Stochastic models for lossy channel systems and automated verification techniques

Nathalie Bertrand

Laboratoire Spécification et Vérification
UMR 8643 (ENS de Cachan & CNRS)

October 6th, 2006
Communication protocols

- telecommunications, electronic devices, etc...
- set of rules required to send information
Communication protocols (cont.)

They are more than just transmitting information.
Communication protocols (cont.)

They are more than just transmitting information.
Communication protocols (cont.)

They are more than just transmitting information.

- initiate conversation
- ensure the other party is ready
- give the context of the communication
- make sure that the message was understood
- foresee faulty behaviors (transmission errors)
- agree on the last message
Formal methods for verification

Motivation

- communication protocols are everywhere
- more and more complex systems
- errors have huge consequences
Formal methods for verification

Motivation

- communication protocols are everywhere
- more and more complex systems
- errors have huge consequences

Techniques

- Test
- Computer-aided proofs
- Model checking
Principles of Model-checking

system

specification
Principles of Model-checking

Does the system satisfy the specification?
Principles of Model-checking

Does the system satisfy the specification?
Principles of Model-checking

Does the system satisfy the specification?

System → Model → formula

φ
Principles of Model-checking

Does the system satisfy the specification?

\[
\phi \models \text{model-checker}
\]
Channel systems

Finite processes that communicate via unbounded FIFO channels
[Brand Zafiropulo 83]
Channel systems

Finite processes that communicate via unbounded FIFO channels

[Brand Zafiropulo 83]

[Pachl 87]
Channel systems

Finite processes that communicate via unbounded FIFO channels

[Brand Zafiropulo 83]

[Pachl 87]
Channel systems

Finite processes that communicate via unbounded FIFO channels

[Brand Zafiropulo 83]

[Pachl 87]
Channel systems

Finite processes that communicate via unbounded FIFO channels

[Brand Zafiropulo 83]

[L0]

[R0]

[L1]

[R1]

[L0] <-> [R0]

[L1] <-> [R1]

[L2] <-> [R2]

[L3] <-> [R3]

[L4] <-> [R4]

[L5] <-> [R5]

Channel c1

Channel c2

[Brand Zafiropulo 83]

[Pachl 87]
Channel systems

Finite processes that communicate via unbounded FIFO channels

[Brand Zafiropulo 83]

[Pachl 87]

channel \( c_1 \)

channel \( c_2 \)
Channel systems

Finite processes that communicate via unbounded FIFO channels

[Brand Zafiropulo 83]

[Pachl 87]
Channel systems

Finite processes that communicate via unbounded FIFO channels

[Brand Zafiropulo 83]

[Pachl 87]
Channel systems

Finite processes that communicate via unbounded FIFO channels

[Brand Zafiropulo 83]

[Pachl 87]
Channel systems

Finite processes that communicate via unbounded FIFO channels

[Brand Zafiropulo 83]

[Pachl 87]
Channel systems

Finite processes that communicate via unbounded FIFO channels

[Brand Zafiropulo 83]

[Pachl 87]
Channel systems

Finite processes that communicate via unbounded FIFO channels

[Brand Zafiropulo 83]

[Pachl 87]
Channel systems

Finite processes that communicate via unbounded FIFO channels

[Brand Zafiropulo 83]

[Turing-powerful]

[Pachl 87]
Lossy Channel Systems (LCS)

- **Unreliable** channels: messages may be lost after read/write operations
Lossy Channel Systems (LCS)

- **Unreliable** channels: messages may be lost after read/write operations
State of the art for LCS

Decidability results

- Termination [Finkel 94]
- Reachability [Pachl87] [Abdulla, Jonsson 96]
- Safety properties [Abdulla, Jonsson 96]
- Eventuality [Abdulla, Jonsson 96]
State of the art for LCS

Decidability results

- Termination [Finkel 94]
- Reachability [Pachl87] [Abdulla, Jonsson 96]
- Safety properties [Abdulla, Jonsson 96]
- Eventuality [Abdulla, Jonsson 96]

Undecidability results

- Liveness LTL properties [Abdulla, Jonsson 96]
- Eventuality assuming fairness [Abdulla, Jonsson 96]
- Boundedness [Mayr 03]
- Equivalence between 2 systems [Schnoebelen 01]
State of the art for LCS

