Verification of nondeterministic channel systems with probabilistic message losses

Joint work with C. Baier and N. Bertrand

Ph. Schnoebelen

Lab. Specification & Verification (LSV)
ENS de Cachan & CNRS
Cachan, France

Outline of the Talk

Lossy Channel Systems

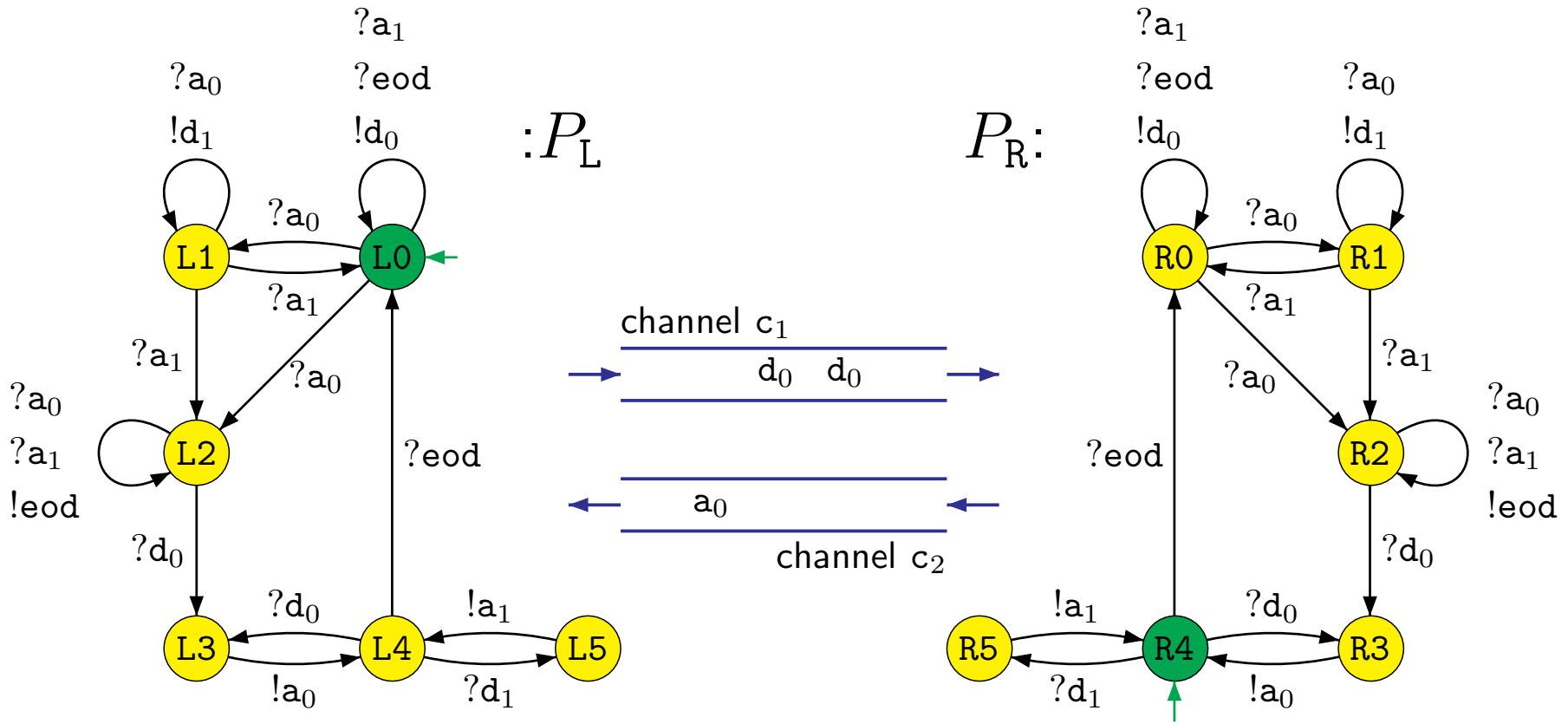
The Problem of Verifying Liveness Properties

Probabilistic Lossy Channel Systems

Nondeterministic and Probabilistic Lossy Channel Systems

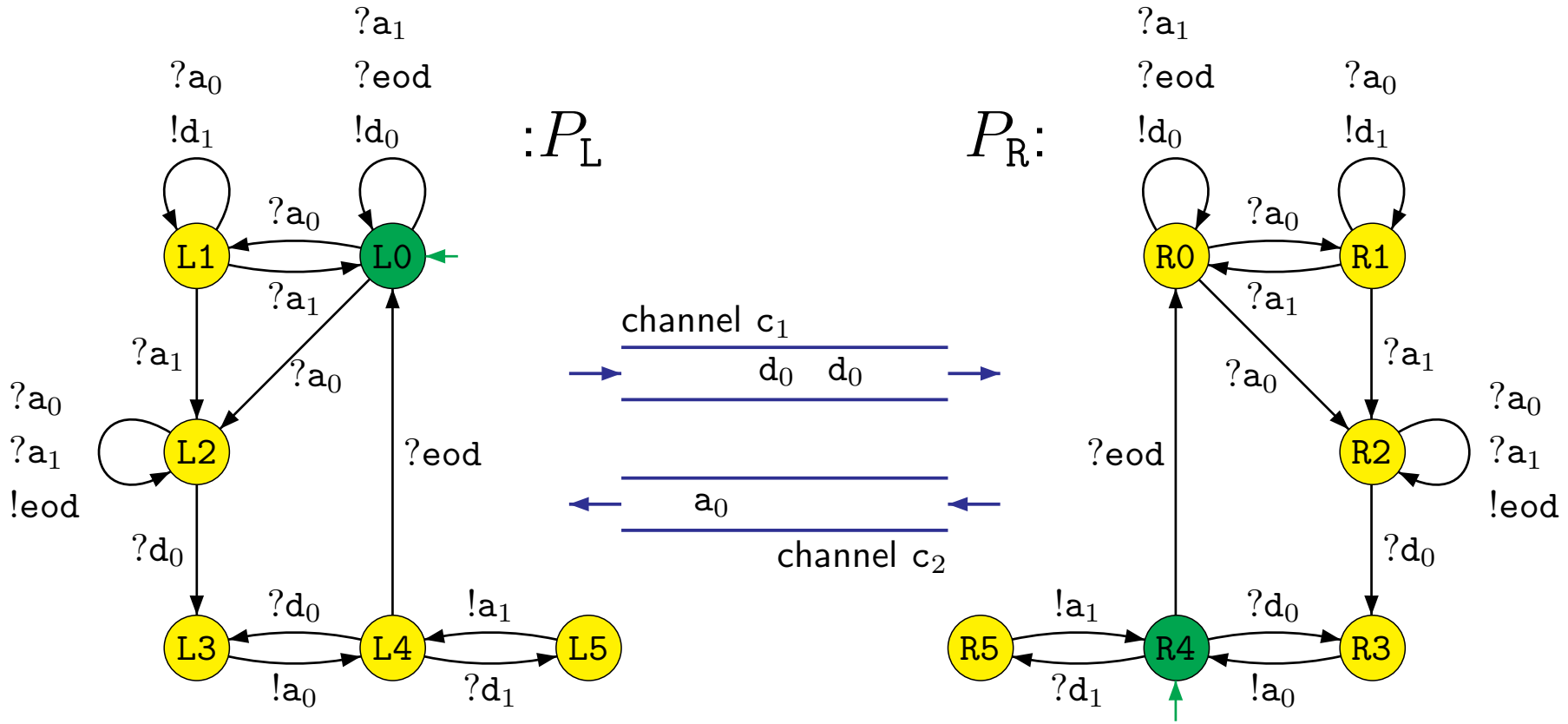
A long trial-and-error search for the right question to answer

