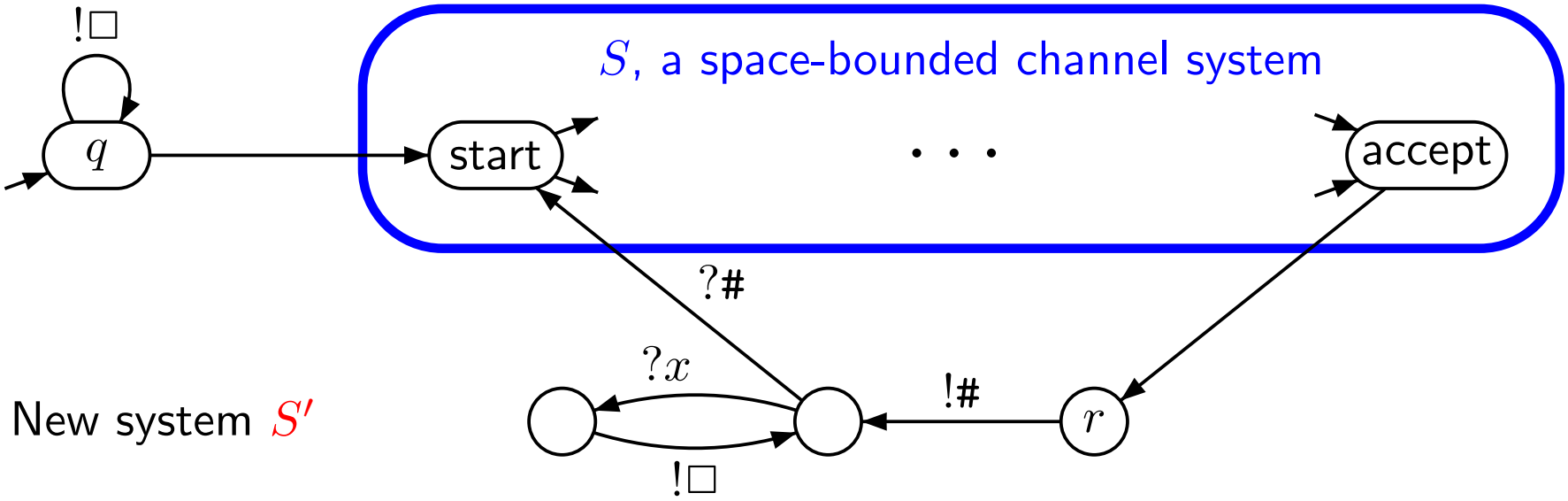
Recurrent Reachability Is Undecidable [AJ96a]

Problem is to decide whether there is a run from some σ_0 that visits some control state r infinitely often.



Recurrent Reachability Is Undecidable [AJ96a]

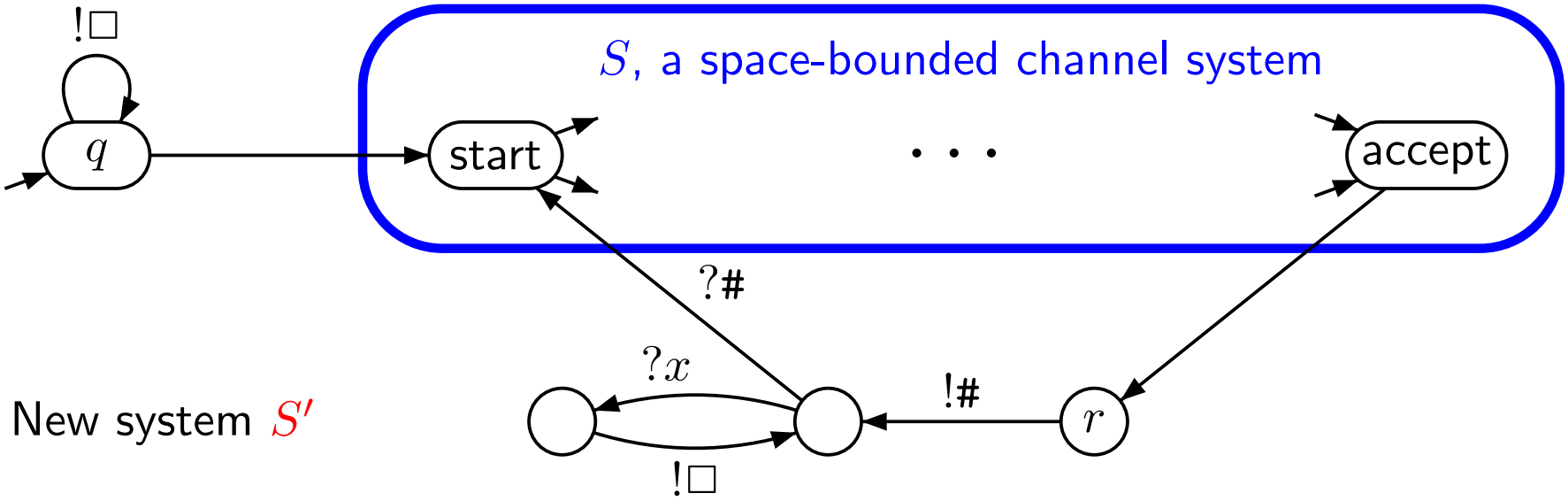
Problem is to decide whether there is a run from some σ_0 that visits some control state r infinitely often.



Observe: lossy S' can visit r infinitely often **iff** perfect S accepts (given enough memory).

Recurrent Reachability Is Undecidable [AJ96a]

Problem is to decide whether there is a run from some σ_0 that visits some control state r infinitely often.



Observe: lossy S' can visit r infinitely often **iff** perfect S accepts (given enough memory).

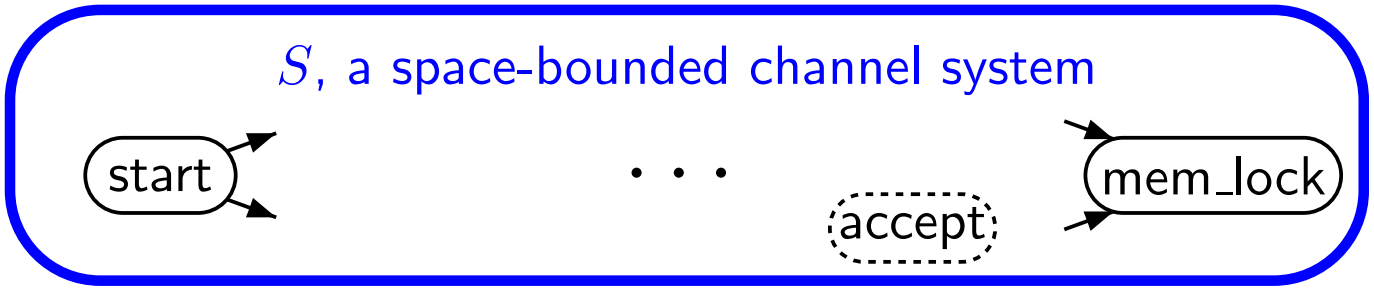
Corollary: Liveness properties of lossy channel systems are undecidable.

