Probabilistic rewrite strategies. Applications to ELAN

Olivier Bournez and Claude Kirchner

LORIA & INRIA,
615 rue du Jardin Botanique, BP 101,
54602 Villers-lès-Nancy Cedex, Nancy, France.
{Olivier.Bournez,Claude.Kirchner}@loria.fr

Abstract. Recently rule based languages focussed on the use of rewriting as a modeling tool which results in making specifications executable. To extend the modeling capabilities of rule based languages, we explore the possibility of making the rule applications subject to probabilistic choices.
We propose an extension of the ELAN strategy language to deal with randomized systems. We argue through several examples that we propose indeed a natural setting to model systems with randomized choices. This leads us to interesting new problems, and we address the generalization of the usual concepts in abstract reduction systems to randomized systems.

1 Introduction

Term rewriting has been developed since the last thirty years, leading to a deep and solid corpus of knowledge about the rewrite relation induced by a set of rewrite rules. More recently, rule based languages focussed on the use of rewriting as a modeling tool, which results in making the out-coming specification executable in a very efficient way [11]. Such languages enlighten the fundamental role of rewrite strategies, either for computation or for deduction. In the ELAN language, the notion of rule and strategy are both first class and this is backed-up by the concept of rewriting calculus [5]. In this framework, that generalizes the lambda-calculus, the basic notions are those of rewrite rule and of rule application. To extend the modeling capabilities of the calculus, we explore in this work the possibility of making the rule application subject to probabilistic choices.

Since the probabilistic firing of a rewrite rule can be seen as a specific kind of strategy, we introduce in the first part of this work a new notion of strategy. From a practical point of view, we added to ELAN the notion of *probabilistic choice* strategy, permitting us to fire under some probability a given set of rules. It is in particular a fundamental design choice to manage probabilities at the level of strategies instead of at the level of the rules themself where the flexibility and the semantics would have been then less clear.

This leads us to interesting new capabilities and problems that we are addressing in this paper.

The combination of the concept of explicit rule application and of probabilistic choice has many applications. In particular for probabilistic data bases [21], probabilistic agents [6], genetic algorithms [9], randomized algorithms [15], randomized proof procedures [14], ...etc. Therefore, the notion of probabilistic strategy is quite useful as a modeling tool. What we address in the second part of this paper is a first step in the understanding of such strategies in the rewriting context.

Indeed we focus on the study of abstract reduction systems extensions under probabilistic choice. For example, we can define a notion of almost-sure termination as the property that the set of infinite derivations of an abstract reduction system is of null measure. A typical situation is the rewrite system $a \to a, a \to b$ where the rewrite strategy consists in applying each rule with probability 0.5. This system is terminating with probability one but clearly not terminating.

Similarly, probabilistic notions for confluence or other usual notions of classical abstract reduction systems can be defined. The design of tools to check these properties at the level of a rewrite *relation* is a challenge and will strengthen the use of the concept as a specification tool. This paper aims at putting a first stone towards these directions.

We presented first ideas about such an approach to control rule firing together with a first prototype in [1]. A different point of view has been developed recently to extend the Constraint Handling Rule process with probabilistic capabilities put on the rewrite rules themselves [8,16,18,17,19]. This is, to the best of our knowledge, the only attempts to formalize probabilistic transitions using rule based languages.

In Section 2 we introduce through some simple examples the strategy operator that we added to ELAN strategy language to deal with probabilistic systems. In Section 3, we argue through several examples that this is indeed a natural setting for specifying systems with probabilistic choices. In Section 4, we discuss on the operational semantic of the proposed strategy operator through a discussion on the way this operator is implemented in the ELAN prototype. This leads us to important questions about the generalization of usual rewriting concepts such as confluence and termination that we discuss for probabilistic abstract reduction systems in Section 5.

2 A general probabilistic rewrite strategy operator.

Let us first consider the simple example of euro coin flipping[1]. It could be modeled using the following rules:

```
[h] x => head
[t] x => tail
```

[1] See `http://www.inference.phy.cam.ac.uk/mackay/abstracts/euro.html` for a summary of the recent attention devoted to the potential bias of the euro coin (thanks to one of the referees).

where `h` and `t` are the names, also call labels of the rules, `head, tail` are just constant function symbols and `x` is a variable.

In order to express the equiprobability of the two sides of a euro coin, we write

```
equi => PC(h:0.5, t:0.5)
```

which means that the two rules fire with the same probability and that the application of the strategy `equi` to `flip` (denoted `(equi flip)`) reduces either to `tail` or to `head` with the same probability 0.5.

If euro coins were biased, we would write something like:

```
bias => PC(h:0.4, t:0.6)
```

Then, we would have with probability 0.4 the derivation

```
(bias flip) => (PC(h:0.4, t:0.6) flip)
            => (h flip)
            => head
```

and, with probability 0.6 the derivation

```
(bias flip) => (PC(h:0.4, t:0.6) flip)
            => (t flip)
            => tail
```

But actually a strategy operator that makes choices with fixed probabilities is not sufficient to model all games, since probabilities often also depend on parameters.