Decidability results

- Termination [Finkel 94]
- Reachability [Pachl87] [Abdulla, Jonsson 96]
- Safety properties [Abdulla, Jonsson 96]
- Eventuality [Abdulla, Jonsson 96]

Undecidability results

- Liveness LTL properties [Abdulla, Jonsson 96]
- Eventuality assuming fairness [Abdulla, Jonsson 96]
- Boundedness [Mayr 03]
- Equivalence between 2 systems [Schnoebelen 01]

[Schnoebelen 02] all decidable problems are non-primitive recursive
Contributions

LCS

Regular model checking
Contributions

Probabilistic LCS
- Local-fault model

Qualitative verification
- $\omega$-regular properties

Finite attractor

LCS
- Regular model checking
Contributions

Nondeterministic Probabilistic LCS

Probabilistic LCS
  local-fault model

Probabilistic LCS
  Qualitative verification
  ω-regular properties
  Finite attractor

LCS
  Regular model checking

Finite-memory (fair) schedulers
  qualitative verification
  ω-regular properties
  Finite attractor
Outline

Introduction

Regular Model Checking for LCS
  Syntactic criterion

Probabilistic LCS
  The Markov chain model
  Attractors
  Qualitative verification

Nondeterministic and Probabilistic LCS
  The Markov decision process model
  Undecidability results
  Finite-memory schedulers

Implementation

Conclusion
Outline

Introduction

Regular Model Checking for LCS
  Syntactic criterion

Probabilistic LCS
  The Markov chain model
  Attractors
  Qualitative verification

Nondeterministic and Probabilistic LCS
  The Markov decision process model
  Undecidability results
  Finite-memory schedulers

Implementation

Conclusion
Symbolic model checking

Issues:

- finite representation for infinite sets of configurations
Symbolic model checking

Issues:
- finite representation for infinite sets of configurations
Symbolic model checking

Issues:

- finite representation for infinite sets of configurations
Symbolic model checking

Issues:
- finite representation for infinite sets of configurations
Symbolic model checking

Issues:

- finite representation for infinite sets of configurations
- convergence of the fixpoint computation
Symbolic model checking

**Issues:**

- finite representation for infinite sets of configurations
- convergence of the fixpoint computation

**Regular model checking:**

work on regular sets of configurations handled via finite description (finite-state automata, regular expressions)
Regions: regular sets of configurations:

\[ R = \sum_{i \in I} (s_i, R_i^1, \ldots, R_i^{|C|}). \]

Example

\((s, \varepsilon, M^* m M^*) + (s, M^2*, \uparrow \varepsilon)\)
Region algebra for LCS

**Regions:** regular sets of configurations:

\[ R = \sum_{i \in I} (s_i, R_i^1, \ldots, R_i^{|C_i|}). \]

**Example**

\[(s, \varepsilon, M^* m M^*) + (s, M^2, \uparrow \varepsilon)\]

**Operators:** union, intersection, \(\uparrow, \downarrow, Pre, \overline{Pre}\)

- monotonic
- effective
- regularity preserving
Guardedness condition

**Definition: Guardedness**

A term $\phi$ of $\mu$-calculus is *guarded* if

- all least-fixpoints subterms $\mu X.\psi$ have $X$ upward-guarded in $\psi$
Definition: Guardedness

A term $\phi$ of $\mu$-calculus is **guarded** if

- all least-fixpoints subterms $\mu X.\psi$ have $X$ upward-guarded in $\psi$.

Theorem [LPAR’06]

Subsets defined by guarded terms are regions, and can be computed effectively provided the region algebra is effective.

- generalizes Higman’s lemma
- gives a syntactic criterion for convergence
Generalized LCS [LPAR’06]

- Transition: (regular) guards over channel contents + operation
Generalized LCS \[\text{[LPAR'06]}\]

- Transition: \((\text{regular})\) guards over channel contents +
  operation
  - more expressive
  - allows emptiness and occurrence tests
  - permits easier products with Büchi automata
Generalized LCS [LPAR’06]

- Transition: \textit{(regular) guards} over channel contents + operation
  - more expressive
  - allows emptiness and occurrence tests
  - permits easier products with Büchi automata

Reachability

\[
Pre^*(A) = \mu X. A \cup Pre(X) = \mu X. A \cup Pre(\uparrow X)
\]