Boundedness Is Undecidable [DJS99; May03]

Problem is to decide whether channel contents can become arbitrarily large.

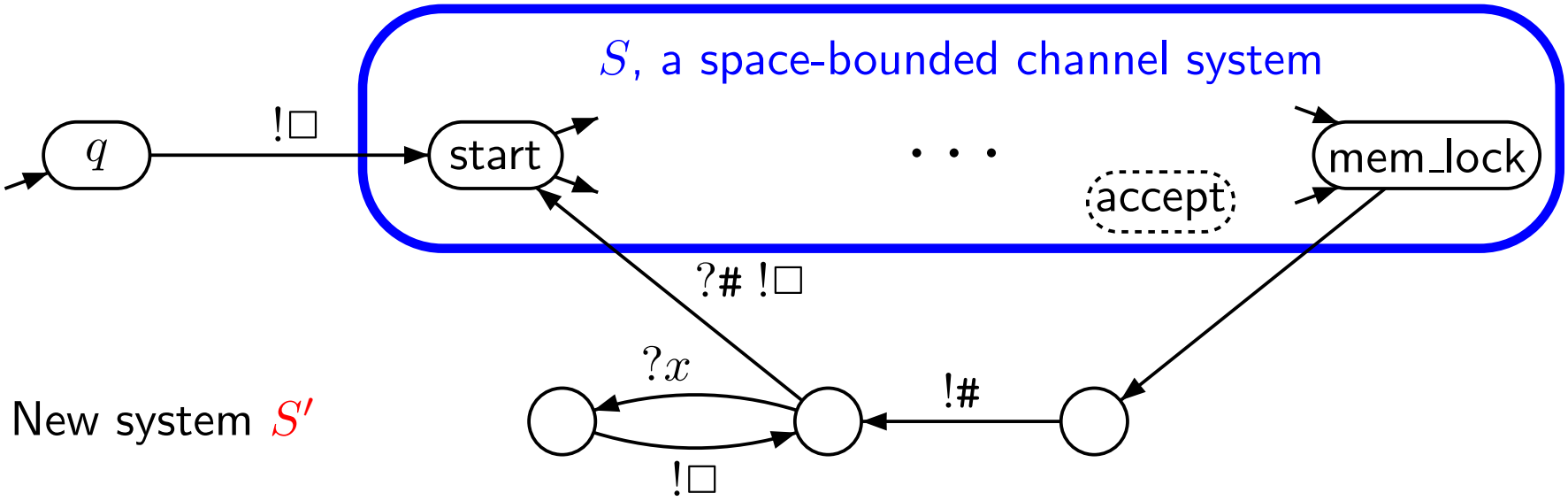
Boundedness Is Undecidable [DJS99; May03]

Problem is to decide whether channel contents can become arbitrarily large.



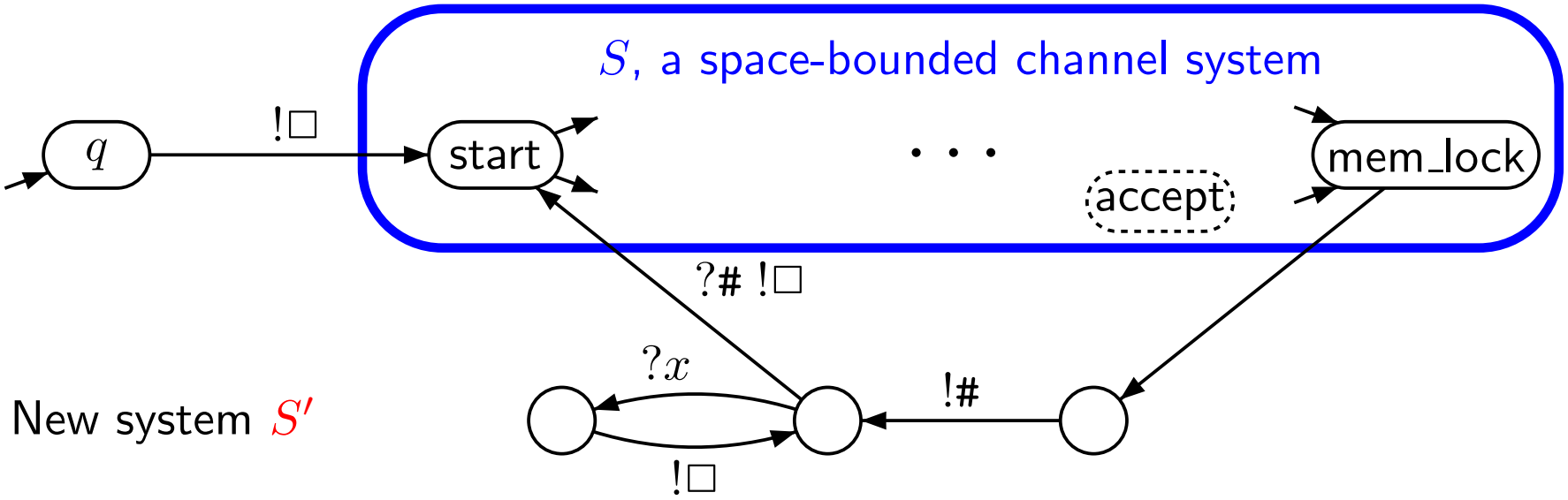
Boundedness Is Undecidable [DJS99; May03]

Problem is to decide whether channel contents can become arbitrarily large.



Boundedness Is Undecidable [DJS99; May03]

