Stochastic languages

Florent Garnier

April 14, 2009
Probabilistic automaton model was defined by M. Rabin [Rabin63].

- A variant of NFA.
- Express probabilistic transitions between states.
- Define stochastics languages.
- Variant are used in various domain, from learning, pattern recognition, signal processing etc ...
Stochastic languages

Introduction

Probabilistic automaton

Definition

A probabilistic automaton is a tuple \((Q, \Sigma, q_0, F, \delta)\) with:

- \(Q\) is a finite set of states.
- \(\Sigma\) is an alphabet.
- \(q_0\) is the initial state.
- \(F\) is the set of accepting states.
- \(\delta\) is a function from \(Q \times \Sigma \rightarrow \text{Dist}(Q)\).
Probabilistic automaton

Example

- States: 0, 1, 2
- Transitions:
  - From 0: b with probability 1, a with probability 1/2 to state 1, and b,c with probability 1 to state 2.
  - From 1: a, 1/2 to state 0.
  - From 2: b,c with probability 1 to state 0, and a, 1/2 to state 1.
Language of a probabilistic automaton (Rabin)

**Definition (Run of a word)**

Given a word $w = w_1 \ldots w_n$ of $\Sigma^*$, a run of $w$ on $A$ is a sequence $q_1 \ldots q_n$ of $Q$ such that:

- $q_0$ is the initial state of $A$.
- For all $0 \leq i < n$, $\delta(q_i, w_i, q_{i+1}) > 0$.
- If $q_n \in F$ the run is accepting.
Runs of a p-automaton

Example

Runs for the word \textit{aba} are:

\[ \{0 \xrightarrow{a} 1 \xrightarrow{b} 0 \xrightarrow{a} 1, 0 \xrightarrow{a} 2 \xrightarrow{b} 0 \xrightarrow{a} 1, 0 \xrightarrow{a} 2 \xrightarrow{b} 0 \xrightarrow{a} 2 \}. \]

The two first are accepting, whereas the latter is not.
A \( p \)-automaton, associates to each word of \( \Sigma^* \) a probability:
Measure of a word

This value corresponds to the weighted ration between the accepting runs over all the run associated to a run. This value is set by zero if a word has no associated execution.

**Definition (Probability of acceptance)**

Given a word \( w = w_1 \ldots w_n \), we note by:

\[
P(w) := \frac{\sum_{\rho \in \text{Accept}(w)} \prod_{i=1}^{n} \delta(\rho_i, w_i, \rho_{i+1})}{\sum_{\rho \in \text{Run}(w)} \prod_{i=1}^{n} \delta(\rho_i, w_i, \rho_{i+1})}.
\]
Stochastic languages

Definition (Stochastic languages)
The set of recognized by probabilistic automata are called stochastic languages.

Definition (Languages recognized by a P-automaton)
Given a real number $\eta < 1$, a Probabilistic automaton $A$, the language $\mathcal{L}_\eta(A)$ is defined as:

$$\mathcal{L}_\eta(A) := \{ w \in \Sigma^* | P(w) \geq \eta \}$$
Stochastic languages

Introduction

Property

- If $w \in \mathcal{L}_\eta(A)$, then $A$ recognize the word $w$ with probability at least $\eta$.
- The set $\mathcal{L}_\eta(A)$ depends on the threshold $\eta$. 

\[ aca \in \mathcal{L}_{\frac{1}{2}}(A). \]
\[ aca \notin \mathcal{L}_\eta(A) \text{ for all } \eta < \frac{1}{2}. \]
Properties of stochastic languages

Property

1. S.L. contains the set of regular languages.
2. There exists Non regular stochastic languages.
A non regular but stochastic language:

Example

PBA for $L = \{a^{k_1}ba^{k_2}b\ldots a^{k_n}b\ldots s.t \prod_{i=1}^{n}(1 - (1/2)^{k_i}) > \eta\}$
Properties on cut-points