Lossy Channel Systems



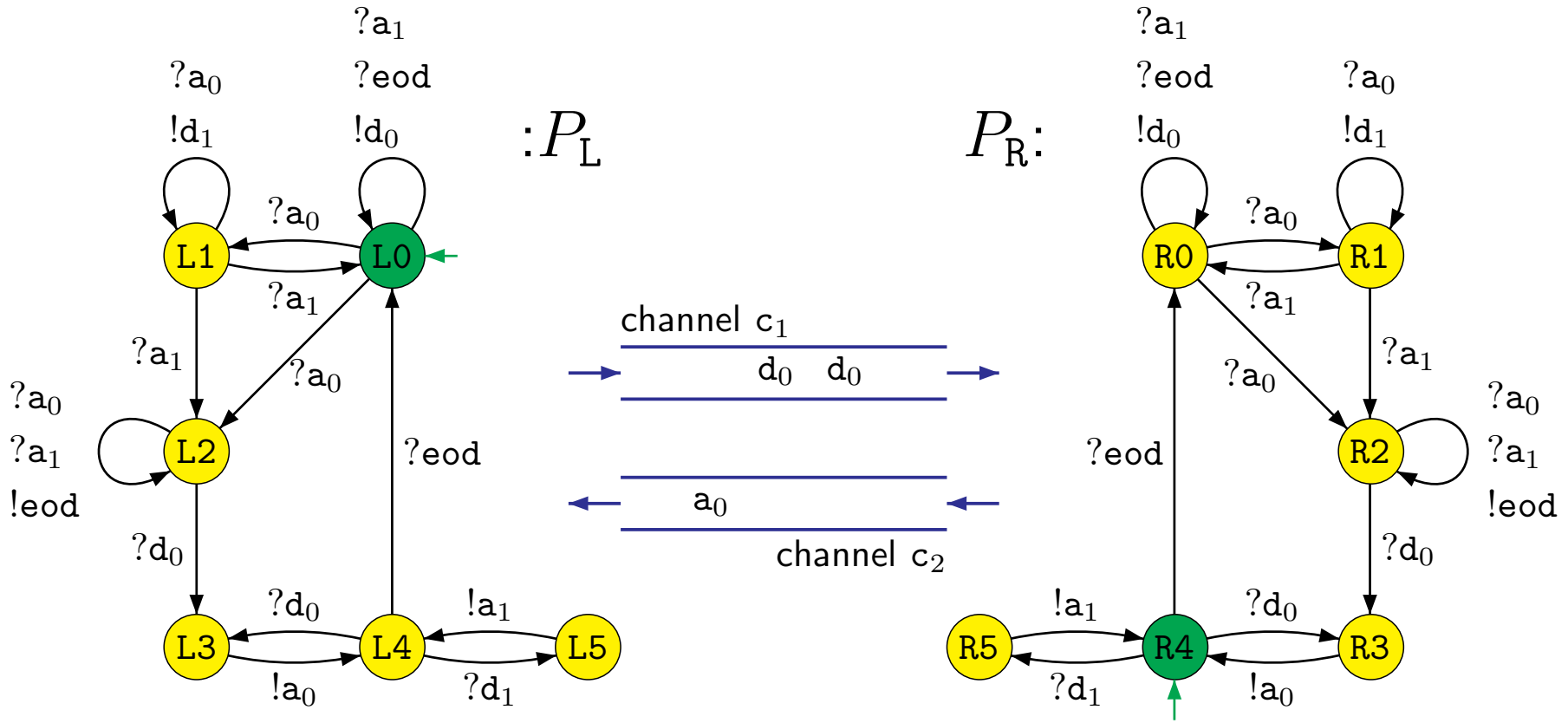
A model for asynchronous protocols that can handle unreliable communication

Lossy Channel Systems



Transition-system semantics: $\langle \mathtt{L0}, \mathtt{R4}, \mathtt{d}_0\,\mathtt{d}_0, \mathtt{a}_0 \rangle \rightarrow \langle \mathtt{L0}, \mathtt{R3}, \mathtt{d}_0, \mathtt{a}_0 \rangle$

Lossy Channel Systems



Message losses: $\langle \mathtt{L0}, \mathtt{R4}, \mathtt{d}_0\,\mathtt{d}_0, \mathtt{a}_0\rangle \rightarrow \langle \mathtt{L0}, \mathtt{R3}, \mathtt{d}_0, \varepsilon\rangle \qquad (\sqsubseteq \langle \mathtt{L0}, \mathtt{R3}, \mathtt{d}_0, \mathtt{a}_0\rangle)$

Verifying Lossy Channel Systems

• Reachability, safety properties, termination, many regular equivalences, are decidable (Abdulla & Jonsson, Finkel & Purush Iyer & Cécé, Kučera & S.)

• Regular model checking works in practice (Abdulla & Annichini & Bouajjani & Jonnson) even though worst-case complexity is nonprimitive-recursive

• Ingredients for decidability: symbolic computation of reachability set + termination based on wqo theory (here Higman's lemma)

Fine for verifying safety properties!

What About Liveness Properties?

Lossy channel systems are not so good for verifying liveness properties:

Problem 1: Liveness properties are undecidable for LCS's

Problem 2: Liveness properties are always false on LCS's

What About Liveness Properties?

Lossy channel systems are not so good for verifying liveness properties:

Problem 1: Liveness properties are undecidable for LCS's

Problem 2: Liveness properties are always false on LCS's

Fix 2: Add fairness assumptions everywhere needed

(should teach the system how to behave like a good system!)

Problem 1+: Adding fairness makes liveness even more undecidable

What About Liveness Properties?

Lossy channel systems are not so good for verifying liveness properties:

Problem 1: Liveness properties are undecidable for LCS's

Problem 2: Liveness properties are always false on LCS's

Fix 2: Add fairness assumptions everywhere needed

(should teach the system how to behave like a good system!)

Problem 1+: Adding fairness makes liveness even more undecidable

Fix 1+: Go for a probabilistic model

(nasty behaviours can be swept under the *measure zero* rug)

Probabilistic Lossy Channel Systems

Basic idea is to assume that *message losses follow probabilistic rules*, e.g. there is a known "failure rate" (Purush Iyer)

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

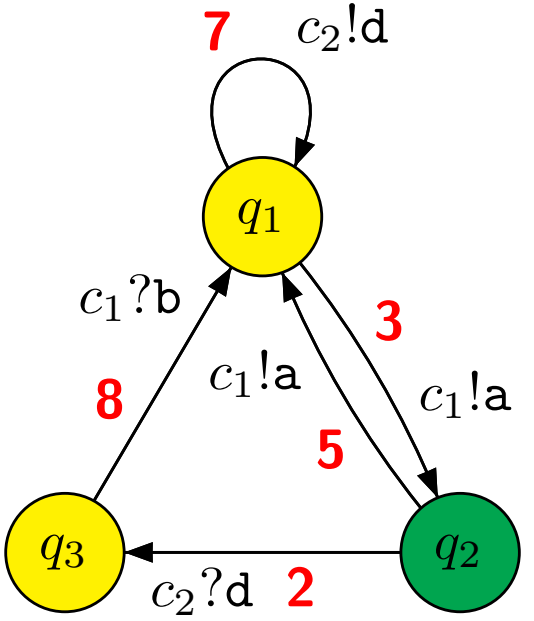
Probabilistic Lossy Channel Systems

Basic idea is to assume that *message losses follow probabilistic rules*, e.g. there is a known "failure rate" (Purush Iyer)