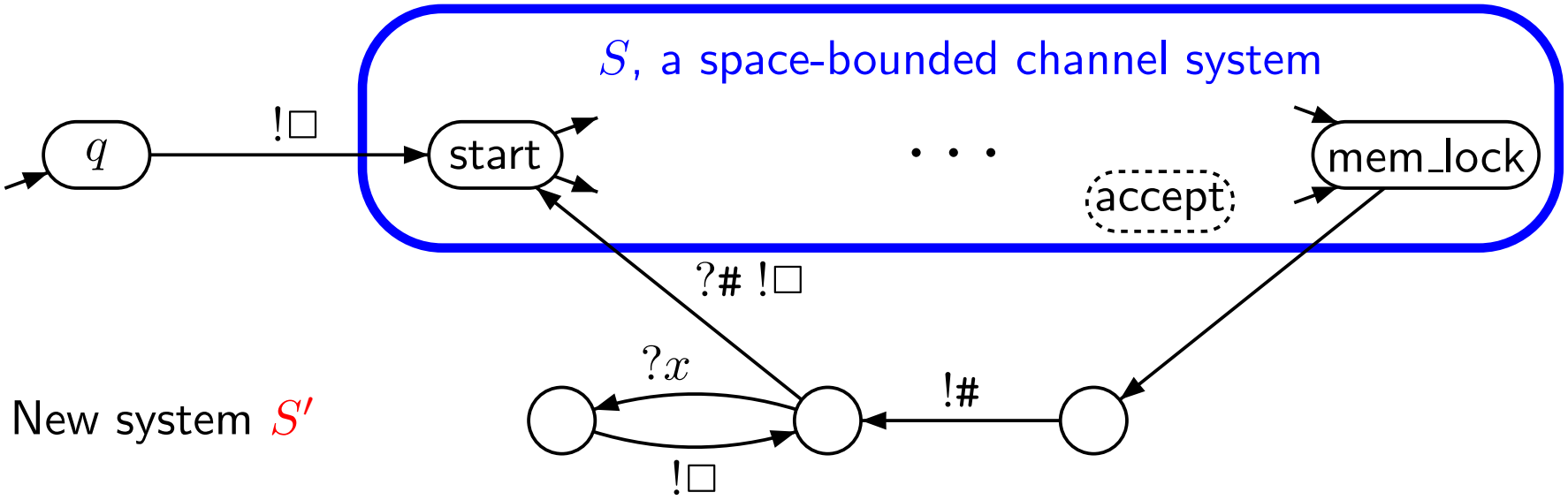
Problem is to decide whether channel contents can become arbitrarily large.



Observe: lossy S' is unbounded **iff** perfect S cannot be given enough memory.

Boundedness Is Undecidable [DJS99; May03]

Problem is to decide whether channel contents can become arbitrarily large.



Observe: lossy S' is unbounded **iff** perfect S cannot be given enough memory.

Corollary: $Post^*$ is not effectively computable (though it is finitely recognizable).

Some Other Results

Theorem (Cécé): Reachability is decidable for ring networks with one lossy channel.

Open Problem: Characterize network topologies that provide decidability results.

Some Other Results

Theorem (Cécé): Reachability is decidable for ring networks with one lossy channel.

Open Problem: Characterize network topologies that provide decidability results.

Theorem (S. 2001): All behavioral equivalences are undecidable between lossy channel systems.

Some Other Results

Theorem (Cécé): Reachability is decidable for ring networks with one lossy channel.

Open Problem: Characterize network topologies that provide decidability results.

Theorem (S. 2001): All behavioral equivalences are undecidable between lossy channel systems.

Theorem (Masson & S. 2002): Termination under fair scheduling is decidable (for systems without multiplexing).

Inevitability under fair scheduling is undecidable.

Verification Is Hard

Let f_k be the usual k-nested primitive recursive map:

$$f_2(n) \stackrel{\text{def}}{=} 2n$$

$$f_k(n) \stackrel{\text{def}}{=} \underbrace{f_{k-1} \circ \cdots \circ f_{k-1}}_{n \text{ times}}(1) \qquad \text{if } k > 2.$$

Thus $Ack(n) \stackrel{\text{def}}{=} f_n(n)$ is Ackermann's function.

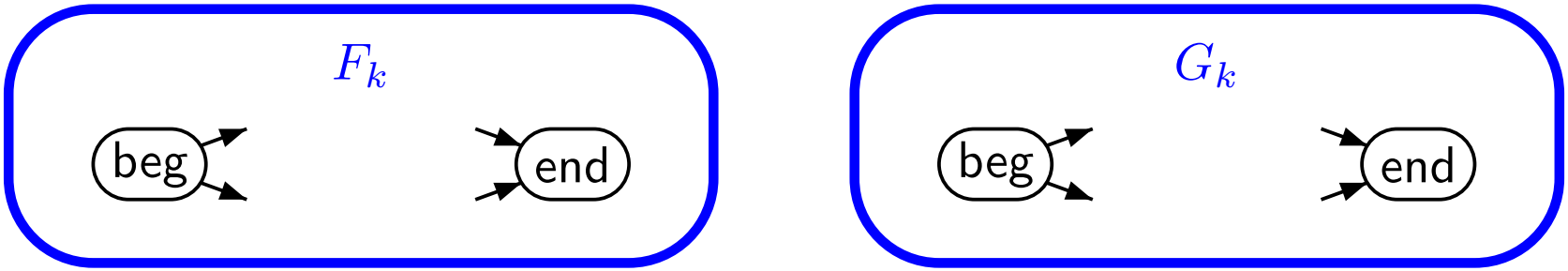
Verification Is Hard

Let f_k be the usual k-nested primitive recursive map:

$$f_2(n) \stackrel{\text{def}}{=} 2n$$

$$f_k(n) \stackrel{\text{def}}{=} \underbrace{f_{k-1} \circ \cdots \circ f_{k-1}}_{n \text{ times}}(1) \qquad \text{if } k > 2.$$

Thus $Ack(n) \stackrel{\text{def}}{=} f_n(n)$ is Ackermann's function.

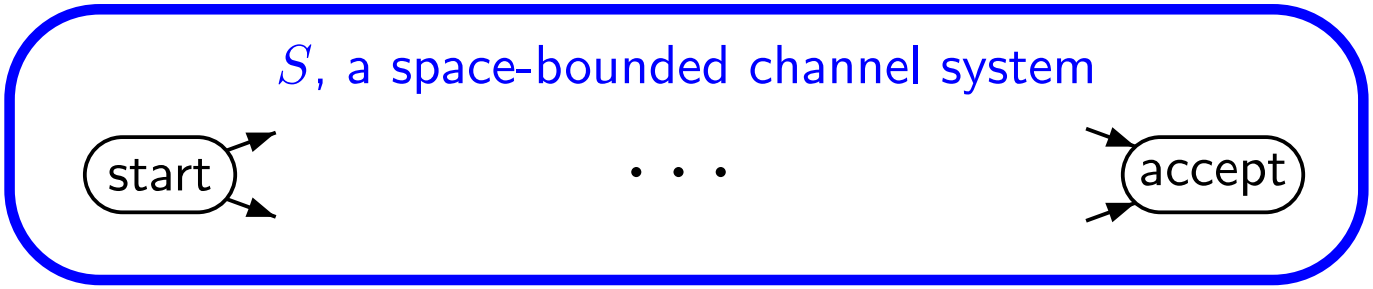


Facts: F_n and G_k have size $O(k)$.