Safety properties

\[
\forall (V_1 R V_2) = \nu X. (V_2 \cap (\overline{Pre}(X) \cup V_1)) = \nu X. (V_2 \cap (\overline{Pre}(K_{\downarrow} X) \cup V_1))
\]
Generalized LCS [LPAR’06]

- Transition: (regular) guards over channel contents + operation
  - more expressive
  - allows emptiness and occurrence tests
  - permits easier products with Büchi automata

Reachability

\[ \text{Pre}^*(A) = \mu X. A \cup \text{Pre}(X) = \mu X. A \cup \text{Pre}(\uparrow X) \]

Safety properties

\[ \forall (V_1 R V_2) = \nu X. (V_2 \cap (\text{Pre}(X) \cup V_1)) = \nu X. (V_2 \cap (\text{Pre}(K \downarrow X) \cup V_1)) \]

Proposition

Reachability and safety properties are decidable for Generalized LCS.
Outline

Introduction

Regular Model Checking for LCS
  Syntactic criterion

Probabilistic LCS
  The Markov chain model
  Attractors
  Qualitative verification

Nondeterministic and Probabilistic LCS
  The Markov decision process model
  Undecidability results
  Finite-memory schedulers

Implementation

Conclusion
Message losses are probabilistic according to a fault rate $\lambda$.

**Definition: Probabilistic LCS**

A *Probabilistic LCS* is an LCS equipped with
- positive weights on rules, and
- a constant probability $\lambda \in (0, 1)$.

$$\lambda = .01$$
Local fault model  [Abdulla,Rabinovich 03] [FOSSACS’03]

- Rules are chosen probabilistically according to weights.
- Message losses are independent events.

![Diagram showing state transitions and probabilities](image)
Attractors in Markov Chains

**Definition: Attractor**

An attractor $W$ in a Markov Chain $M$ is a set of states that is visited almost surely from any starting state.

$$\forall \sigma_0, \ P(\sigma_0 \models \diamond W) = 1$$

**Sufficient criterion [IPL’06]**

Given $(S_i)_{i \in \mathbb{N}}$ a partition of the state-space, if the level has a uniform tendency to decrease (*), then $S_0$ is an attractor.

$$\exists \delta > 0, \ \forall i > 0 \ \forall s \in S_i, \ E(s) \leq i - \delta$$

(*)

Corollary: The finite set of configurations with empty channels is an attractor for Probabilistic LCS.
Decidability result

Almost-sure model checking:

**Given** a Probabilistic LCS $\mathcal{P}$, a configuration $\sigma_0$, an LTL formula $\phi$ over regions,

**Question** does $\mathbb{P}(\sigma_0 \models \phi) = 1$?

**Theorem** [FOSSACS’03] [IC’05]

Almost-sure model checking is decidable whatever $\lambda \in (0, 1)$.

Extensions: Probabilistic LCS with duplication, corruption or insertion errors.
Outline

Introduction

Regular Model Checking for LCS
  Syntactic criterion

Probabilistic LCS
  The Markov chain model
  Attractors
  Qualitative verification

Nondeterministic and Probabilistic LCS
  The Markov decision process model
  Undecidability results
  Finite-memory schedulers

Implementation

Conclusion
Motivations

In the purely probabilistic setting, rules having non-zero weights mean:

- the rules will be selected in a strongly fair way
- any sequence of rules will be selected in a strongly fair way
- ratio between selection rates tend to some limit
Motivations

In the purely probabilistic setting, rules having non-zero weights mean:

▷ the rules will be selected in a strongly fair way
▷ any sequence of rules will be selected in a strongly fair way
▷ ratio between selection rates tend to some limit

Nondeterminism in rules comes from:

▷ arbitrary interleaving of asynchronous components
▷ abstraction of complex programs
▷ early designs
The Markov decision process model

- Choices between enabled actions are nondeterministic.
- Message losses are probabilistic.

Each nondeterministic choice is followed by probabilistic losses.
- 1 and 1/2-player game
The Markov decision process model

- Choices between enabled actions are nondeterministic.
- Message losses are probabilistic.

Each nondeterministic choice is followed by probabilistic losses.
- 1 and 1/2-player game

Scheduler: makes the nondeterministic choices, based on history
Qualitative questions

Markov Chains
\[ \mathbb{P}(\phi) = 1 \]?