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

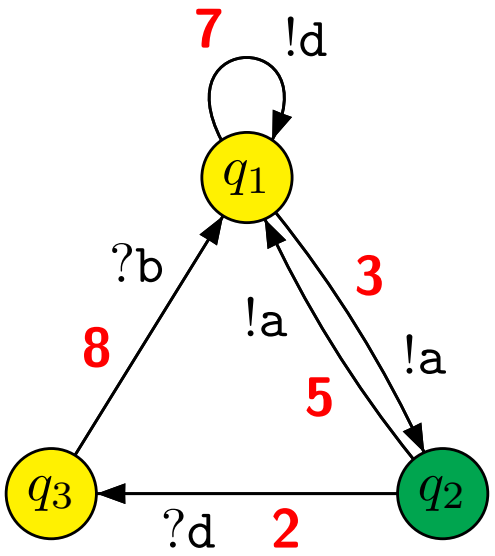
Definition: A Probabilistic LCS (a "PLCS") is
a LCS equipped with
– positive weights on rules,
– a constant probability $p_{\text{loss}} \in (0, 1)$

$$p_{\text{loss}} = .01 + \quad \frac{7}{8} \quad \frac{3}{5} \quad \frac{2}{}$$



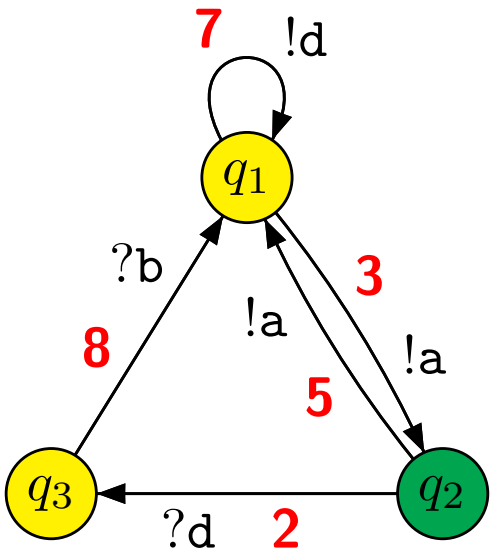
Markovian Semantics

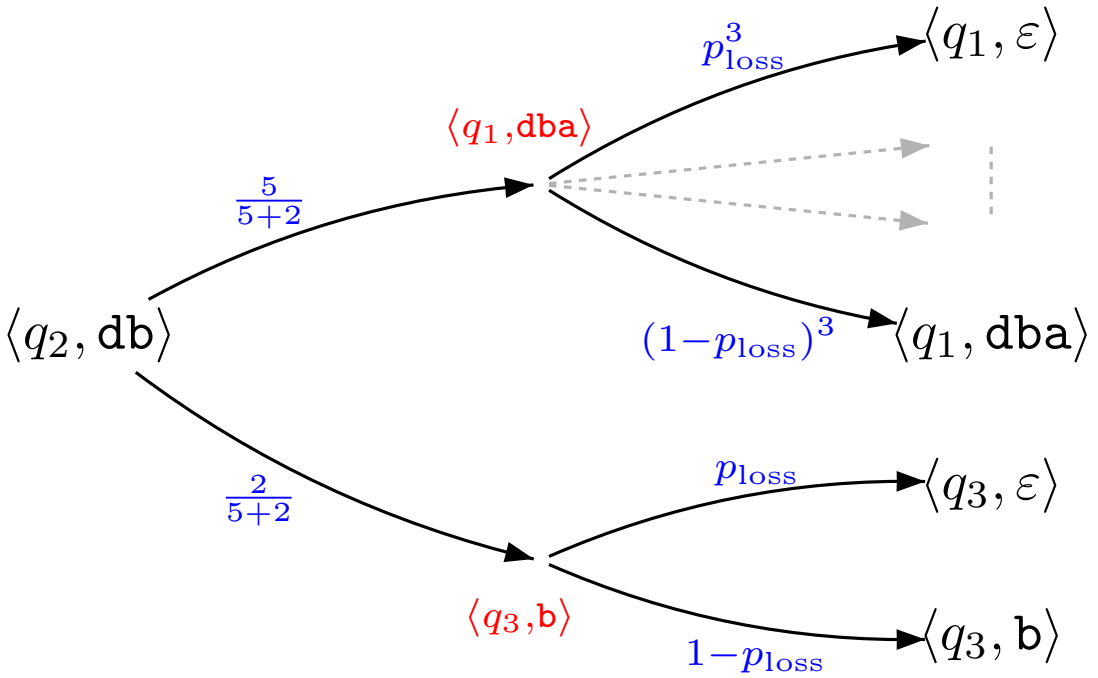
Semantics in form of a countable Markov chain



Markovian Semantics

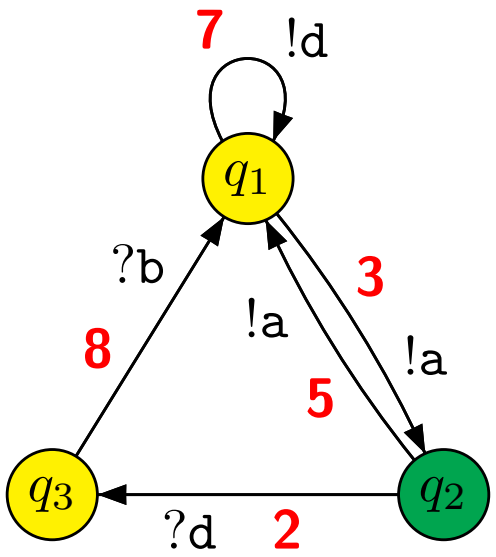
Semantics in form of a countable Markov chain

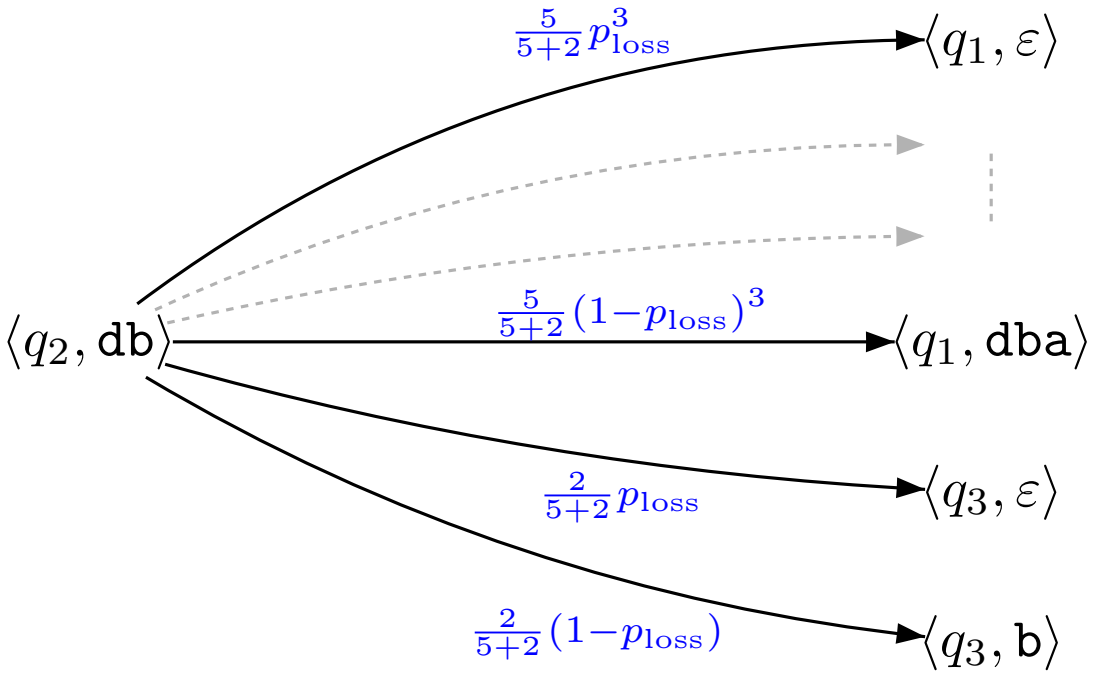




Markovian Semantics

Semantics in form of a countable Markov chain





Verification of PLCS's

Thm. Qualitative verification "*does φ hold almost surely?*" of linear-time properties is decidable (Purush Iyer, Baier, Abdulla, Jonsson, Rabinovich, Bertrand, S.)

Thm. Approximate quantitative verification is decidable (Rabinovich).