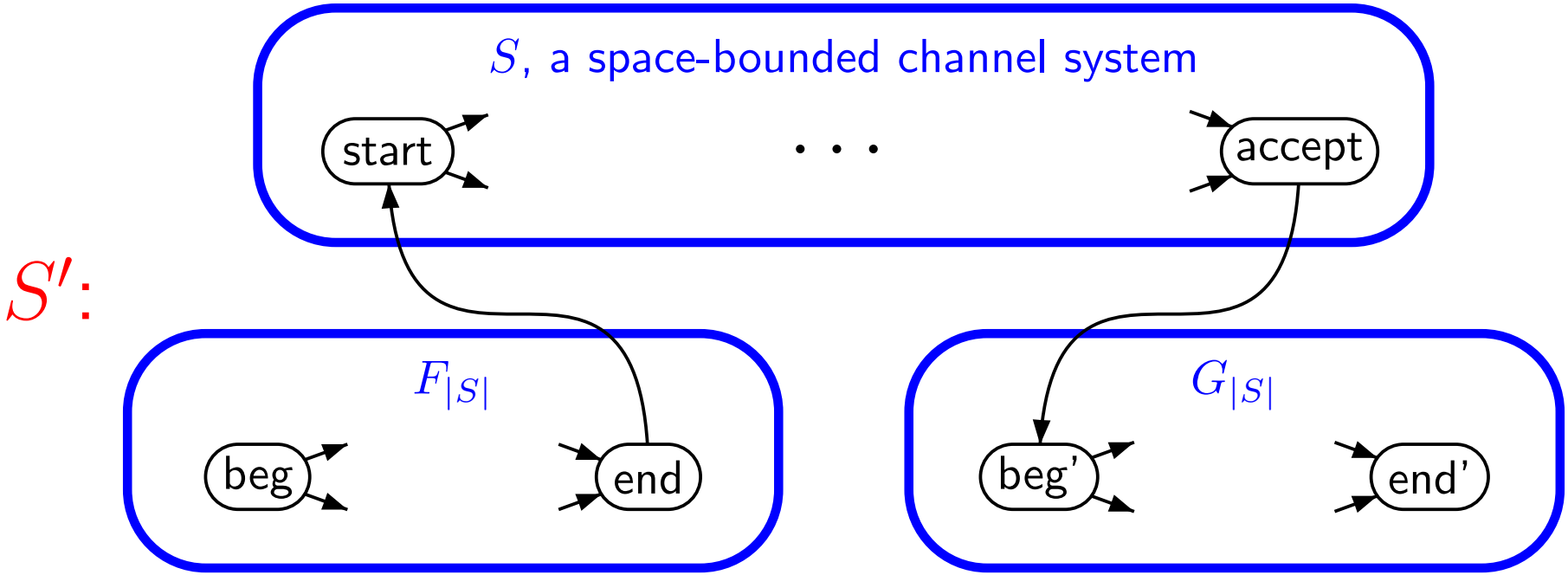
In F_k, $\langle \text{beg}, \text{a}\Box^n\text{b} \rangle \xrightarrow{*} \langle \text{end}, \text{a}\Box^m\text{b} \rangle$ **iff** $m \leq f_k(n)$.

In G_k, $\langle \text{beg}, \text{a}\Box^n\text{b} \rangle \xrightarrow{*} \langle \text{end}, \text{a}\Box^m\text{b} \rangle$ **iff** $f_k(m) \leq n$.

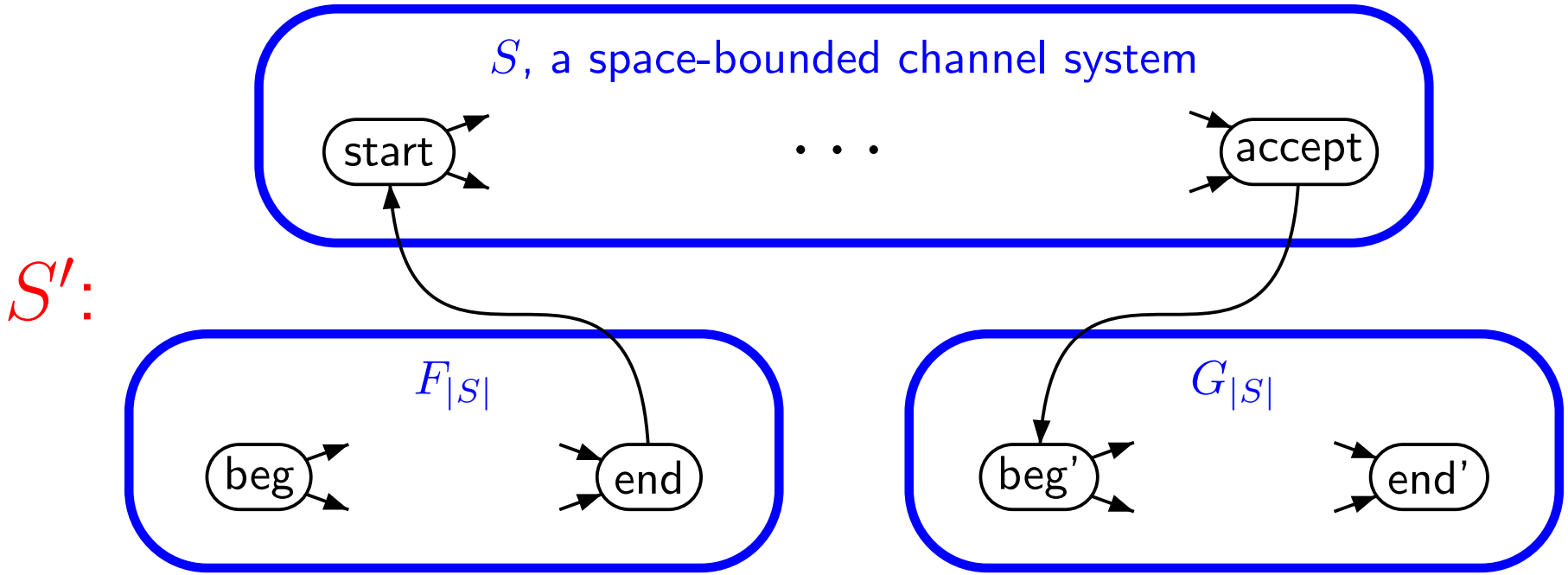
Verification Is Hard – cont'd



Verification Is Hard – cont'd



Verification Is Hard – cont'd



Observe: $\langle \text{beg}, \text{a}\Box^{|S|}\text{b} \rangle \xrightarrow{*} \langle \text{end'}, \text{a}\Box^{|S|}\text{b} \rangle$ in lossy S' **iff** perfect S accepts in $Ack(|S|)$ space.

Corollary (S. 2002): Reachability in lossy channel systems is not primitive recursive.

(NB: Applies to all other known decidable problems.)

Probabilistic Lossy Channel Systems

Definition: A Probabilistic LCS is a LCS equipped with
– positive weights on rules, and
– a constant probability $p_{\text{loss}} \in (0, 1)$.

Semantics in form of a countable Markov chain.