Suppose for example, that in addition to bias problems there were no harmonization between the countries of the euro zone. Then, the bias would depend on the country and we would need somehow a function `probaH`, from countries to numbers in the interval $[0, 1]$ to represent the bias.

It could be natural to model this latter function as a strategy itself, mapping (normalizing) terms representing countries to real numbers in the interval $[0, 1]$.

This could be done in ELAN using something like:

```
[probaH] france  => 0.49
[probaH] belgium => 0.50
...
```

The `bias` operator would then be simply described by:

```
bias   => PC(h:probaH, t:1 - probaH)
```

and we would write `(bias luxembourg)` to simulate Luxembourg euro flipping.

Hence, we actually introduce the following strategy operator: for strategies $p_1, \ldots, p_n$ mapping terms (which sort is un-important here) to terms of the sort of real numbers so that on any term t, $\Sigma_i(p_i\ t) = 1$, and when $s_1, \ldots, s_n$ are other strategies which application is subject to a certain probability law, $PC(s_1 : p_1, \ldots, s_n : p_n)$ is the strategy that consists, applied on some term t, in choosing an $i \in \{1, \ldots, n\}$ with probability $(p_i\ t)$, and then returning $(s_i\ t)$, the application of strategy s_i on the term t.

Before formally describing this strategy combinator, let us play with more examples.

3 Examples

We now develop several examples to argue that the PC operator is indeed a natural setting for modeling probabilistic systems. We choose first an example that requires probabilistic choices with fixed probabilities. The second example requires probabilistic choices with run-time varying probabilities. We choose as third example to take a simple algorithm of the book [15] to exemplify how easy it is to transform any randomized algorithm into a ELAN program using the previous operator.

Example 1. Suppose we want to model the following game between two players. Initially, two players have M euros ($M \geq 2$). An unbiased coin is flipped. If it falls on head, player one wins one euro from player 2, otherwise player 2 wins two euros from player 1. The game stops when one of the two players is ruined.

This can be modeled as follows.

```
[h] game(M1,M2) => game(M1+1,M2-1)
                if M2>0
[h] game(M1,0)  => player-1-wins
[t] game(M1,M2) => game(M1-2,M1+2)
                if M1>1
[t] game(0,M2)  => player-2-wins
[t] game(1,M2)  => player-2-wins
```

The strategy modeling the repetition of coin flipping is:

```
play => repeat(PC(h:0.5, t:0.5))
```

Considering the derivations of `(play game(M,M))`, we get a modeling of our game.

This is clearly not a terminating system, since there are infinite derivations starting from `(play game(M,M))`. However, according to probability theory, the probability of such an infinite derivation is null: with probability 1, a derivation terminates. So we can say, as we will develop in Section 5, that this system is *almost-surely terminating* (but not terminating).

Of course we must also have a picking balls example:

Example 2. Consider a urn with N green balls and two yellow balls. We pick a ball until we get a yellow one. We loose if we pick all the green balls.

```
[pickgreen]  urn(N) => urn(N-1)
                    if N>1
[pickgreen]  urn(1) => loose
[pickyellow] urn(N) => win
```

Transition probabilities now depend on time, i.e. on terms. If balls are picked equiprobably, then the probability of picking a green ball should be $N/(N+2)$.

So we write:

```
play => repeat(PC(pickgreen: p, pickyellow: 1-p))
```

where p is itself a strategy.

```
[p] urn(N) => N / (N+2)
```

We could add many other examples to argue that this setting is quite natural for prototyping probabilistic games, random walks, or randomized algorithms.

To argue that randomized algorithms can easily been turned into a ELAN program, we take the first example of the book [15]. This is a sorting algorithm in expected time $O(2nH_n) = O(n \log n)$, where H_n is the nth Harmonic number [15].

Example 3. We need first to choose an element uniformly in a list L. This is obtained as the application of the probabilistic strategy `any` to `L` with:

```
[thisone]  y.L => y
[otherone] y.L => L

[pthisone]  L => 1/length(L)
[potherone] L => (length(L)-1)/length(L)

any => repeat*(PC(thisone: pthisone, otherone: potherone))
```

And now, we sort as follows:

```
[] sort(L) => sort(subset-lower(L,y)).y.sort(subset-greater(L,y))
          where y:= (any L)
[] sort(nil) => nil

[] subset-lower(z.L,y) => z.subset-lower(L,y)      if z < y
[] subset-lower(z.L,y) => subset-lower(L,y)        if z >= y
[] subset-lower(nil,y) => nil

[] subset-greater(z.L,y) => z.subset-greater(L,y)  if z > y
[] subset-greater(z.L,y) => subset-greater(L,y)    if z <= y
[] subset-greater(nil,y) => nil
```

4 Operational semantics

The previous `PC` operator is available in the ELAN system and the previous examples can be check in the current prototype.

We now present the operational semantic of the probabilistic choice strategy. This is easy to get from following discussion on its implementation, using the background semantics of the ELAN language [3,5].

Actually, the `PC` operators are implemented in ELAN itself, using the classical trick of imperative programming to simulate any probabilistic distribution

using a uniform one [13]. Concretely, we just added to ELAN a built-in operator `uniform-random-generator()` that returns a real number in interval $[0, 1]$ with uniform distribution.