Ingredients for decidability:

1. Reduction to reachability questions for LCS's.
2. Existence of a finite attractor, i.e., a finite set of configurations that will be visited almost surely from any starting configuration.

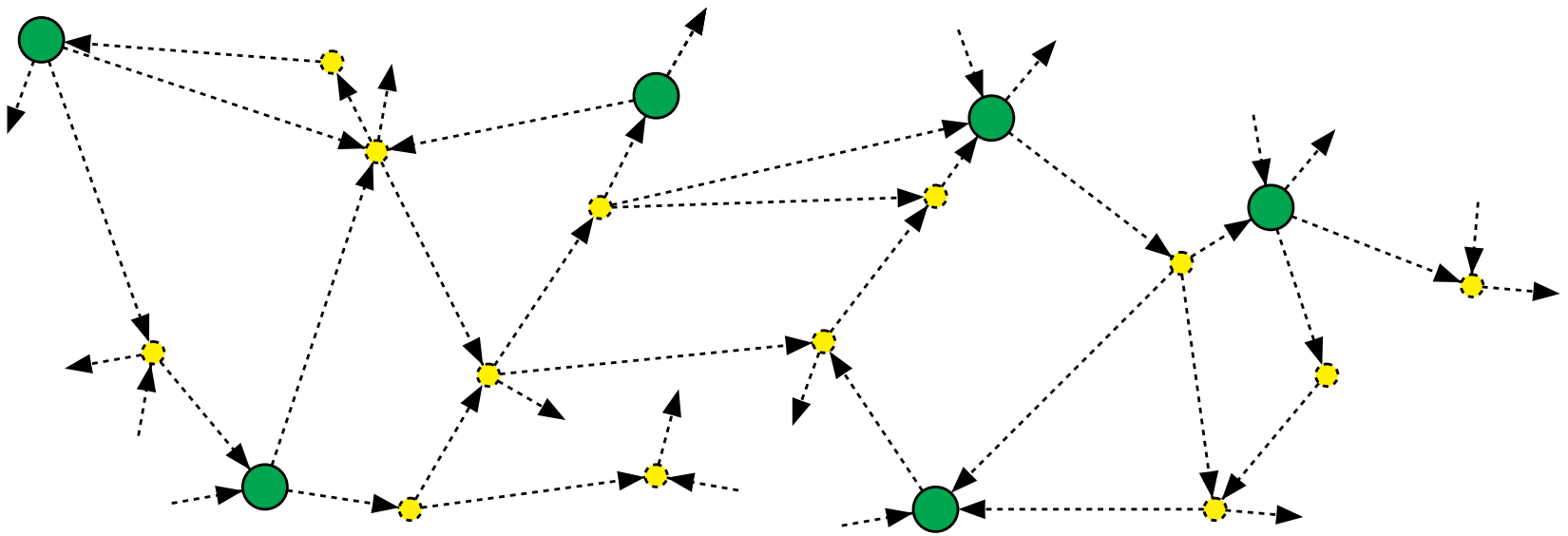
Verification of PLCS's

Thm. Qualitative verification "*does φ hold almost surely?*" of linear-time properties is decidable (Purush Iyer, Baier, Abdulla, Jonsson, Rabinovich, Bertrand, S.)

Thm. Approximate quantitative verification is decidable (Rabinovich).

Ingredients for decidability:

1. Reduction to reachability questions for LCS's.
2. Existence of a finite attractor, i.e., a finite set of configurations that will be visited almost surely from any starting configuration.



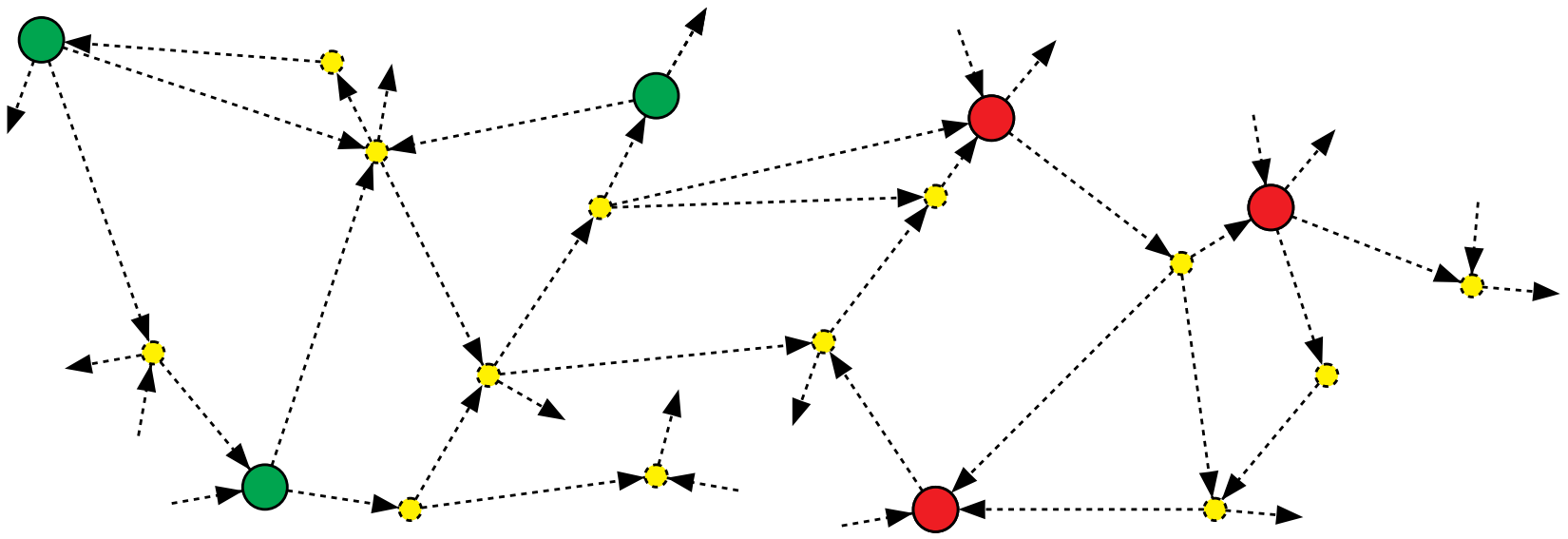
Verification of PLCS's

Thm. Qualitative verification "*does φ hold almost surely?*" of linear-time properties is decidable (Purush Iyer, Baier, Abdulla, Jonsson, Rabinovich, Bertrand, S.)

Thm. Approximate quantitative verification is decidable (Rabinovich).

Ingredients for decidability:

1. Reduction to reachability questions for LCS's.
2. Existence of a finite attractor, i.e., a finite set of configurations that will be visited almost surely from any starting configuration.