**Definition (Isolated cut-point)**

Let be $A$ a probabilistic automaton. A cut-point $\eta$ is called *isolated* cut-point if there exists $\delta > 0$ such that:

$$|P(w) - \eta| \geq \delta, \quad \forall w \in \Sigma^*.$$ 

**Property**

*If $\eta$ is an isolated cutpoint, then $L_\eta(A)$ is a regular language.*
A $p$-adic language is the set of strings in the alphabet \{0, \ldots, (p - 1)\}, such that:

$$L_\eta(p) := \{0.n_1n_2n_3 \ldots | 0 \leq n_k < p \land 0.n_1n_2n_3 \ldots > \eta\}$$

**Property**

- $L_\eta(p)$ is rational iff $\eta \in \mathbb{Q}$.
- The number of stochastic languages is not countable.
A really powerful model of automaton

**Property**

- If $w \in \mathcal{L}_\eta(\mathcal{P})$, then $P(A(w)) \geq \eta$.
- If $w \notin \mathcal{L}_\eta(\mathcal{P})$, then $P(A(w)) = 0$.

*Stochastic languages are in the RP-complexity class.*

Question: How to test whether a word $w$ belongs to this language with high probability?
The following problem is undecidable: Given $A$ a PFA, and $0 < \eta < 1$, does it exist $w \in \Sigma^*$, such that $P(w) > \eta$?

Deciding language emptiness is mandatory for dedicing:

- Universality.
- Equality of two languages.
- Language inclusion.
How to define a probabilistic bisimulation

Definition

Let $\mathcal{R}$ be an equivalence relation over the set $X$. Two probability spaces $(\Omega_1, \mathcal{F}_1, P_1)$ and $(\Omega_2, \mathcal{F}_2, P_2)$ of $\text{Probs}(X)$ are $\mathcal{R}$–equivalent, written $(\Omega_2, \mathcal{F}_2, P_2) \equiv_{\mathcal{R}} (\Omega_2, \mathcal{F}_2, P_2)$, iff for every class $C$ of $X/\mathcal{R}$

$$\sum_{x \in \Omega_1 \cap C} P_1[x] = \sum_{x \in \Omega_1 \cap C} P_1[x]$$

In other words, $(\Omega_1, \mathcal{F}_1, P_1)$ and $(\Omega_2, \mathcal{F}_2, P_2)$ are $\mathcal{R}$–equivalent if they assign the same probability measure to each equivalence class of $\mathcal{R}$. 

Strong bisimulation

**Definition (Strong bisimulation)**

A strong bisimulation between two simple probabilistic automata $M_1$ and $M_2$ is an equivalence relation $\mathcal{R}$ over $\text{states}(M_1) \cup \text{states}(M_2)$ such that

1. each start state of $M_1$ is related to at least one start state of $M_2$ and vice versa,

2. for each $s_1 \mathcal{R} s_2$ and each transition $s_1 \xrightarrow{a} P_1$ of either $M_1$, $M_2$, there exists a transition $s_2 \xrightarrow{a} P_2$ of either $M_1$, $M_2$ such that $P_1 \equiv_{\mathcal{R}} P_2$. We write $M_1 \simeq M_2$ whenever $\text{acts}(M_1) = \text{acts}(M_2)$ and there is a strong bisimulation between $M_1$ and $M_2$. 

A strong bisimulation between two simple probabilistic automata $M_1$ and $M_2$ is an equivalence relation $\mathcal{R}$ over $\text{states}(M_1) \cup \text{states}(M_2)$ such that

1. each start state of $M_1$ is related to at least one start state of $M_2$.

2. for each $s_1 \mathcal{R} s_2$ and each transition $s_1 \xrightarrow{a} P_1$ of either $M_1$ there exists a transition $s_2 \xrightarrow{a} P_2$ of either $M_2$ such that $P_1 \equiv_{\mathcal{R}} P_2$. We write $M_1 \sqsubseteq M_2$ whenever $\text{acts}(M_1) = \text{acts}(M_2)$ and there is a strong bisimulation between $M_1$ and $M_2$. 
Model Checking in short consists in:

- Modeling a system with a formal language.
- Expressing specification or system properties.
- Proving that every run of the system complies with the specification.