Nondeterministic Systems
\[ \forall \pi, \pi \models \phi \]?
Qualitative questions

Markov Chains
\[ \mathbb{P}(\phi) = 1 ? \]

Nondeterministic Systems
\[ \forall \pi, \pi \models \phi ? \]

Markov Decision Processes
\[ \forall \mathcal{U}, \mathbb{P}(\mathcal{U} \models \phi) = 1 ? \]
Qualitative questions

Markov Chains
\[ P(\phi) = 1 \, ? \]
\[ P(\phi) > 0 \, ? \]

Nondeterministic Systems
\[ \forall \pi, \pi \models \phi \, ? \]

Markov Decision Processes
\[ \forall U, P(U \models \phi) = 1 \, ? \]
\[ \forall U, P(U \models \phi) > 0 \, ? \]
Undecidability results

Repeated reachability with positive probability:

Given region $A$, configuration $\sigma$

Question Does there exist a scheduler $\mathcal{U}$ with $\mathbb{P}_\mathcal{U}(\sigma \models \Box \Diamond A) > 0$?

Proposition The repeated reachability problem with positive probability is undecidable.
Undecidability results

Repeated reachability with positive probability:

**Given** region $A$, configuration $\sigma$

**Question** Does there exist a scheduler $U$ with

$$\mathbb{P}_U(\sigma \models \square\Diamond A) > 0?$$

**Proposition**
The repeated reachability problem with positive probability is undecidable.

**Theorem**
Qualitative verification of $\omega$-regular properties is undecidable.
Proof idea

\[ s_0 \] any

\[ S \]
Proof idea

\[ S' : \]

- out
- in
- retry
- success
- fail

Cleaning gadget

\[ s_0 \]

\[ S \]

\[ \text{any} \]
Proof idea

Does there exist a scheduler $U$ that makes $S'$ visit success infinitely often with $> 0$ probability?
Proof idea

Does there exist a scheduler $U$ that makes $S'$ visit `success` infinitely often with $>0$ probability?

Yes iff $S$ is unbounded.
**Finite-memory schedulers**

**Definition: Finite-memory scheduler**

A scheduler is *finite-memory* if it picks the rules based on:

1. current configuration, and
2. information from the history via a finite automaton.
Finite-memory schedulers

**Definition: Finite-memory scheduler**

A scheduler is *finite-memory* if it picks the rules based on:
(1) current configuration, and
(2) information from the history via a finite automaton.

**Theorem** [ACMToCL’05]

Qualitative verification of \( \omega \)-regular properties under finite-memory schedulers is decidable.
Finite-memory schedulers

**Definition: Finite-memory scheduler**

A scheduler is *finite-memory* if it picks the rules based on:
1. current configuration, and
2. information from the history via a finite automaton.

**Theorem [ACMToCL’05]**

Qualitative verification of $\omega$-regular properties under finite-memory schedulers is decidable.

**Scheme of the proof:**

- show decidability for Streett properties $\bigwedge_{i=1}^{n} \square \Diamond A_i \rightarrow \square \Diamond B_i$
## Finite-memory schedulers

### Definition: Finite-memory scheduler

A scheduler is *finite-memory* if it picks the rules based on:

1. current configuration, and
2. information from the history via a finite automaton.

### Theorem [ACMToCL’05]

Qualitative verification of $\omega$-regular properties under finite-memory schedulers is decidable.

### Scheme of the proof:

- show decidability for Streett properties $\bigwedge_{i=1}^{n} \Box \Diamond A_i \rightarrow \Box \Diamond B_i$
- build the product of $\mathcal{N}$ with a DSA for $\phi$, and solve a Streett question.
Decidability proofs

The four variants in all cases ($\diamond A$, $\Box \diamond A$, etc...) rely on:
Decidability proofs

The four variants in all cases ($\diamondsuit A$, $\Box \diamondsuit A$, etc...) rely on:
- characterization of the set of “winning configurations” via a $\mu$-calculus term
Decidability proofs

The four variants in all cases (◊A, □◊A, etc...) rely on:

- characterization of the set of “winning configurations” via a μ-calculus term
- application of the general theorem on guarded terms to show the convergence
Decidability proofs

The four variants in all cases ($\diamond A$, $\Box \diamond A$, etc...) rely on:

- characterization of the set of “winning configurations” via a $\mu$-calculus term
- application of the general theorem on guarded terms to show the convergence