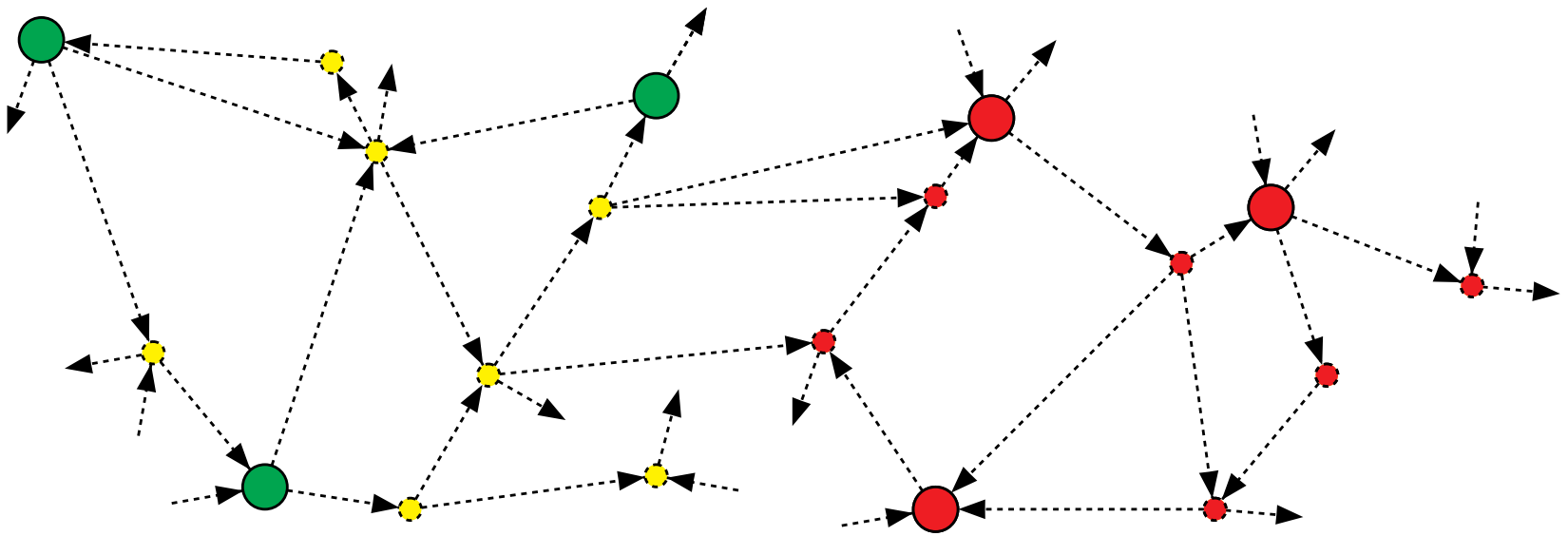
Verification of PLCS's

Thm. Qualitative verification "*does φ hold almost surely?*" of linear-time properties is decidable (Purush Iyer, Baier, Abdulla, Jonsson, Rabinovich, Bertrand, S.)

Thm. Approximate quantitative verification is decidable (Rabinovich).

Ingredients for decidability:

1. Reduction to reachability questions for LCS's.
2. Existence of a finite attractor, i.e., a finite set of configurations that will be visited almost surely from any starting configuration.



Markov-Chain Model Still not Adequate

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

Classically, nondeterminism in rules is used for:

– arbitrary interleaving of asynchronous components

– abstraction of real-life programs

– open systems

– early designs

– unpredictable message losses

Markov-Chain Model Still not Adequate

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

Classically, nondeterminism in rules is used for:

– arbitrary interleaving of asynchronous components

– abstraction of real-life programs

– open systems

– early designs

– unpredictable message losses

Not all of this can be adequately modeled by probabilities.

Markov-Chain Model Still not Adequate

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

Classically, nondeterminism in rules is used for:

– arbitrary interleaving of asynchronous components

– abstraction of real-life programs

– open systems

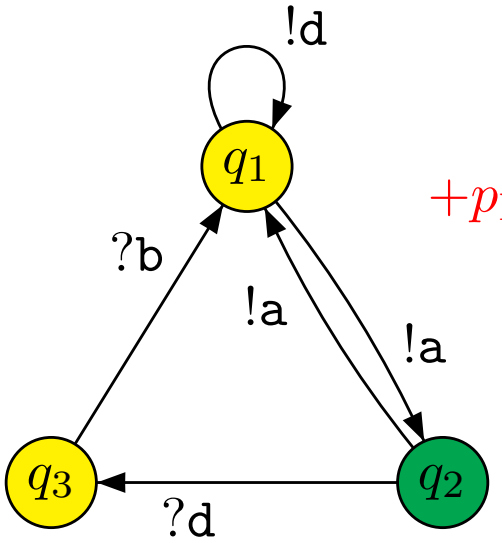
– early designs

– unpredictable message losses

Not all of this can be adequately modeled by probabilities.

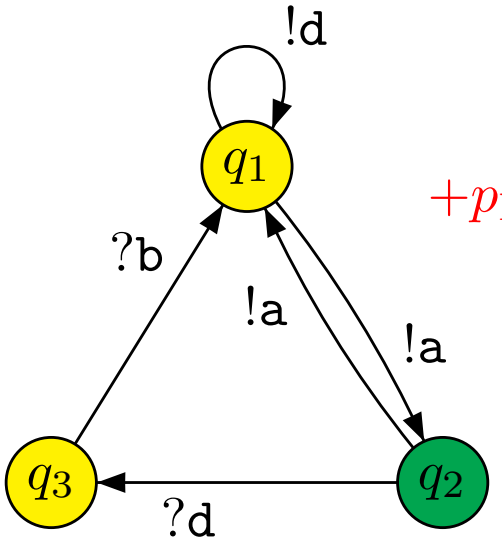
You want a *Markov decision process* model, where rules are nondeterministic and losses are probabilistic!! [Bertrand & S. 2003].