Now, when the p_i are strategies (possibly constant) that, on any term t, evaluate to a positive (or null) real number such that $\sum_{i=1}^{n}(p_i\ t) = 1$, then the application of the strategy $PC(s_1 : p_1, \ldots, s_n : p_n)$ on a term t is defined as follows:

$$[\ ]\ (PC(s_1 : p_1, \ldots, s_n : p_n)\ t) \Rightarrow (\texttt{choose}(y, s_1 : (p_1\ t), \ldots, s_n : (p_n\ t))\ t) \quad \texttt{where}\ y := \texttt{uniform-random-generator}()$$

With, for reals $\xi_i \in [0, 1]$ (that result from the evaluation of p_i on t) such that $\sum_{i=1}^{n} \xi_i = 1$:

$$\begin{aligned}
[\ ]\ \texttt{choose}(y, \tau_1 : \xi_1, \ldots, \tau_n : \xi_n) &\Rightarrow \tau_1 \quad \texttt{if}\ y \leq \xi_1 \\
[\ ]\ \texttt{choose}(y, \tau_1 : \xi_1, \ldots, \tau_n : \xi_n) &\Rightarrow \tau_2 \quad \texttt{if}\ \xi_1 < y \leq \xi_1 + \xi_2 \\
&\ldots \\
[\ ]\ \texttt{choose}(y, \tau_1 : \xi_1, \ldots, \tau_n : \xi_n) &\Rightarrow \tau_n \quad \texttt{if}\ \xi_1 + \xi_2 + \ldots + \xi_{n-1} < y
\end{aligned}$$

This ELAN code simulates a non-uniform distribution using a uniform one.

For example, we have the derivations

```
(bias france) => (PC(h : probaH, t:1 - probaH) france)
              => (choose(0.7, h: (probaH france),
                              t: ((1- probaH) france)) france)
              => (choose(0.7, h: 0.49, t: 0,51) france)
              => (t france)
              => tail
```

and

```
(bias france) => (PC(h: probaH, t:1 - probaH) france)
              => (choose(0.25, h: (probaH france),
                               t: ((1- probaH) france)) france)
              => (choose(0.25, h: 0.49, t: 0,51) france)
              => (h france)
              => head
```

5 Probabilistic Abstract Reduction Systems

Dealing with probabilistic systems gives rise to interesting new problems. For example, what are the generalizations of the classical notions of rewriting such as confluence, termination? Which results still hold, and which can be extended

to that setting? These are the kind of problems that we address in this section by studying the generalization of the classical results about abstract reduction systems to probabilistic ones.

In a first step, we are not dealing with rewriting (ad posteriori nor with conditional rewriting, nor rewriting controlled by strategies) but with reduction systems. However, we claim that this study can give some clues to understand what probabilistic rewriting could be.

Observe also that we took the approach of starting from results from classical abstract reduction systems and studying whether they are still valid in that setting. This is a priori different from considering probabilistic abstract reduction systems by themselves.

We first come back to the classical setting (see for example [2,12]). An *abstract reduction system (ARS)* is $\mathcal{A} = (A, \rightarrow)$ consisting of a set A and a binary relation $\rightarrow \subset A \times A$ on A . A *derivation* is a finite, or infinite sequence $\pi = \pi_0 \rightarrow \pi_1 \ldots \rightarrow \pi_n$ with $(\pi_i, \pi_{i+1}) \in \rightarrow$ for all i.

$\rightarrow^n$ denotes the n-step composition of $\rightarrow$: $\rightarrow^0$ is the identity, and $\rightarrow^{n+1} = \rightarrow^n \circ \rightarrow$. $\rightarrow^{\leq n}$ denotes less-than-n-step composition: $\rightarrow^{\leq n} = \bigcup_{0 \leq i \leq n} \rightarrow^i$. The reflexive transitive closure of a relation $\rightarrow$ is denoted by $\rightarrow^*$ and its symmetric closure is denoted by $\leftrightarrow$.

The reduction relation $\rightarrow$ is said *locally confluent (LC)* if $b \leftarrow a \rightarrow c \Rightarrow \exists d\ b \rightarrow^* d \wedge c \rightarrow^* d$. It is said *confluent (C)* if $b \leftarrow^* a \rightarrow^* c \Rightarrow \exists d\ b \rightarrow^* d \wedge c \rightarrow^* d$.

The following results are well-known.

Proposition 1 (Classical Setting [2,12]). *The following are equivalent:*

1. $\rightarrow$ *is confluent (C)*
2. $\rightarrow^*$ *is locally confluent (LC)*
3. $\rightarrow^*$ *is confluent (C)*
4. *the relation is semi-confluent:* $b \leftarrow a \rightarrow^* c \Rightarrow \exists d\ b \rightarrow^* d \leftarrow^* c$.
5. *the relation is Church Rosser:* $a \leftrightarrow^* b \Rightarrow \exists c\ a \rightarrow^* c \leftarrow^* b$.

Let $\mathcal{A}$ be an ARS. $a \in A$ is said to be in *normal form* if there is no b with $a \rightarrow b$. We say that a *has a normal form (hnf)* if there is a b in normal form with $a \rightarrow^* b$. $a, b \in A$ are said to be *convertible* if $a \leftrightarrow^* b$.