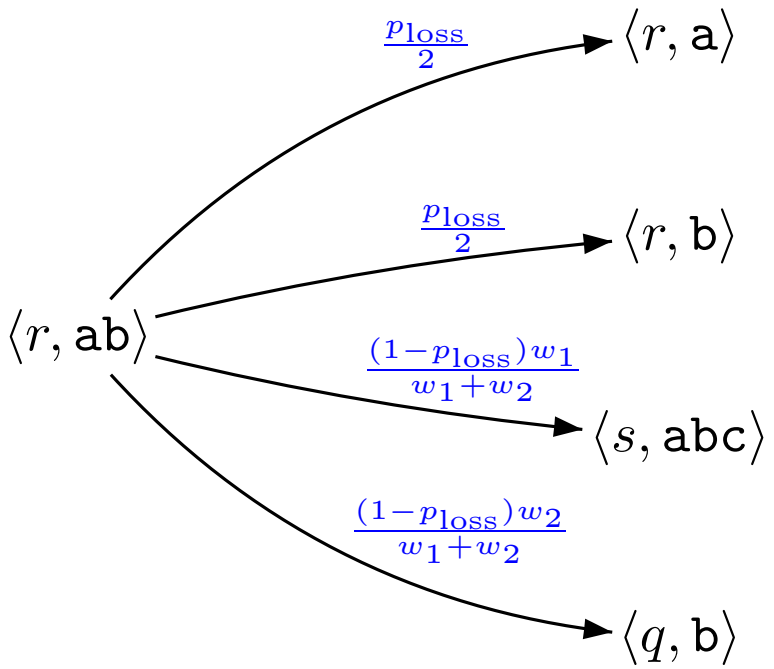
Two interpretations of p_{loss}: global-fault vs. local-fault model.

Probabilistic Lossy Channel Systems

Definition: A Probabilistic LCS is a LCS equipped with
– positive weights on rules, and
– a constant probability $p_{\text{loss}} \in (0, 1)$.

Semantics in form of a countable Markov chain.

Two interpretations of p_{loss}: global-fault vs. local-fault model.

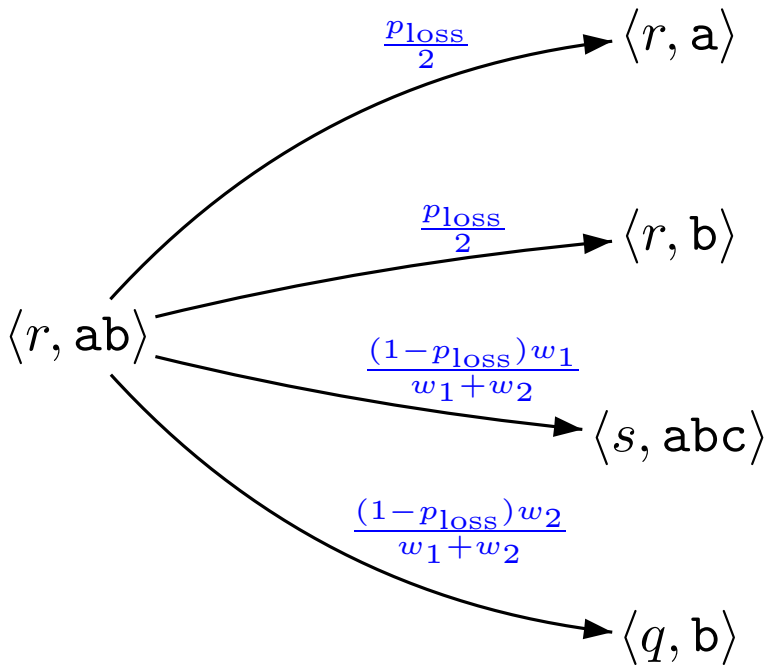


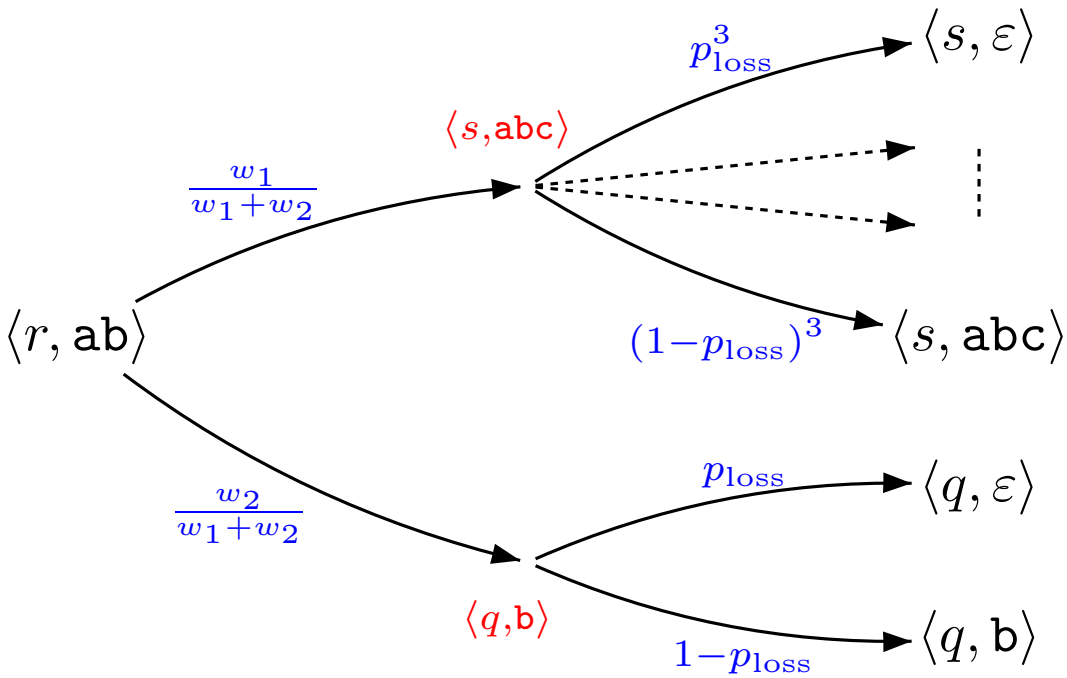
Probabilistic Lossy Channel Systems

Definition: A Probabilistic LCS is a LCS equipped with
– positive weights on rules, and
– a constant probability $p_{\text{loss}} \in (0, 1)$.

Semantics in form of a countable Markov chain.

Two interpretations of p_{loss}: global-fault vs. local-fault model.





Why Study Probabilistic Systems?

More realist than just non-deterministic losses (protocols are designed with the idea that losses are not that likely).

Randomization rules out unrealistically nasty behaviors

Why Study Probabilistic Systems?

More realist than just non-deterministic losses (protocols are designed with the idea that losses are not that likely).

Randomization rules out unrealistically nasty behaviors

Theorem (Baier & Engelen 1999): Decidability of $\mathbb{P}(\varphi) = 1$ in global-fault model when $p_{\text{loss}} \geq \frac{1}{2}$.