NPLCS and Markov Decision Processes

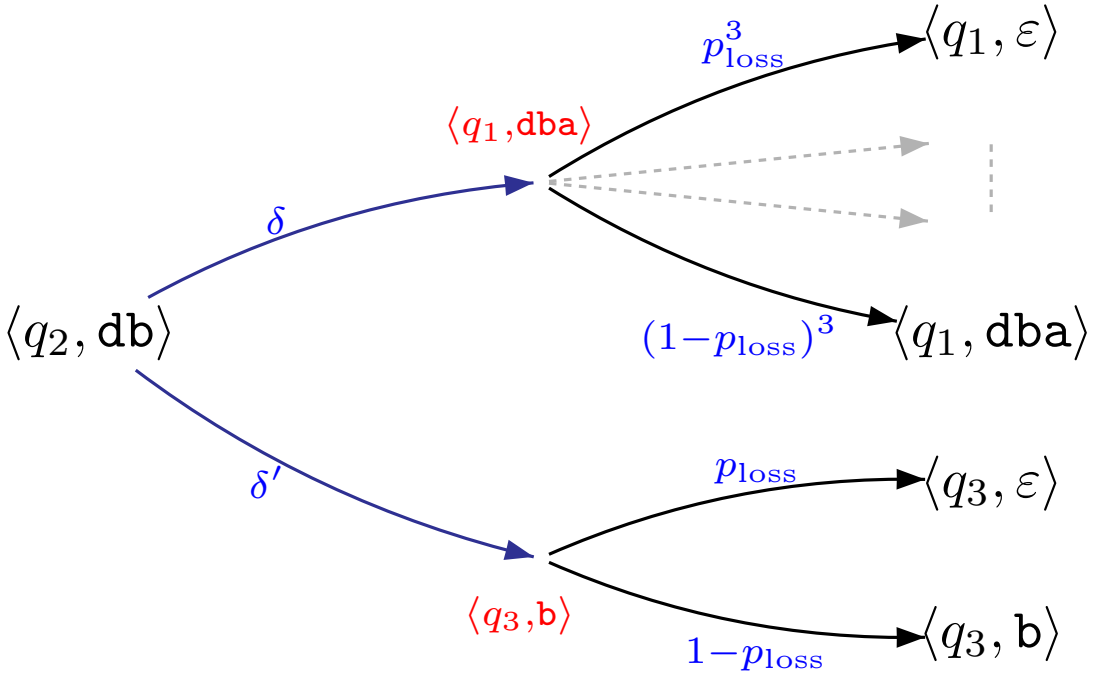


$+p_{\text{loss}} = 0.01$

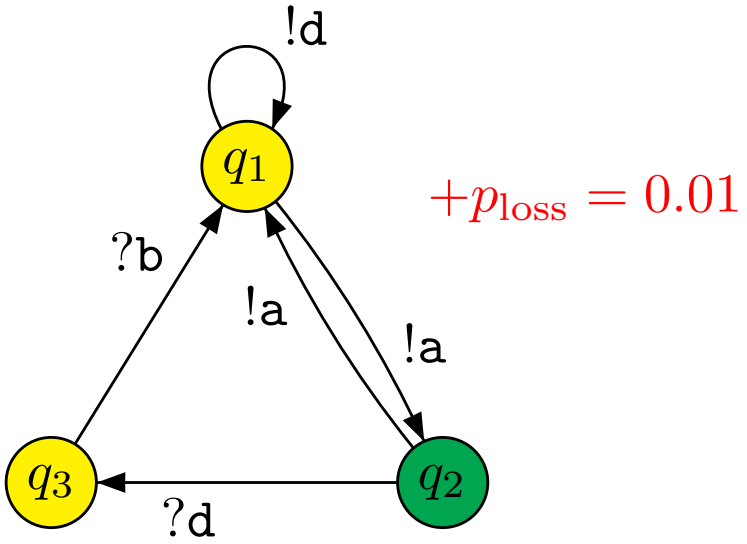
NPLCS and Markov Decision Processes



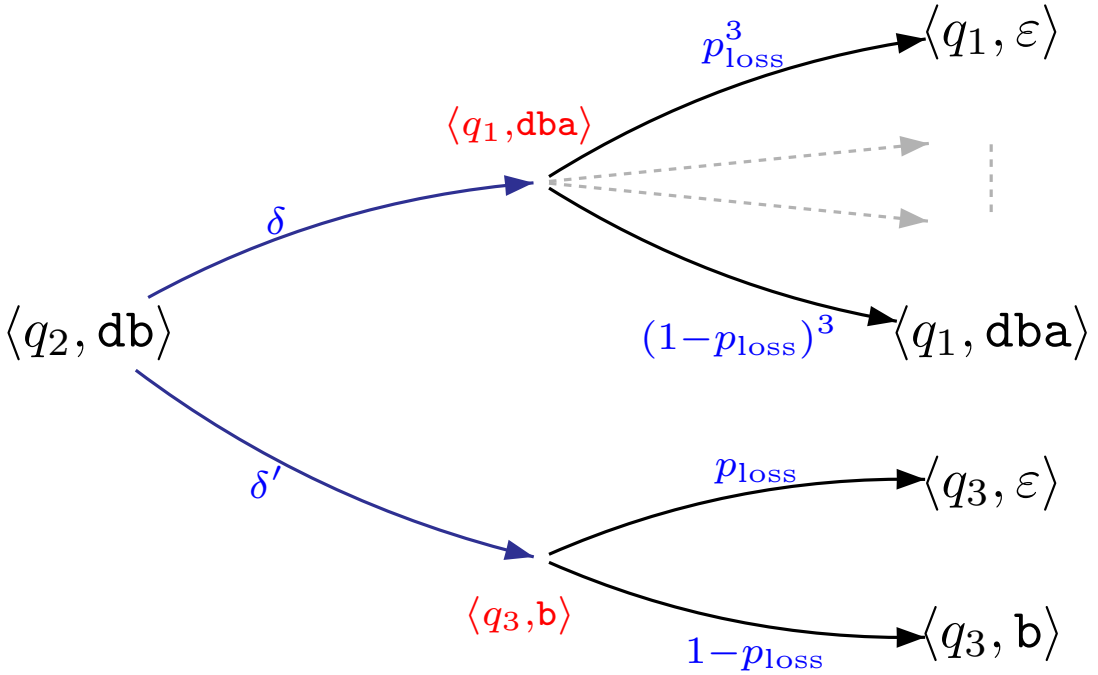
$+p_{\text{loss}} = 0.01$



NPLCS and Markov Decision Processes



$+p_{\text{loss}} = 0.01$



Def. A *scheduler* $\mathcal{U} : \text{Conf}^+ \to \Delta$ picks an enabled rule $\delta = \mathcal{U}(h)$ based on the complete history of the system.

Given $\mathcal{U}$, a NPLCS $\mathcal{N}$ generates a Markov chain $\mathcal{M}(\mathcal{N},\mathcal{U})$ and a probability measure $\mathbb{P}_{\mathcal{U}}(s \models \varphi)$.

Verifying NPLCS's

Thm. Qualitative verification of the form

$$\text{does } \mathbb{P}_{\mathcal{U}}(s \models \varphi) = 1 \text{ for all finite-memory } \mathcal{U}?$$

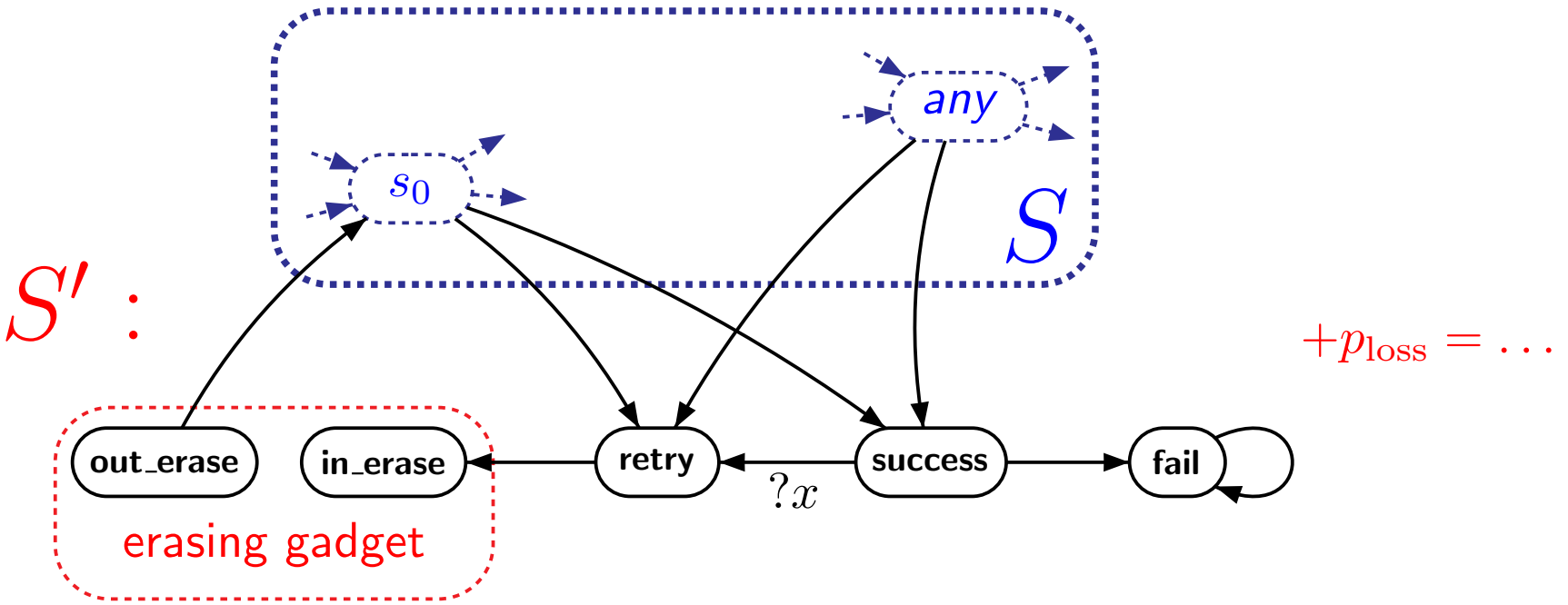
is decidable for NPLCS's (Baier, Bertrand, S.)