The reduction relation $\rightarrow$, or $\mathcal{A}$, is *normalizing (N)* if every $a \in A$ has a normal form. It is *terminating (T)* if there is no infinite chains $\pi_0 \rightarrow \pi_1 \ldots \rightarrow \pi_n \ldots$.

It has the *unique normal form property (UN)* if for all $a, b \in A$, if a and b are convertible and in normal form, then a and b are identical. It has the *normal form property (NF)* if for all $a, b \in A$, if a is in normal form and convertible with b, then $b \rightarrow^* a$.

It is *inductive (Ind)* if every reduction sequence $\pi_0 \rightarrow \pi_1 \ldots \rightarrow \pi_n$ (possibly infinite) there is an a with $\pi_i \rightarrow^* a$ for all i. It is *increasing (Inc)* if there is a map f from A to $\mathbb{N}$ such that $a \rightarrow b$ implies $f(a) < f(b)$. It is *finitely branching (FB)* if for all a, the set of b with $a \rightarrow b$ is finite.

The following relations are well-known.

Proposition 2 (Classical Setting [2,12]).

1. *confluent (C)* $\Rightarrow$ *normal form property (NF)* $\Rightarrow$ *unique normal form property (UN)*
2. *terminating (T) and locally confluent (LC)* $\Rightarrow$ *confluent (C) (Newman's Lemma)*
3. *unique normal form property (UN) and normalizing (N)* $\Rightarrow$ *confluent (C)*
4. *unique normal form property (UN) and normalizing (N)* $\Rightarrow$ *inductive (Ind)*
5. *inductive (Ind) and increasing (Inc)* $\Rightarrow$ *terminating (T)*
6. *locally confluent (LC) and normalizing (N) and increasing (Inc)* $\Rightarrow$ *terminating (T)*

We can now switch to the probabilistic case. The idea behind the probabilistic setting is to consider reductions with probabilities.

Let us first come back to school [10,7,20]: a *σ-algebra* on a set Ω is a set of subsets of Ω which contains the empty-set, and is stable by countable union and complementation. In particular, the set of subsets is a natural σ-algebra for any countable set. A *measurable space* (Ω, σ) is a set with a σ-algebra on it. If (Ω, σ) and (Ω', σ') are measurable spaces, a function $f : \Omega \to \Omega'$ is *measurable* if for all W in σ', $f^{-1}(W) \in \sigma$.

A *probability* is a function P from a σ-algebra to $[0, 1]$, which is countably additive, and such that $P(\Omega) = 1$. A triplet (Ω, σ, P) is called *a probability space*. A *random variable* is a measurable function on some probability space.

Given two probabilities P and P' on σ and σ' respectively, one can consider $P \oplus P'$ which is the product measure defined on the σ-algebra $\sigma \times \sigma'$, and is characterized by $P \oplus P'(A \times A') = P(A)P(A')$. Given $A, B \in \sigma$, when $P(B) > 0$, *the conditional probability of A given B* is by definition $P(A|B) = P(A \cap B)/P(B)$.

A *stochastic sequence on a set S* is a family $(X_i)_{i\in\mathbb{N}}$, of random variables defined on some fixed probability space (Ω, σ, P) with values on S. It is said to be *Markovian* if its conditional distribution function satisfies the so-called Markov property, that is for all n

$$P(X_n = s|X_0 = \pi_0, X_1 = \pi_1, \ldots, X_{n-1} = \pi_{n-1}) = P(X_n = s|X_{n-1} = \pi_{n-1}),$$

and *homogeneous* if furthermore this probability is independent of n.

We are ready to define probabilistic abstract reduction systems (PARS). The idea is that a PARS is given by some set A, and a function $[s \rightsquigarrow t]$ that gives, for all s,t in A, the probability of going from s to t. Formally:

Definition 1 (PARS). *A probabilistic abstract reduction system (PARS) is a pair $\mathcal{A} = (A, [\_ \rightsquigarrow \_])$ consisting of a countable set A and a function $[\_ \rightsquigarrow \_]$ from $A \times A$ to $[0, 1]$, such that for all s, $\sum_{t\in A}[s \rightsquigarrow t]$ is 0 or 1.*

Note that we allow $\sum_{t\in A}[s \rightsquigarrow t]$ to be 0. This happens exactly for *terminal* states: a state $a \in A$ is *terminal* if there is no b with $[a \rightsquigarrow b] > 0$. For non-terminal states, $\sum_{t\in A}[s \rightsquigarrow t]$ is necessarily 1.