Example: $\diamond A$ almost surely

$$\exists \mathcal{U} \text{ finite-memory, } \mathbb{P}_\mathcal{U}(\sigma \models \diamond A) = 1 \text{ iff } \sigma \in \nu X.\hat{\text{Pre}}_X^*(A).$$
Decidability proofs

The four variants in all cases (◊A, □◊A, etc...) rely on:

- characterization of the set of “winning configurations” via a μ-calculus term
- application of the general theorem on guarded terms to show the convergence

Example: ◊A almost surely

∃U finite-memory, \( \mathbb{P}_U(\sigma \models ◊A) = 1 \) iff \( \sigma \in \nu X.\widehat{\text{Pre}^*_{K\downarrow X}}(A) \).
Decidability proofs

The four variants in all cases ($\diamond A$, $\square \diamond A$, etc...) rely on:

- characterization of the set of “winning configurations” via a $\mu$-calculus term
- application of the general theorem on guarded terms to show the convergence

Example: $\diamond A$ almost surely

$$\exists U \text{ finite-memory}, \quad P_U(\sigma \models \diamond A) = 1 \iff \sigma \in \nu X \hat{P}re^*_K X (A).$$

Example: $\phi = \bigwedge_{i=1}^n \square \diamond A_i$ with positive probability

$$\exists U \text{ finite-memory}, \quad P_U(\sigma \models \phi) > 0 \iff \sigma \in \mu Y. Pre(Y) \lor \left[ \nu X \hat{P}re^+ X (A_1 \land (\hat{P}re^+ X (A_2) \land (\cdots \hat{P}re^+ X (A_n)))) \right].$$
Fair schedulers

**Goal:** verification under fairness assumptions.

Fairness assumption: $\mathcal{F} \in 2^{2^\Delta}$ set of sets of transition rules.
Fair schedulers

**Goal**: verification under *fairness assumptions*.

Fairness assumption: $\mathcal{F} \in 2^{2^\Delta}$ set of sets of transition rules.

**Definition**

A scheduler is $\mathcal{F}$-fair if almost all paths it generates are fair.
Fair schedulers

**Goal:** verification under fairness assumptions.

Fairness assumption: $\mathcal{F} \subseteq 2^{2^\Delta}$ set of sets of transition rules.

**Definition**

A scheduler is $\mathcal{F}$-fair if almost all paths it generates are fair.

**Theorem [FORTE’06]**

Verification of $\omega$-regular properties against finite-memory fair schedulers is decidable.
Outline

Introduction

Regular Model Checking for LCS
  Syntactic criterion

Probabilistic LCS
  The Markov chain model
  Attractors
  Qualitative verification

Nondeterministic and Probabilistic LCS
  The Markov decision process model
  Undecidability results
  Finite-memory schedulers

Implementation

Conclusion
Implementation in OCaml

**Restricted regions:** upward closure closed under prefix and complementation

**Example**

1. \((L0R4, \uparrow \varepsilon, \uparrow \varepsilon)\): control region \((L0R4)\)
2. \((L0R4, \uparrow \varepsilon, a_0 \uparrow \varepsilon)\): configurations where \(L0 \xrightarrow{c_2?a_0} L1\) is enabled
3. \((L0R4, \uparrow d_0a_1 - d_0 \uparrow a_1a_0, \uparrow \varepsilon) - \sum_{m \in M} m \uparrow \varepsilon\)
Implementation in OCaml

Restricted regions: upward closure closed under prefix and complementation

Example

1. \((L0R4, \uparrow \varepsilon, \uparrow \varepsilon)\): control region \((L0R4)\)
2. \((L0R4, \uparrow \varepsilon, a_0 \uparrow \varepsilon)\): configurations where \(L0 \xrightarrow{c_2 ? a_0} L1\) is enabled
3. \((L0R4, \uparrow d_0 a_1 - d_0 \uparrow a_1 a_0, \uparrow \varepsilon) - \sum_{m \in M} m \uparrow \varepsilon\)

- easy representation in OCaml
- closed under Boolean operators
- allow to check first message in channel
  - emptiness tests
  - rule enabledness
How to encode Pachl’s protocol?

- 36 control locations
- 216 transition rules
How to encode Pachl’s protocol?

- 36 control locations
- 216 transition rules

In order to express fairness, we add a history variable recording the last transition rule fired.
How to encode Pachl’s protocol?