Ingredients for decidability:

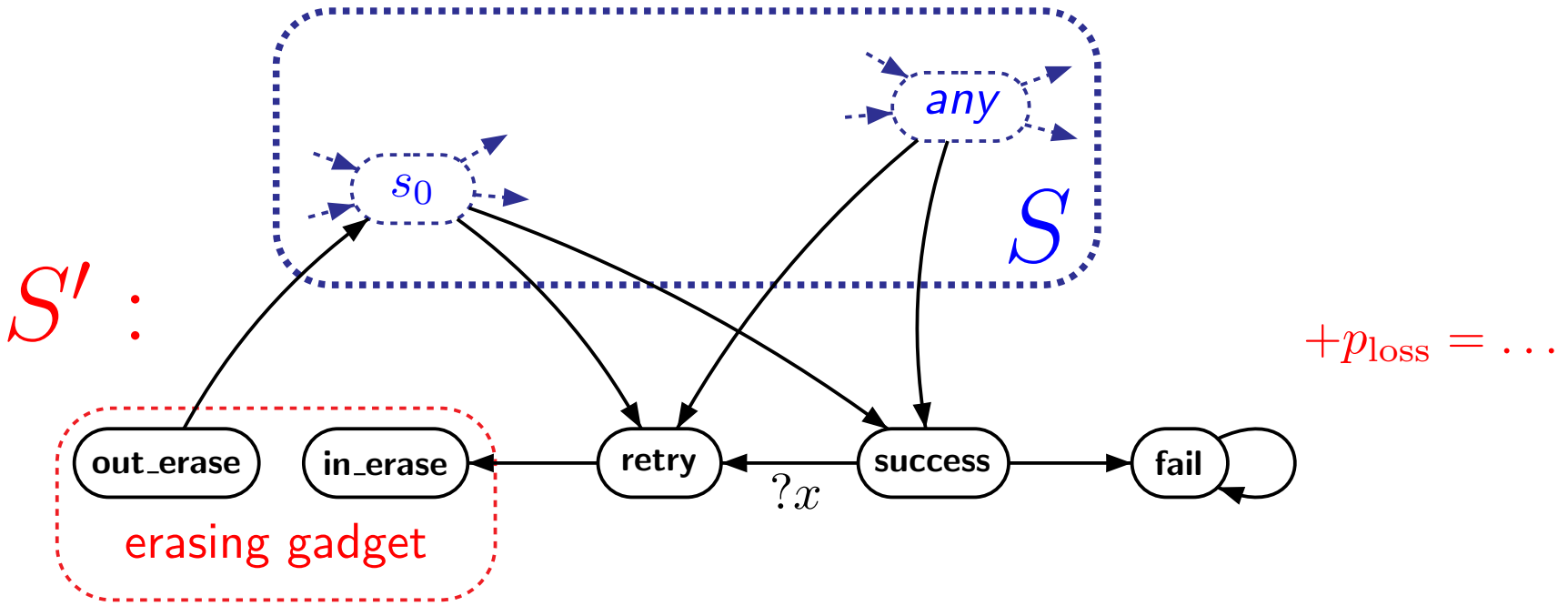
1. restriction to finite-memory schedulers
2. existence of a finite attractor
3. reduction to constrained reachability questions with fixpoints
4. symbolic decision method for the extended reachability questions

What Unrestricted Schedulers Can Do

What Unrestricted Schedulers Can Do

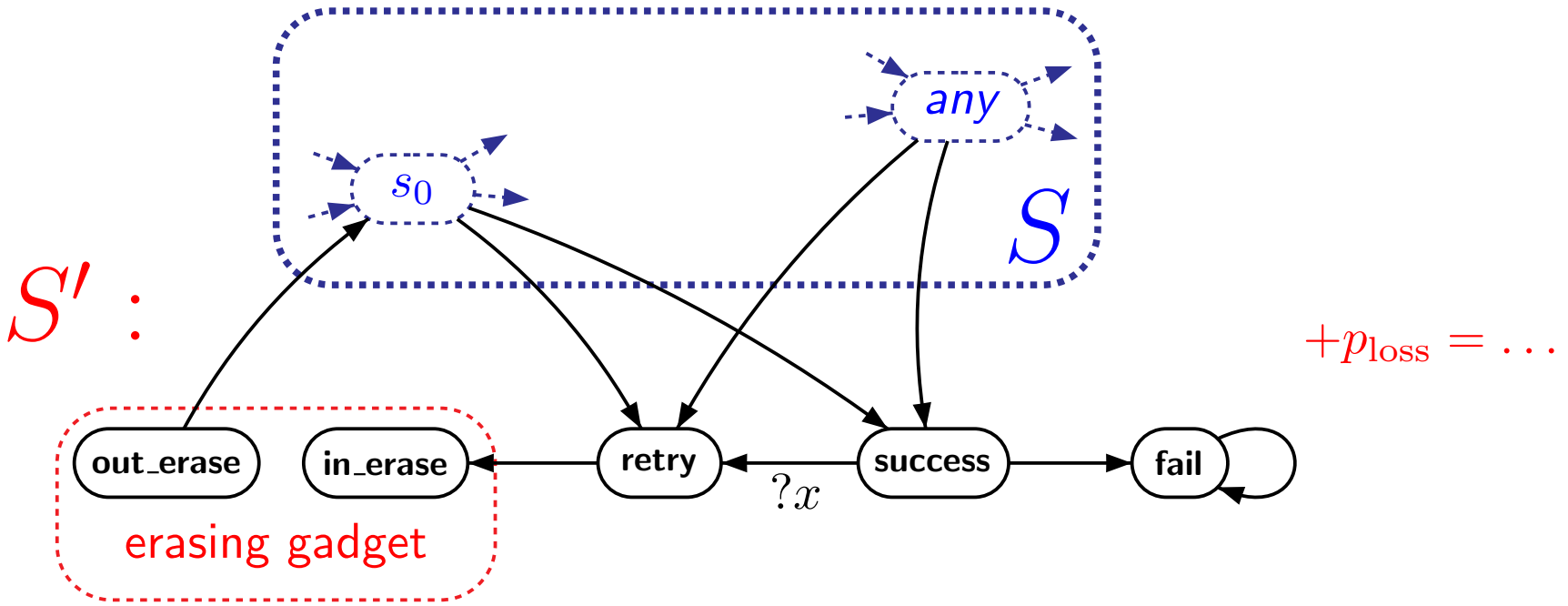


What Unrestricted Schedulers Can Do



Question: is there a scheduler $\mathcal{U}$ that makes S' visit **success** infinitely often with > 0 probability?

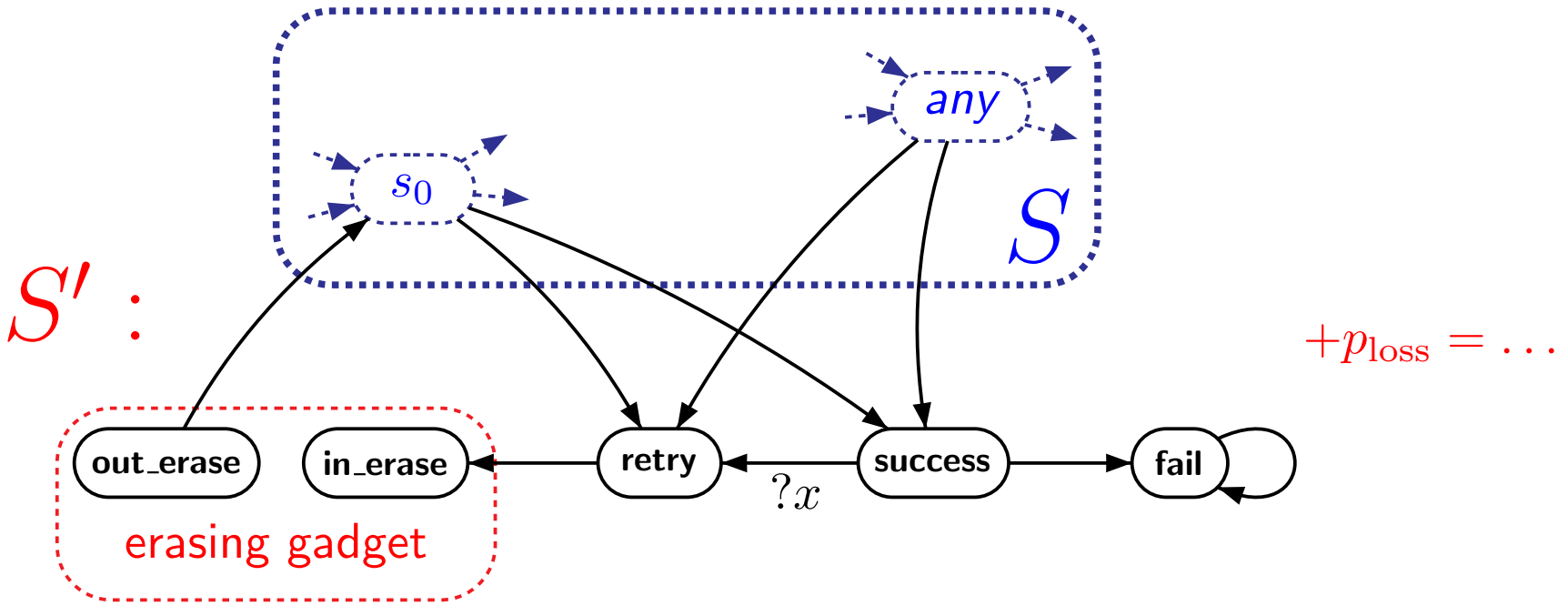
What Unrestricted Schedulers Can Do



Question: is there a scheduler $\mathcal{U}$ that makes S' visit **success** infinitely often with > 0 probability?