A *derivation of* $\mathcal{A}$ is then a finite or infinite homogeneous Markovian stochastic sequence whose transition probabilities are given by $[\_ \rightsquigarrow \_]$. Formally:

Definition 2 (Derivations). *A derivation* π *of* $\mathcal{A}$ *is an homogeneous Markovian stochastic sequence* $\pi = (\pi_i)_{i\in\mathbb{N}}$ *on*

$$S \cup \{\bot\}$$

such that for all i*,*

$$\begin{aligned} P(\pi_{i+1} = t | \pi_i = s) &= [s \rightsquigarrow t] \text{ if } s \neq \bot \text{ non-terminal} \\ &= 1 \text{ if } t = \bot \text{ and } s \neq \bot \text{ terminal} \\ &= 1 \text{ if } t = \bot \text{ and } s = \bot \\ &= 0 \text{ otherwise.} \end{aligned}$$

PARS correspond to the extension of ARS with probabilities on derivations. Indeed, to a finitely branching[2] ARS $\mathcal{A} = (A, \rightarrow)$, we can associate a PARS $\mathcal{A}' = (A, [\_ \rightsquigarrow \_])$ where, for all $x, y \in A$, $[x \rightsquigarrow y]$ is 0 if $(x, y) \notin \rightarrow$, $[x \rightsquigarrow y] = 1/cardinal(\{t|(s,t) \in \rightarrow\})$ otherwise. Derivations of the ARS A and of the PARS $\mathcal{A}'$ are in correspondence: a non-terminating chain $\pi_0 \rightarrow \pi_1 \ldots \rightarrow \pi_n \ldots$ of $\mathcal{A}$ corresponds to derivation $\pi_0\pi_1 \ldots \pi_n \ldots$ of $\mathcal{A}'$ (observe that we have $\pi_i \in A\ \forall i$). A terminating chain $\pi_0 \rightarrow \pi_1 \ldots \rightarrow \pi_n$ of $\mathcal{A}$ corresponds to derivation $\pi_0\pi_1 \ldots \pi_n \bot\bot \ldots$ of $\mathcal{A}'$.

Clearly, to an ARS can correspond several PARS. Conversely, to a PARS $\mathcal{A}' = (A, [\_ \rightsquigarrow \_])$ corresponds a unique ARS $\mathcal{A} = (A, \rightarrow)$ which is obtained by forgetting probabilities: the relation $\rightarrow \subset A \times A$ of $\mathcal{A}$ is defined by $s \rightarrow t$ iff $[s \rightsquigarrow t] > 0$. This unique ARS, or the corresponding relation $\rightarrow$, will be called the *projection of* $\mathcal{A}'$.

The matrix $(p_{t,s}) = (P(\pi_i = t | \pi_{i-1} = s))$ on $S \cup \{\bot\}$ is what is called a stochastic matrix (even when S is an infinite set) [4]. It has the nice property that columns sum to 1. Actually our setting is the one of Homogeneous Markov Chain (HMC) theory [4]. The main difference is that we will focus on the generalization of classical rewriting notions and not on the usual Markov chain problematics: see [4] for a presentation of HMC theory.

Definition 3 (Relations $\rightsquigarrow$, $\rightsquigarrow^n$,$\rightsquigarrow^*$). *Let* $s \in A$ *be a state, and* $\pi = \pi_0\pi_1 \ldots \pi_n$ *a derivation with* $\pi_0 = s$*. We write*

1. $s \rightsquigarrow t$ *iff* $\pi_1 = t$,
2. $s \rightsquigarrow^n t$ *iff* $\pi_n = t$,
3. $s \rightsquigarrow^{\leq n} t$ *iff there is some* $i \leq n$ *with* $\pi_i = t$,
4. $s \rightsquigarrow^* t$ *iff there is some* $i \in \mathbb{N}$ *with* $\pi_i = t$.

[2] To an unrestricted (non-necessarily finitely-branching) ARS associate similarly a PARS $\mathcal{A}'$ by putting probabilities $1/2, 1/4, 1/8, \ldots$ on the outgoing transitions of a non-terminal s when the set $\{t|(s,t) \in \rightarrow\}$ is not finite.

For $s, t \in A$, we write $[s \rightsquigarrow t]$ (respectively: $[s \rightsquigarrow^n t]$,$[s \rightsquigarrow^{\leq n} t]$,$[s \rightsquigarrow^* t]$) for the probability that $s \rightsquigarrow t$ (resp. $s \rightsquigarrow^n t$, $s \rightsquigarrow^{\leq n} t$, $s \rightsquigarrow^* t$) holds on a derivation $\pi = \pi_0\pi_1 \dots \pi_n \dots$ given that $\pi_0 = s$.

Derivations have the nice property of being preserved by shifting: a *stopping time* with respect to a stochastic sequence $(X_n)_{n\in\mathbb{N}}$ is a random variable τ taking its values in $\mathbb{N} \cup \{\infty\}$ such that for all integer $i \geq 0$, the event $\tau = m$ can be expressed in terms of $X_0, X_1, \dots, X_m$ [4]. A typical example is *the hitting time* of some state $s \in S$ defined as $T = \inf\{i | X_i = s\}$. An other typical example is a constant time which is a particular stopping time [4].

Observation 1 (Strong Markov Property) *Let τ be a stopping time with respect to a derivation $\pi = \pi_0\pi_1 \dots \pi_n \dots$ of $\mathcal{A}$. The derivation after τ, (also called the τ-shift of π) is by definition the derivation $(\pi'_i)_{i\in\mathbb{N}}$ defined by $\pi'_i = \pi_{\tau+i}$.*

Then for any state $s \in S$, given that $X_\tau = s$ (hence we suppose $\tau \neq \infty$), the derivation after τ is a derivation of $\mathcal{A}$.