Question: Is it possible to use randomness to improve verification?
Modeling a system

Low level modeling formalism Transition system.

Definition (Transition system)

A transition system is a tuple $\mathcal{K} = (S, I, \mathcal{A}, \delta)$ where:
- $S$ system states
- $I \subseteq S$ initial states
- $\mathcal{A}$ set of actions
- $\delta \subseteq S \times \mathcal{A} \times S$ transition relation

For every $s \in S$ there exists $(a, t) \in S \times \mathcal{A}$ such that $(s, a, t) \in \delta$. 
Semantic of a transition system

**Definition (Run of a transition system)**

A run of $\mathcal{K}$ is an infinite sequence $\rho = s_0 \ldots s_n \ldots$ of states $s_i \in S$ such that:

- $s_0 \in I$
- For all $i \in \mathbb{N}$, $(s_i, a_i, s_{i+1}) \in \delta$ for some $a_i \in A$
High(er) level modeling formalism

- Kripke Structures
- Process Algebra
- Petri Nets
- Communicating agents
- Automata.
- etc ...
Specifying properties using logics

- Linear logic: PTL, LTL.
  - Fairness, mutual exclusion, eventual access to critical section etc ...
- Branching Logic: CTL
  - Combine temporal modalities and quantification over paths.
- Logic subsuming Linear and Branching logic: CTL*, μ-calcul.
The **LTL** logic express time dependent properties of system runs. Evaluated over infinite sequences of labels.

<table>
<thead>
<tr>
<th>Type</th>
<th>Formula</th>
<th>$\rho \models \phi$ iff ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic</td>
<td>$p \in AP$</td>
<td>$p$ holds for $\rho</td>
</tr>
<tr>
<td>Boolean</td>
<td>$\neg \phi$</td>
<td>$\rho \not\models \phi$</td>
</tr>
<tr>
<td></td>
<td>$\phi \lor \psi$</td>
<td>$\rho \models \phi$ or $\rho \models \psi$</td>
</tr>
<tr>
<td>Temporal</td>
<td>$X\phi$</td>
<td>$\rho</td>
</tr>
<tr>
<td></td>
<td>$F\phi$</td>
<td>$\rho</td>
</tr>
<tr>
<td></td>
<td>$G\phi$</td>
<td>$\rho</td>
</tr>
<tr>
<td></td>
<td>$\phi U \psi$</td>
<td>There is $i \in \mathbb{N}$ s.t. $\rho</td>
</tr>
<tr>
<td></td>
<td>$\phi W \psi$</td>
<td>$\rho \models \phi U \psi$ or $\rho \models G\phi$</td>
</tr>
</tbody>
</table>
**Stochastic languages**

**Introduction**

**Specifying**

**LTL: Exemples**

**Exemple**

- **invariants** $GP$
  - $G \neg (\text{crit}_1 \land \text{crit}_2)$ mutual exclusion
  - $G(\text{preset}_1 \lor \ldots \lor \text{preset}_n)$ deadlock freedom

- **Response, recurrence** $G(P \Rightarrow FQ)$
  - $G(\text{try}_1 \Rightarrow F\text{crit}_1)$ eventual access to critical section
  - $GF \neg \text{crit}_1$ no starvation in critical section

- **Reactivity, Steet** $GFP \Rightarrow GFQ$
  - $GF(\text{try}_1 \land \neg \text{crit}_2) \Rightarrow G\text{Fcrit}_1$ strong fairness

- **Precedence** $G(P_1 W \ldots WP_n)$
  - $G(\text{try}_1 \land \neg \text{try}_2 \Rightarrow \neg \text{crit}_2 W\text{crit}_2 W \neg \text{crit}_2 W\text{crit}_1)$
  - 1-bounded overtaking
How to check that $\mathcal{K} \models \phi$?