Why Study Probabilistic Systems?

More realist than just non-deterministic losses (protocols are designed with the idea that losses are not that likely).

Randomization rules out unrealistically nasty behaviors

Theorem (Baier & Engelen 1999): Decidability of $\mathbb{P}(\varphi) = 1$ in global-fault model when $p_{\text{loss}} \geq \frac{1}{2}$.

Theorem (Bertrand & S. 2003; Abdulla & Rabinovich 2003): Decidability of $\mathbb{P}(\varphi) = 1$ in local-fault model whatever $p_{\text{loss}} \in (0, 1)$.

All these positive results rely on finite attractors.

Why Study Probabilistic Systems?

More realist than just non-deterministic losses (protocols are designed with the idea that losses are not that likely).

Randomization rules out unrealistically nasty behaviors

Theorem (Baier & Engelen 1999): Decidability of $\mathbb{P}(\varphi) = 1$ in global-fault model when $p_{\text{loss}} \geq \frac{1}{2}$.

Theorem (Bertrand & S. 2003; Abdulla & Rabinovich 2003): Decidability of $\mathbb{P}(\varphi) = 1$ in local-fault model whatever $p_{\text{loss}} \in (0, 1)$.

All these positive results rely on finite attractors.

Theorem (Rabinovich 2003): Approximability of $\mathbb{P}(\varphi)$ when a finite attractor exists.

Randomization helps!!

Beyond Markov Chains

The problem with PLCS's is that you have to view rules as probabilistic instead of nondeterministic.

Classically, nondeterminism in rules comes from:

– arbitrary interleaving of asynchronous components

– abstraction of real-life programs

– open systems

– early designs

Beyond Markov Chains

The problem with PLCS's is that you have to view rules as probabilistic instead of nondeterministic.

Classically, nondeterminism in rules comes from:

– arbitrary interleaving of asynchronous components

– abstraction of real-life programs

– open systems

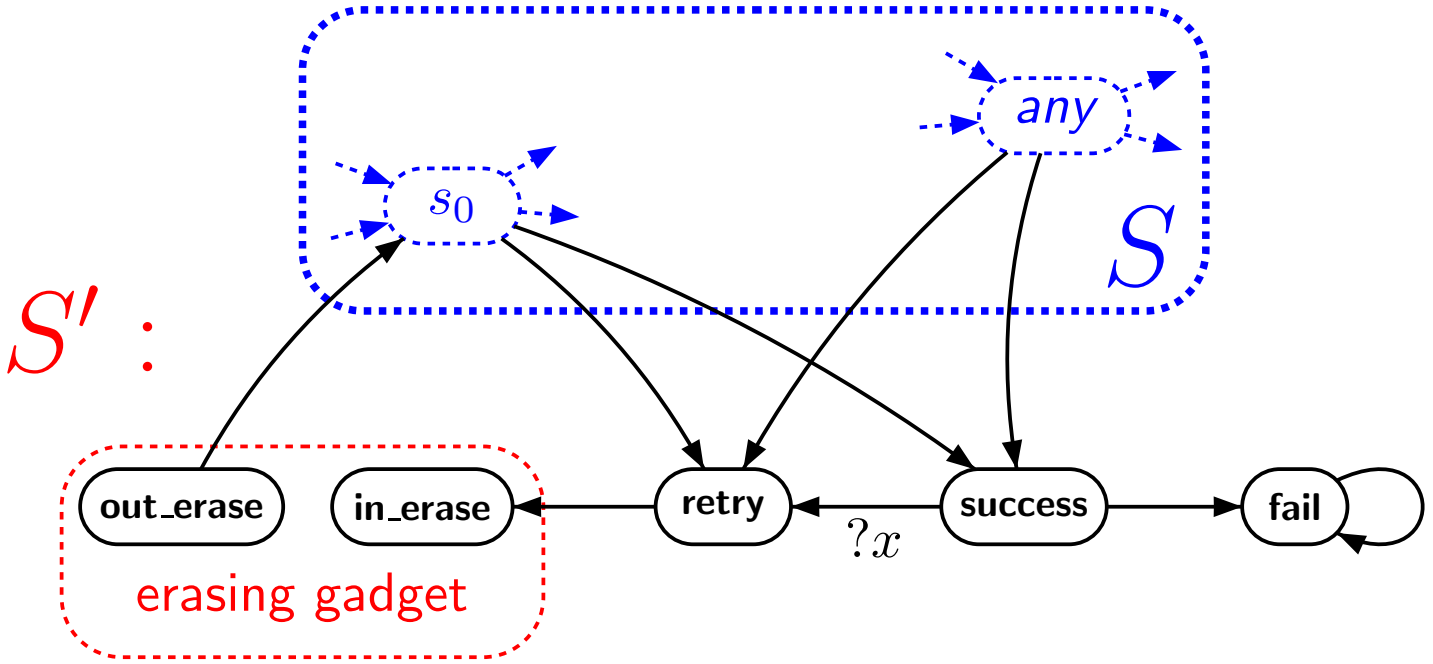
– early designs

You want a *Markov decision process* model, where rules are nondeterministic and losses are probabilistic!! [Bertrand & S. 2003].

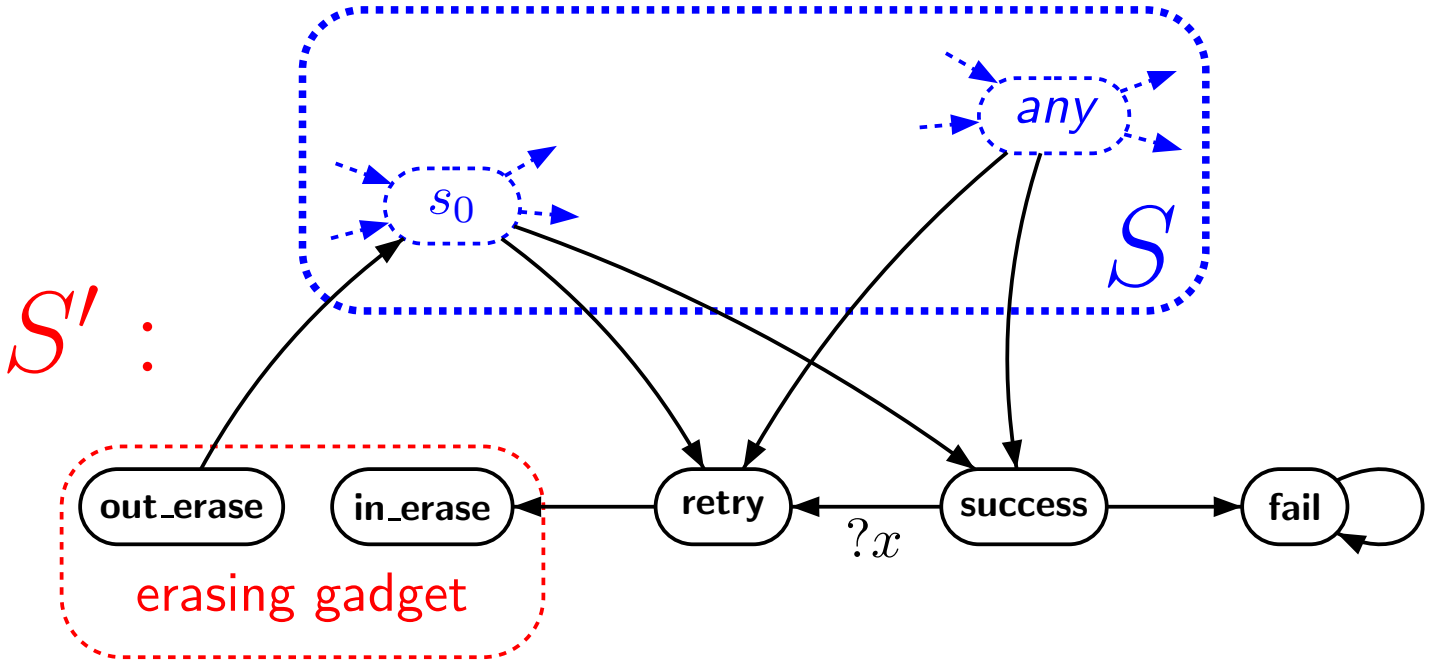
Then we can ask questions such as "*does* $\mathbb{P}(\varphi) = 1$ *under all scheduling policies?*" (This is the adversarial qualitative viewpoint)