Proof. This is a restatement of the so-called "Strong Markov Property" for HMC: see [4] for example.

Clearly, the function $[\_ \rightsquigarrow \_]$ corresponds to the function $[\_ \rightsquigarrow \_]$ of $\mathcal{A}$. From the (strong) homogeneous Markov properties, we can also easily prove:

Proposition 3. *for all $s, t \in A$, $n \geq 1$*

1. $[s \rightsquigarrow^1 t] = [s \rightsquigarrow t]$
2. $[s \rightsquigarrow^n t] = \sum_{u\in A}[s \rightsquigarrow^{n-1} u][u \rightsquigarrow t]$
3. $[s \rightsquigarrow^* t] = \lim_{n\to\infty}[s \rightsquigarrow^{\leq n} t]$

Example 4. Let $\mathcal{A}$ be the PARS defined by $A = \{a, b\}$ and

1. $[a \rightsquigarrow a] = 1/2$
2. $[a \rightsquigarrow b] = 1/2$

Then.

1. $[a \rightsquigarrow^n a] = 1/2^n$
2. $[a \rightsquigarrow^n b] = 1/2^n$
3. $[a \rightsquigarrow^{\leq n} a] = 1$
4. $[a \rightsquigarrow^{\leq n} b] = 1 - 1/2^n$
5. $[a \rightsquigarrow^* a] = 1$
6. $[a \rightsquigarrow^* b] = 1$

Recall that in the classical setting that an ARS $\mathcal{A}$ is *locally confluent (LC)* if $b \leftarrow a \rightarrow c \Rightarrow \exists d\ b \rightarrow^* d \wedge c \rightarrow^* d$. We will then say that a PARS $\mathcal{A}$ is *probabilistically locally confluent (pLC)* if $P(\exists d\ b \rightsquigarrow^* d \wedge c \rightsquigarrow^* d) > 0$ whenever $a \rightsquigarrow b$ and $a \rightsquigarrow c$ by two independently chosen derivations.

Of course, such a definition must be understood as follows: a PARS $\mathcal{A}$ is *probabilistically locally confluent*, if for all $a, b, c \in A$, if $a \rightsquigarrow b$ by some derivation π, and $a \rightsquigarrow c$ by some independently chosen derivation π', then the probability

that there is a $d \in A$ such that simultaneously $b \rightsquigarrow^* d$ by derivation π after stopping time $\tau = 1$ and $c \rightsquigarrow^* d$ by derivation π' after stopping time $\tau = 1$ is positive. The probability measure considered here is the product measure $P \oplus P$, since we consider two independently randomly chosen derivations π and π'.

Observe that from Strong Markov Property, this is equivalent to say that, for all $b, c \in A$, the probability that for two independently randomly chosen derivations π and π' starting from b and c respectively, there is a $d \in A$ such that simultaneously $b \rightsquigarrow^* d$ by derivation π and $c \rightsquigarrow^* d$ by derivation π', is positive, whenever $[a \rightsquigarrow b] > 0$ and $[a \rightsquigarrow c] > 0$ for some $a \in A$.

In the same vein, we will say that $\rightsquigarrow$ is *almost-surely locally confluent (PLC)* if $P(\exists d\ b \rightsquigarrow^* d \wedge c \rightsquigarrow^* d) = 1$ whenever $a \rightsquigarrow b$ and $a \rightsquigarrow c$ by two independently chosen derivations.

We let the reader guess what *probabilistically confluent (pC)* , and *almost-surely confluent (PC)* are.

Proposition 1 has the following equivalent:

Proposition 4. *The following are equivalent:*

1. $\rightsquigarrow$ *is almost-surely confluent (PC) (resp. probabilistically confluent (pC))*
2. $\rightsquigarrow^*$ *is almost-surely locally confluent (PLC) (resp. probabilistically locally confluent (pLC))*
3. $\rightsquigarrow^*$ *is almost-surely confluent (PC) (resp. probabilistically confluent (pC))*
4. $a \rightsquigarrow b$, $a \rightsquigarrow^* c$ *implies there is a* d *with* $b \rightsquigarrow^* d, c \rightsquigarrow^* d$ *with probability one (resp. positive probability).*

Recall that an $a \in A$ is in normal form if there is no b with $[a \rightsquigarrow b] > 0$.

We say that $a \in A$ *probabilistically has a normal form (phnf)* if $P(\exists b\ nf\ a \rightsquigarrow^* b) > 0$. We say that $a \in A$ *almost-surely has a normal form (Phnf)* if $P(\exists b\ nf\ a \rightsquigarrow^* b) = 1$.

A relation is *probabilistically normalizing (pN)* if every $a \in A$ probabilistically has a normal form (phnf) . A relation is *almost-surely normalizing (PN)* if every $a \in A$ almost-surely has a normal form (Phnf) .

A relation is *almost-surely terminating (PT)* if every derivation $\pi_0, \pi_1, \ldots, \pi_n$ is terminal (ultimately equal to $\bot$) with probability 1. It is *probabilistically terminating (pT)* if this latter probability is positive.