Answer: yes iff (plain LCS) S is unbounded!

What Unrestricted Schedulers Can Do



Question: is there a scheduler $\mathcal{U}$ that makes S' visit **success** infinitely often with > 0 probability?

Answer: yes iff (plain LCS) S is unbounded!

Corollary: model checking qualitative properties under all schedulers is undecidable.

Reducing to Reachability

Thm. 1. There is a scheduler $\mathcal{U}$ s.t. $\mathbb{P}_{\mathcal{U}}(s \models \Box A) = 1$ iff $s \models \mathit{Safe}(A)$
2. There is a scheduler $\mathcal{U}$ s.t. $\mathbb{P}_{\mathcal{U}}(s \models \Box A) > 0$ iff $s \models \exists(A \text{ Until } \mathit{Safe}(A))$

with $\mathit{Safe}(A) \stackrel{\text{def}}{=} \nu X.A \wedge \bigvee_{\delta \in \Delta} \big(\langle \delta \rangle \top \wedge [\delta]\, X\big)$

Reducing to Reachability

Thm. 1. There is a scheduler $\mathcal{U}$ s.t. $\mathbb{P}_{\mathcal{U}}(s \models \Box A) = 1$ iff $s \models \mathit{Safe}(A)$
2. There is a scheduler $\mathcal{U}$ s.t. $\mathbb{P}_{\mathcal{U}}(s \models \Box A) > 0$ iff $s \models \exists(A \text{ Until } \mathit{Safe}(A))$

with $\mathit{Safe}(A) \stackrel{\text{def}}{=} \nu X.A \wedge \bigvee_{\delta \in \Delta} \big(\langle \delta \rangle \top \wedge [\delta]\, X\big)$

Used to verify $\forall \mathcal{U}.\mathbb{P}_{\mathcal{U}}(s \models \Diamond \overline{A}) > 0$ and $\forall \mathcal{U}.\mathbb{P}_{\mathcal{U}}(s \models \Diamond \overline{A}) = 1$

Reducing to Reachability

Thm. 1. There is a scheduler $\mathcal{U}$ s.t. $\mathbb{P}_{\mathcal{U}}(s \models \Box A) = 1$ iff $s \models \mathit{Safe}(A)$
2. There is a scheduler $\mathcal{U}$ s.t. $\mathbb{P}_{\mathcal{U}}(s \models \Box A) > 0$ iff $s \models \exists(A \text{ Until } \mathit{Safe}(A))$

with $\mathit{Safe}(A) \stackrel{\text{def}}{=} \nu X.A \wedge \bigvee_{\delta \in \Delta}\big(\langle\delta\rangle\top \wedge [\delta]\, X\big)$

Used to verify $\forall\mathcal{U}.\mathbb{P}_{\mathcal{U}}(s \models \Diamond\overline{A}) > 0$ and $\forall\mathcal{U}.\mathbb{P}_{\mathcal{U}}(s \models \Diamond\overline{A}) = 1$

Thm. 3. There is a finite-memory scheduler $\mathcal{U}$ s.t. $\mathbb{P}_{\mathcal{U}}(s \models \Box\Diamond A \wedge \Diamond\Box B) = 1$ iff $s \models \exists \mathsf{F}\mathit{Prom}_B(A)$

with $\mathit{Prom}_B(A) \stackrel{\text{def}}{=} \nu X.\Big[A \wedge \mu Y.B \vee \bigvee_{\delta}\big(\langle\delta\rangle Y \wedge [\delta]\, X\big)\Big]$

Reducing to Reachability

Thm. 1. There is a scheduler $\mathcal{U}$ s.t. $\mathbb{P}_{\mathcal{U}}(s \models \Box A) = 1$ iff $s \models \mathit{Safe}(A)$
2. There is a scheduler $\mathcal{U}$ s.t. $\mathbb{P}_{\mathcal{U}}(s \models \Box A) > 0$ iff $s \models \exists(A \text{ Until } \mathit{Safe}(A))$

with $\mathit{Safe}(A) \stackrel{\text{def}}{=} \nu X.A \wedge \bigvee_{\delta \in \Delta}\big(\langle\delta\rangle\top \wedge [\delta]\, X\big)$

Used to verify $\forall\mathcal{U}.\mathbb{P}_{\mathcal{U}}(s \models \Diamond\overline{A}) > 0$ and $\forall\mathcal{U}.\mathbb{P}_{\mathcal{U}}(s \models \Diamond\overline{A}) = 1$

Thm. 3. There is a finite-memory scheduler $\mathcal{U}$ s.t. $\mathbb{P}_{\mathcal{U}}(s \models \Box\Diamond A \wedge \Diamond\Box B) = 1$ iff $s \models \exists \mathsf{F}\mathit{Prom}_B(A)$

with $\mathit{Prom}_B(A) \stackrel{\text{def}}{=} \nu X.\Big[A \wedge \mu Y.B \vee \bigvee_{\delta}\big(\langle\delta\rangle Y \wedge [\delta]\, X\big)\Big]$

Used to verify $\forall\mathcal{U}.\mathbb{P}_{\mathcal{U}}(s \models \Box\Diamond A \Rightarrow \Box\Diamond\overline{B}) = 1$

NB. When verifying liveness, sets A and B cannot just be defined in terms of control locations. One has to take channel contents into account!

Symbolic Computation of Reachability Sets

Prefixed closure $\quad \theta, \sigma \ ::= \ \alpha \uparrow u \qquad \alpha \in \mathsf{M} \cup \{\varepsilon\}, \ \ u \in \mathsf{M}^*$

Symbolic set $\quad \gamma \ ::= \ \sum_i \langle q_i, \theta_i - \sigma_{i,1} - \cdots - \sigma_{i,n-i} \rangle$

A region is any set of configurations that can be denoted symbolically.

Symbolic Computation of Reachability Sets

Prefixed closure $\quad \theta, \sigma \ ::= \ \alpha \uparrow u \qquad \alpha \in \mathsf{M} \cup \{\varepsilon\}, \ \ u \in \mathsf{M}^*$

Symbolic set $\quad \gamma \ ::= \ \sum_i \langle q_i, \theta_i - \sigma_{i,1} - \cdots - \sigma_{i,n-i} \rangle$

A region is any set of configurations that can be denoted symbolically.

Thm. Regions are closed under Boolean operations, Pre-image, $\exists(\_ \ \mathsf{Until} \ \_)$, $Safe()$, $Prom_{(\_)}(\_)$.

Termination of fixpoint symbolic computations requires careful analysis of how restrictions appear in symbolic sets.

Prototype implementation performs encouragingly well!

Concluding remarks

A model that combines two hard features: probabilities and infinite state space.

Qualitative verification "*for all finite-memory schedulers*" is decidable.

Positive result requires many ingredients. Hard to see what can be stated in a more generalized framework.

Many open questions remain. The priority should be on assessing the model and strengthening its foundations.