- 144 control locations
- 948 transition rules

In order to express fairness, we add a history variable recording the last transition rule fired.
Case studies

**Goal:** prove progress in the protocol under fairness hypothesis

**Progress:**

\[ \phi = \bigwedge_i \Box \Diamond L_i \land \bigwedge_j \Box \Diamond R_j \]
Case studies

**Goal:** prove progress in the protocol under fairness hypothesis

Progress:

\[ \phi = \bigwedge_i \lozenge \Diamond L_i \land \bigwedge_j \Box \Diamond R_j \]

Fairness assumptions:
- fairness on losses
Case studies

**Goal:** prove progress in the protocol under fairness hypothesis

**Progress:**

\[ \phi = \bigwedge_i \square \Diamond L_i \land \bigwedge_j \square \Diamond R_j \]

**Fairness assumptions:**

- fairness on losses (given by probabilities)
Case studies

**Goal:** prove progress in the protocol under fairness hypothesis

**Progress:**

\[ \phi = \bigwedge_{i} \Box \Diamond L_i \land \bigwedge_{j} \Box \Diamond R_j \]

**Fairness assumptions:**

- fairness on losses (given by probabilities)
- fairness on processes
Case studies

**Goal:** prove progress in the protocol under fairness hypothesis

**Progress:**

\[ \phi = \bigwedge_i \Box \Diamond L_i \land \bigwedge_j \Box \Diamond R_j \]

**Fairness assumptions:**

- fairness on losses (given by probabilities)
- fairness on processes
- fairness on read actions
Liveness analysis (cont.)

- $\mathcal{F} = \mathcal{F}_{\text{read}}$ and $A = After_{\text{left}}$

$\text{Init}$ satisfies the following property:

$$\forall U \text{ } \mathcal{F}\text{-fair}, \mathbb{P}_U(\text{Init} \models \square \diamond A) = 1$$
Liveness analysis (cont.)

- $F = \mathcal{F}_{\text{read}}$ and $A = \text{After}_{\text{left}}$

$\text{Init}$ satisfies the following property:

$$\forall U \; \mathcal{F}\text{-fair}, \; P_U(\text{Init} \models \Box \Diamond A) = 1$$

- $F = \mathcal{F}_{\text{read}}$ and $A = \text{After}_{\text{left-move}}$
Liveness analysis (cont.)

- $\mathcal{F} = \mathcal{F}_{\text{read}}$ and $A = \text{After}_{\text{left}}$

Init satisfies the following property:

$$\forall U \; \mathcal{F}\text{-fair}, \; \mathbb{P}_U(\text{Init} \models \square \Diamond A) = 1$$

- $\mathcal{F} = \mathcal{F}_{\text{read}}$ and $A = \text{After}_{\text{left-move}}$

- $\mathcal{F} = \{F_{\text{read}}, F_{\text{right-read}}\}$ and $A = \text{After}_{\text{left}}$
Liveness analysis (cont.)

- \( \mathcal{F} = \mathcal{F}_{\text{read}} \) and \( A = \text{After}_{\text{left}} \)

Delete: \( \text{Init} \) satisfies the following property:
\[
\forall U \mathcal{F}\text{-fair}, \ P_U(\text{Init} \models \Box \Diamond A) = 1
\]

- \( \mathcal{F} = \mathcal{F}_{\text{read}} \) and \( A = \text{After}_{\text{left-move}} \)
- \( \mathcal{F} = \{ \mathcal{F}_{\text{read}}, \mathcal{F}_{\text{right-read}} \} \) and \( A = \text{After}_{\text{left}} \)

Proposition

Progress is almost sure in Pachl’s protocol assuming fairness.
Outline

Introduction

Regular Model Checking for LCS
   Syntactic criterion

Probabilistic LCS
   The Markov chain model
   Attractors
   Qualitative verification

Nondeterministic and Probabilistic LCS
   The Markov decision process model
   Undecidability results
   Finite-memory schedulers

Implementation

Conclusion
Summary

Probabilistic LCS

- local fault model
- finite attractor
- decidability result for qualitative verification of $\omega$-regular properties

Nondeterministic Probabilistic LCS

- Markov decision process model
- undecidability for full class of schedulers
- decidability for finite-memory schedulers
- implementation of restricted (but expressive enough) framework
Future work

- turn the prototype into a tool
- study games on LCS
- tackle quantitative questions
- consider richer models (with counters, clocks, etc...)
Thank you for your attention!