Two states $a, b \in A$ are said to be *convertible* if a and b are convertible in the classical sense for the projection of $\mathcal{A}'$, that is for the classical binary relation $[\_ \rightsquigarrow \_] > 0$.

A relation has *unique normal form property (UN)* if for all $a, b \in A$, if a and b are convertible and in normal form, then a and b are identical.

A relation has the *probabilistic normal form property (pNF)* if $P(b \rightsquigarrow^* a) > 0$ whenever a is in normal form and convertible with b. A relation has the *almost-sure normal form property (PNF)* if $P(b \rightsquigarrow^* a) = 1$ whenever a is in normal form and convertible with b.

We are now ready to study the properties of the above notions.

We will then say that a PARS is *locally confluent (LC)* (respectively confluent (C) , normalizing (N) , terminating (T) , unique normal form property (UN) , inductive (Ind) , increasing (Inc) , finitely branching (FB)) if its projection is.

It is noticeable that the projection operator commutes with finite iteration and (reflexive) transitive iteration. In other words, probabilistic joinability and joinability corresponds. Formally,

Proposition 5. *Let $\mathcal{A}' = (A, [\_ \rightsquigarrow \_])$ be a PARS and $\mathcal{A} = (A, \rightarrow)$ its projection: $s \rightarrow t$ iff $[s \rightsquigarrow t] > 0$.*

Then, for all $s, t \in A$,

1. *$[s \rightsquigarrow^n t] > 0$ iff $s \rightarrow^n t$*
2. *$[s \rightsquigarrow^* t] > 0$ iff $s \rightarrow^* t$*

Proof. The two assertions are proved by induction over n. They are clear for $n = 0$ and $n = 1$. For $n > 1$, we have $s \rightarrow^n t$ iff there is a u with $s \rightarrow^{n-1} u$ and $u \rightarrow t$ iff there is a u with $[s \rightsquigarrow^{n-1} u] > 0$ and $[u \rightsquigarrow t]$ iff $[s \rightsquigarrow^n t] = \sum_{u \in A} [s \rightsquigarrow^{n-1} u][u \rightsquigarrow t] > 0$. For the second assertion, we have $s \rightarrow^* t$ iff there is a n with $s \rightarrow^n t$ iff there is a n with $[s \rightsquigarrow^n t] > 0$ iff $[s \rightsquigarrow^* t] > 0$. Indeed, if there is such a n, $[s \rightsquigarrow^* t]$ is at least $[s \rightsquigarrow^n t]$. For the other direction, we clearly have by sigma-additivity of probabilities $[s \rightsquigarrow^* t] \leq \sum_{n \in \mathbb{N}} [s \rightsquigarrow^n t]$.

We then claim.

Proposition 6. 1. *almost-surely locally confluent (PLC) $\Rightarrow$ probabilistically locally confluent (pLC) $\Leftrightarrow$ locally confluent (LC) .*
2. *The reverse implication is false in general.*

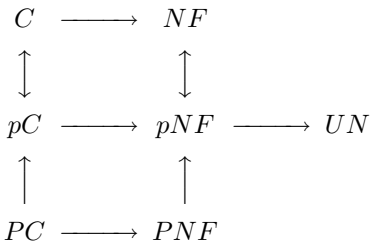
Proof. Only the equivalence between probabilistically locally confluent (pLC) and locally confluent (LC) is not trivial. Clearly $P(\exists d\ a \rightsquigarrow^* d \wedge b \rightsquigarrow^* d) > 0$ iff $\exists d\ P(a \rightsquigarrow^* d \wedge b \rightsquigarrow^* d) > 0$ iff $\exists d\ P(a \rightsquigarrow^* d) > 0 \wedge P(b \rightsquigarrow^* d) > 0$. From previous lemma, that happens iff $[\_ \rightsquigarrow \_] > 0$ is locally confluent (LC) .

In the same vein, we can prove:

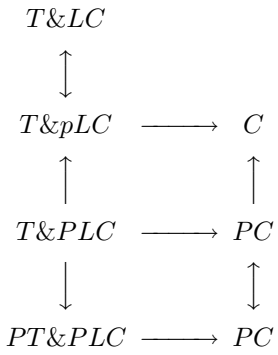
Proposition 7. 1. *almost-surely confluent (PC) $\Rightarrow$ probabilistically confluent (pC) $\Leftrightarrow$ confluent (C)*
2. *almost-surely has a normal form (Phnf) $\Rightarrow$ probabilistically has a normal form (phnf) $\Leftrightarrow$ has a normal form (hnf)*
3. *almost-surely normalizing (PN) $\Rightarrow$ probabilistically normalizing (pN) $\Leftrightarrow$ normalizing (N)*
4. *almost-sure normal form property (PNF) $\Rightarrow$ probabilistic normal form property (pNF) $\Leftrightarrow$ normal form property (NF)*
5. *terminating (T) $\Rightarrow$ almost-surely terminating (PT) $\Leftrightarrow$ almost-surely normalizing (PN) $\Rightarrow$ probabilistically terminating (pT)*
6. *The reverse implications are false in general.*

We then get the following picture:

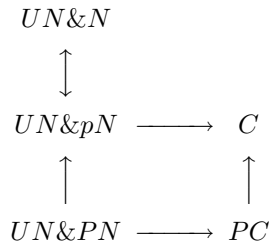
Proposition 8. *1.*



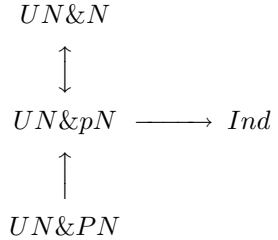
2. Newman's Lemma.