Unrealistically nasty scheduling policies

Unrealistically nasty scheduling policies

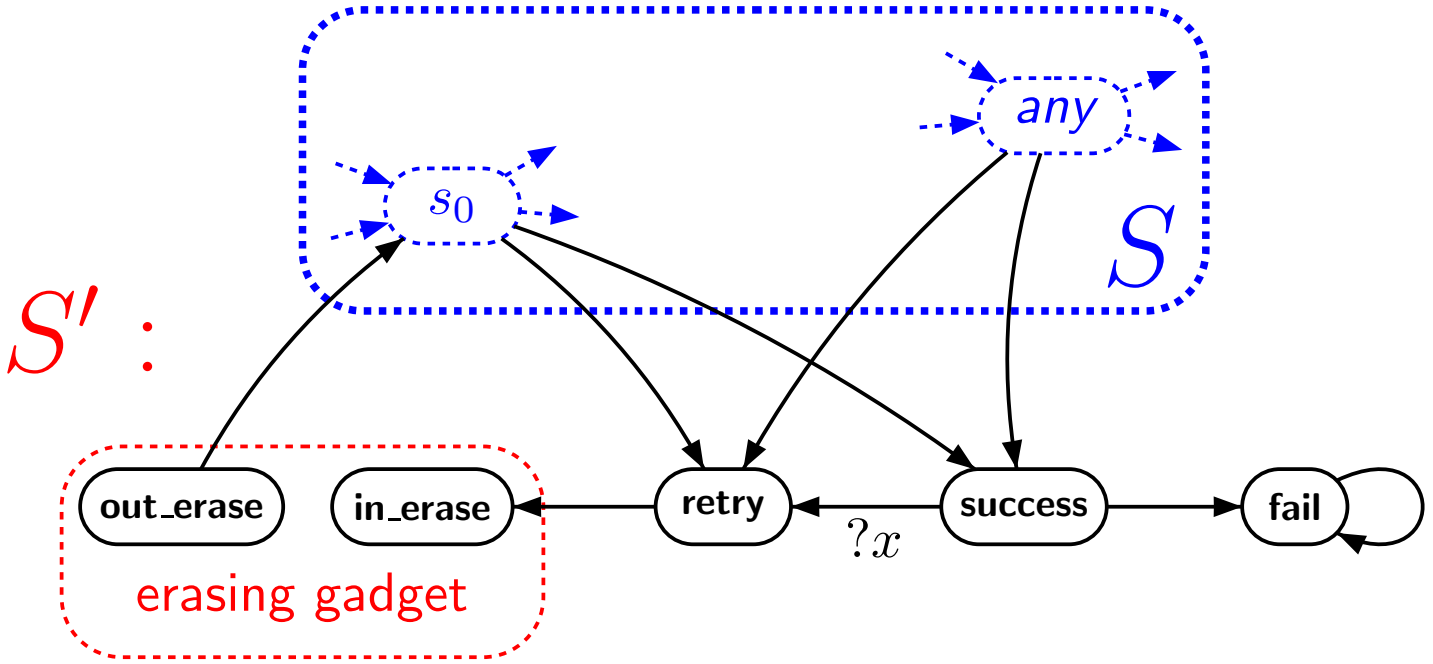


Unrealistically nasty scheduling policies



Question: is there a scheduling policy that makes S' visit **success** infinitely often with > 0 probability?

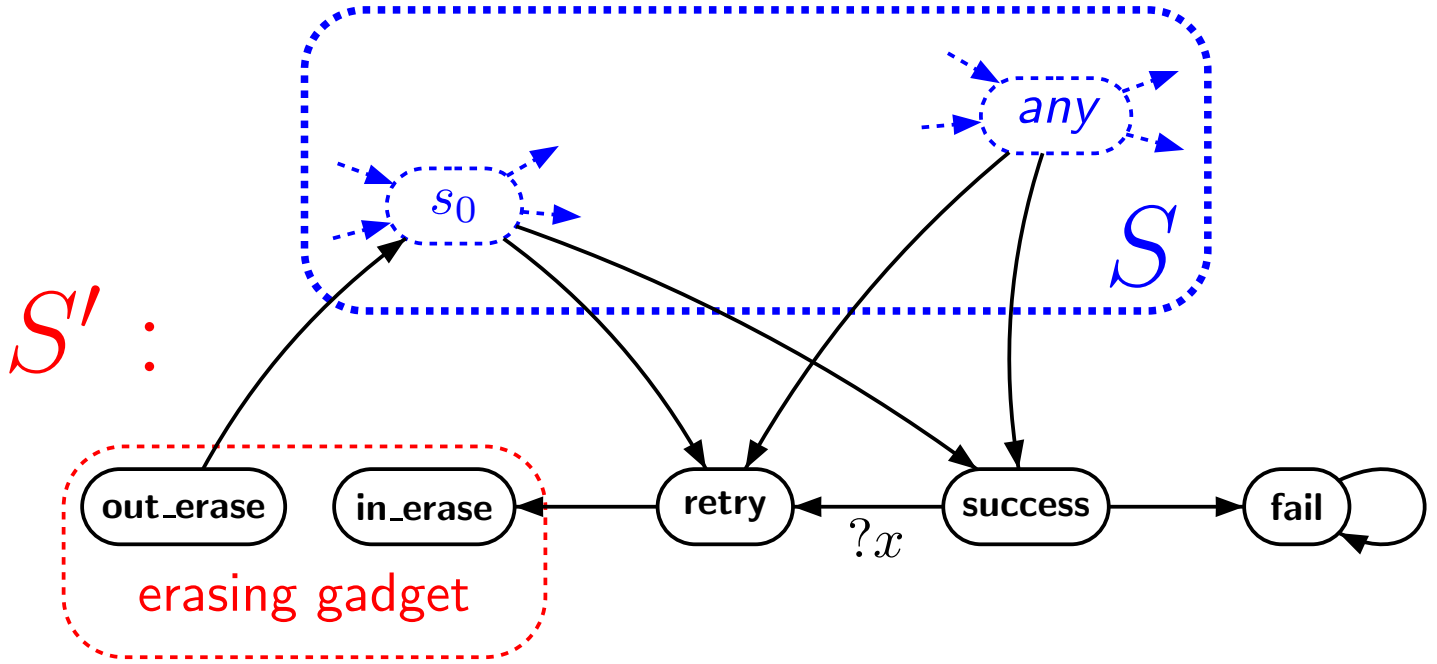
Unrealistically nasty scheduling policies