- LTL formulae can be represented by $\omega$-automata.
- $\omega$-automata are expressive enough to model transition systems.
- $\omega$-automata are finite automata that accept infinite length language, that can model program runs and runs constraints.

Let’s now define and list $\omega$-automata properties ...
Finite automata on \(\omega\)-words

**Definition (Büchi-automata)**

A \(\omega\)-automaton is a tuple \(B = (Q, I, \delta, F)\):
- \(Q\): finite set of states
- \(I \subseteq Q\): initial states
- \(\delta \subseteq Q \times \Sigma \times Q\): transition relation
- \(F \subseteq Q\): accepting states

**Definition (Run of \(B\) on \(\omega\)-words \(a_0a_1\ldots \in \Sigma^\omega\))**

- **sequence**: \(q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \ldots\)
- **initialisation**: \(q_0 \in I\)
- **consecution**: \((q_i, a_i, q_{i+1}) \in \delta\) for all \(i \in \mathbb{N}\)
- **accepting**: \(q_i \in F\) for infinitely many \(i \in \mathbb{N}\)
ω-regular languages

**Definition (ω-language defined by $\mathcal{B}$)**

$L = \{ w \in \Sigma^\omega \mid \mathcal{B} \text{ has some accepting run on } w \}$

**Definition**

ω-regular languages class of $\omega$ languages definable by Büchi automata.
Büchi automata: basic properties

Property (Decidability emptiness problem)

\[ \mathcal{L} \neq \emptyset \text{ iff exists } q_0 \in I, q \in F \text{ such that } q_0 \xrightarrow{\Sigma^*} q \xrightarrow{\Sigma^+} q. \]

Complexity linear in \(|Q|\).

Property (Closure properties)

- union
- intersection
- complement, difficult construction \(O(2^{n\log n})\) states
- projection
Exemple of Büchi automata

Property

Deterministic Büchi automata are strictly less expressive than Non deterministic Büchi automata.
From LTL to Büchi automata

Basic insight

- Given $\phi$ a LTL formula, let note $\mathcal{L}$ the set of sequences of labels satisfying $\phi$.
- Construct automaton $\phi$ that accepts $\mathcal{L}(\phi)$.

Idea of construction

- **states**: sets of "subformulas of $\phi$ promised to be true
- **initial states**: states containing $\phi$
- **transition relation**: Ensures satisfaction of non-temporal formulas using recursion laws:
  
  \[ G\phi \equiv \phi \land XG\phi \]
  \[ F\phi \equiv \phi \lor XF\phi \]
  \[ \phi U\psi \equiv \psi \lor (\phi \land X(\phi U\psi)) \]

- **accepting states**: defined from $F\phi$ or $\phi U\psi$. 
Problem: Given $\mathcal{K}$ and $\phi$, decide whether $\mathcal{K} \models \phi$

Automata-theoretic solution

- Consider $\mathcal{K}$ as a $\omega$-automaton with all states final
- Define $(L)(\mathcal{K}) =$ set of computations of $\mathcal{K}$
- The following assertions are equivalent:

\[
\begin{align*}
\mathcal{K} & \models \phi \\
L(\mathcal{K}) & \subseteq L(\phi) \\
L(\mathcal{K}) \cap L(\neg \phi) & = \emptyset \\
L(\mathcal{K}) \times B_{\neg \phi} & = \emptyset
\end{align*}
\]

Complexity $O(|\mathcal{K}| \times |B_{\neg \phi}|) = O(|\mathcal{K}| \times 2^{|\phi|})$
The paper [BG] describes a model of PBA –resp. uPBA that accepts a subset of $\omega$-regular language –resp. $\omega$-regular languages.

The size of uPBA can be exponentially more concise than NSA (Strong fairness).

Emptiness is not decidable for PBA.