Question: is there a scheduling policy that makes S' visit **success** infinitely often with > 0 probability?

Answer: yes iff (nondeterministic) S is unbounded!

Unrealistically nasty scheduling policies



Question: is there a scheduling policy that makes S' visit **success** infinitely often with > 0 probability?

Answer: yes iff (nondeterministic) S is unbounded!

Corollary: model checking qualitative properties under all scheduling policies is undecidable.

All Is Not Lost!

In previous proof, the nasty scheduling policy is unrealistic.

E.g. it needs remember infinitely many things.

All Is Not Lost!

In previous proof, the nasty scheduling policy is unrealistic.

E.g. it needs remember infinitely many things.

Theorem (Bertrand & S. 2003): model checking qualitative properties under all *finite-memory policies* is decidable.

All Is Not Lost!

In previous proof, the nasty scheduling policy is unrealistic.

E.g. it needs remember infinitely many things.

Theorem (Bertrand & S. 2003): model checking qualitative properties under all *finite-memory policies* is decidable.

Some remaining open problems:

– What about cooperative qualitative model checking?

– What about computing minimal and maximal probabilities?

Concluding remarks

A fascinating model, bordering on undecidability.

Many open questions remain. Join us!!

Concluding remarks

A fascinating model, bordering on undecidability.

Many open questions remain. Join us!!

Lossy channel systems are useful too!

Algorithms for reachability and safety properties do work in practice.

Concluding remarks

A fascinating model, bordering on undecidability.

Many open questions remain. Join us!!

Lossy channel systems are useful too!

Algorithms for reachability and safety properties do work in practice.

Today, the Markovian decision process model with decidable adversarial qualitative properties is the best approximation to decidable model checking.

Will it work in practice?

Bibliography – 1

[AJ96a] P. A. Abdulla and B. Jonsson. Undecidable verification problems for programs with unreliable channels. *Information and Computation*, 130(1):71–90, 1996. http://www.docs.uu.se/docs/avds/publications/icalp94_ic.ps.

[AJ96b] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996. http://www.docs.uu.se/docs/avds/publications/lics93_ic.ps.

[AR03] P. A. Abdulla and A. Rabinovich. Verification of probabilistic systems with faulty communication. In *Proc. 6th Int. Conf. Foundations of Software Science and Computation Structures (FOSSACS'2003), Warsaw, Poland, Apr. 2003*, volume 2620 of *Lecture Notes in Computer Science*, pages 39–53. Springer, 2003. http://www.math.tau.ac.il/~rabinoa/fossacs.ps.gz.

[BE99] C. Baier and B. Engelen. Establishing qualitative properties for probabilistic lossy channel systems: An algorithmic approach. In *Proc. 5th Int. AMAST Workshop Formal Methods for Real-Time and Probabilistic Systems (ARTS'99), Bamberg, Germany, May 1999*, volume 1601 of *Lecture Notes in Computer Science*, pages 34–52. Springer, 1999. http://web.informatik.uni-bonn.de/I/papers/arts99.ps.

[Boc78] G. von Bochmann. Finite state description of communication protocols. *Computer Networks and ISDN Systems*, 2:361–372, 1978.

Bibliography – 2

[BS03] N. Bertrand and Ph. Schnoebelen. Model checking lossy channels systems is probably decidable. In *Proc. 6th Int. Conf. Foundations of Software Science and Computation Structures (FOSSACS'2003), Warsaw, Poland, Apr. 2003*, volume 2620 of *Lecture Notes in Computer Science*, pages 120–135. Springer, 2003. http://www.lsv.ens-cachan.fr/Publis/PAPERS/BerSch-fossacs2003.ps.

[BZ83] D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, 1983.

[DJS99] C. Dufourd, P. Jančar, and Ph. Schnoebelen. Boundedness of Reset P/T nets. In *Proc. 26th Int. Coll. Automata, Languages, and Programming (ICALP'99), Prague, Czech Republic, July 1999*, volume 1644 of *Lecture Notes in Computer Science*, pages 301–310. Springer, 1999. http://www.lsv.ens-cachan.fr/Publis/PAPERS/DJS-icalp99.ps.

[Fin94] A. Finkel. Decidability of the termination problem for completely specificied protocols. *Distributed Computing*, 7(3):129–135, 1994.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc. (3)*, 2(7):326–336, 1952.

[May03] R. Mayr. Undecidable problems in unreliable computations. *Theoretical Computer Science*, 297(1–3):337–354, 2003. http://www.informatik.uni-freiburg.de/~mayrri/lcmtcs.ps.

Bibliography – 3

[MS02] B. Masson and Ph. Schnoebelen. On verifying fair lossy channel systems. In *Proc. 27th Int. Symp. Math. Found. Comp. Sci. (MFCS'2002), Warsaw, Poland, Aug. 2002*, volume 2420 of *Lecture Notes in Computer Science*, pages 543–555. Springer, 2002. http://www.lsv.ens-cachan.fr/Publis/PAPERS/MS-mfcs2002-long.ps.

[Sch01] Ph. Schnoebelen. Bisimulation and other undecidable equivalences for lossy channel systems. In *Proc. 4th Int. Symp. Theoretical Aspects of Computer Software (TACS'2001), Sendai, Japan, Oct. 2001*, volume 2215 of *Lecture Notes in Computer Science*, pages 385–399. Springer, 2001. http://www.lsv.ens-cachan.fr/Publis/PAPERS/Sch-tacs2001.ps.

[Sch02] Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261, 2002. http://www.lsv.ens-cachan.fr/Publis/PAPERS/Sch-IPL2002.ps.