Emptiness is decidable for uPBA.
A model of probabilistic automata

**Definition (Probabilistic Büchi Automata (PBA))**

A PBA over the alphabet $\Sigma$ is a tuple $\mathcal{B} = (Q, \delta, \mu, F)$ where:

- $Q$ is a finite set of states.
- $\delta : Q \times \Sigma \times Q \rightarrow [0, 1]$ is the transition probabilities function.
- $\mu$ is the initial distribution, $\mu(Q) = 1$.
- $F \subseteq Q$ is the set of accepting states.
Exemple of PBA

PBA for \((a + b)^* a^\omega\)
Accepted language of a PBA

Definition (Accepting run)
An infinite run $\pi = s_0, \ldots s_n \ldots$ of $B$ is accepting if $\inf(\pi) \cap F \neq \emptyset$.

Definition (Accepted language of a PBA)
If one note $Pr(\rho) = P(\pi : \pi$ is an accepting run for $\rho)$, $L(B)$ consists of all words $\rho \in \Sigma^\omega$ s.t $Pr(\rho) > 0$.
Exemple of PBA

PBA for \((ab + ac)^*(ab)^\omega\)
Expressiveness of PBA

**Theorem**

The class of languages that can be accepted by a PBA strictly contains the class of $\omega$-regular languages.

**Lemma**

There exists PBA that accepts non $\omega$-regular languages.
Non $\omega$-regular accepting PBA

PBA for $L = \{a^{k_1}b a^{k_2}b a^{k_3}b \ldots s.t \prod_{i \geq 1} (1 - (1/2)^{k_i}) > 0\}$
Deciding emptiness of PBA

**Theorem**

*The emptiness of PBA is undecidable* [BBG08].

However, this problem is decidable for a subclass of PBA.
End components

Definition (End components)

Let $B = (Q, \delta, \mu, F)$ be a PBA. An end component of $B$ is a pair $(P, A)$ where:

- $P \subseteq Q$
- $A : P \rightarrow 2^\Sigma$
- $\sum_{q \in P} \delta(p, a, q) = 1$ for $a \in A(p)$
- The graph $(P, A)$ is strongly connected.

$(P, A)$ is **accepting** if $P \cap F \neq \emptyset$. 
Uniform PBA

Definition (Uniform PBA)

A PBA $\mathcal{B}$ is called uniform if there exists $\theta > 0$ s.t for all $(P, A)$ accepting end component, all $p \in P$ and all finite word $w \in \Sigma^*$ one of the following condition holds:

1. $\Pr_{(P,A)}(p \xrightarrow{w}) \leq \theta$
2. $\Pr_{(P,A)}(p \xrightarrow{w}) = 1 \land \Pr_{(P,A)}(p \xrightarrow{w} q) \geq \theta$
Acceptance on uPBA

Definition

Let be $B$ an uniform PBA and $\rho = a_0 \ldots \in \Sigma^\omega$. Then $\rho \in L(B)$ iff there exists $(P, A)$ accepting end component and a strongly prob fair run $\pi = q_0 \ldots$ for $\rho$ s.t $\inf(\pi) = (P, A)$ and $q_I$ s.t

$$Pr_{(P,A)}(q_I \xrightarrow{w} ) = 1$$
Properties of uPBA

Theorem

*uPBA exactly accept the ω- regular languages.*

Proof.

For any uPBA \( P \) there exists a NSA \( A \) such that \( \mathcal{L}(P) = \mathcal{L}(A) \), with \( |A| = O(\exp(|P|)) \).
Properties of uPBA

Theorem

*The emptiness problem for uPBA is decidable (NP-hard).*

Proof.

The transformation uPBA → NSA is polynomial. Deciding emptiness for NSA is NP-hard.
The uniform PBA $uPBA$ can be exp better than NSA.

There exists languages $L_n$ which are accepted by $uPBA$ with $2n$ states, while any NSA for $L_n$ has at least $\frac{2^n}{n}$.