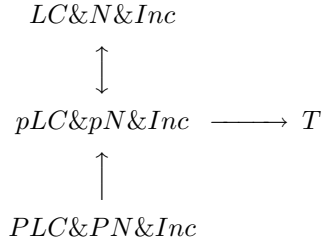
3.



4.



5.



The previous implications can all be derived by adapting the proofs from the classical abstract reduction systems towards probabilistic ones.

Observe that they were obtained by considering the classical rewriting settings and testing whether it generalizes to that context. That means first that we are not at all exhaustive: some reverse implications, or other implications may hold. Second, the classical theory of HMC can a priori also be used to derived other facts, or relations between our notions.

6 Conclusion

In this paper we presented an extension of the ELAN system to deal with the prototyping and modeling of systems with probabilistic evolutions. We proposed to extend the system by adding a new "probabilistic choose" operator to the strategy language of the system.

We argued through simple classical examples that this approach that puts probabilities at the level of strategies, i.e. at the level of the control of rewriting is very natural. This is in contrast with similar works where probabilities are in some sense hard-coded in rules [8]. Furthermore, this approach has the advantage that the problem of probabilities normalization (probabilities must be normalized to sum to 1) in the above mentioned work is avoided in a nice manner: this is put at the level of the definition of the operators which are coded in the language itself.

Dealing with probabilistic systems gives rise to many very interesting questions about the generalization of the classical rewriting notions. We put the first stones in that direction by defining several notions and stating generalizations of classical results for abstract derivation systems. Our results mainly say that the notions defined by considering joinability with positive probability corresponds to classical notions by putting away probabilities, and that the notions defined by considering almost-sure joinability are stronger.

These are of course only the first steps in the direction of fully understanding probabilistic rewrite systems. Actually, when dealing with rewrite systems, the objects of interest are; (1) abstract rewrite systems, (2) the rewrite relation that should be stable by context and substitution, (3) the rewrite logic, (4) the rewrite calculus. The generalization of all these concepts and their relations dealing with probabilistic systems/theories give rise to many potentially interesting problems. We touched only (1), and on this topics we think that some other results could also been derived from Homogeneous Markov Chain Theory.

References

1. D. Abdemouche. Stratégies probabilistes en elan et applications. Mémoire de dea en informatique, Université Henri Poincaré – Nancy 1, July 2001.
2. F. Baader and T. Nipkow. *Term Rewriting and all That*. Cambridge University Press, 1998.

3. P. Borovanský, C. Kirchner, H. Kirchner, and C. Ringeissen. Rewriting with strategies in ELAN: a functional semantics. *International Journal of Foundations of Computer Science*, 12(1):69–98, February 2001.
4. P. Brémaud. *Markov Chains.* Springer, 1991.
5. H. Cirstea and C. Kirchner. The rewriting calculus — Part I *and* II. *Logic Journal of the Interest Group in Pure and Applied Logics*, 9(3):427–498, May 2001.
6. A. Dutech, O. Buffet, and F. Charpillet. Multi-Agent Systems by Incremental Gradient Reinforcement Learning. In *17th International Joint Conference on Artificial Intelligence, Seattle, WA, USA*, volume 2, pages 833–838, August 2001.
7. W. Feller. *An introduction to probability theory and its applications, volume 1 and 2.* Wiley, 1968.
8. T. Frühwirth, A. Di Pierro, and H. Wiklicky. Toward probabilistic constraint handling rules. In S. Abdennadher and T. Frühwirth, editors, *Proceedings of the third Workshop on Rule-Based Constraint Reasoning and Programming (RCoRP'01)*, Paphos, Cyprus, December 2001. Under the hospice of the International Conferences in Constraint Programming and Logic Programming.
9. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Massachusetts, 1989.
10. G. Grimmett. *Probability Theory.* Cambridge University Press, 1993.
11. H. Kirchner and P.-E. Moreau. Promoting rewriting to a programming language: A compiler for non-deterministic rewrite programs in associative-commutative theories. *Journal of Functional Programming*, 11(2):207–251, 2001.
12. J. W. Klop. Term rewriting systems. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 1, pages 1–117. Oxford University Press, Oxford, 1992.
13. D. E. Knuth. *The art of Computer Programming*, volume 2. Addison-Wesley Publishing Company, second edition, 1981.
14. P. Lincoln, J. Mitchell, and A. Scedrov. Stochastic interaction and linear logic. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 147–166. Volume 222, London Mathematical Society Lecture Notes, Cambridge University Press, 1995.
15. R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.
16. A. D. Pierro and H. Wiklicky. An operational semantics for probabilistic concurrent constraint programming. In *Proceedings of the 1998 International Conference on Computer Languages*, pages 174–183. IEEE Computer Society Press, 1998.
17. D. Pierro and Wiklicky. A markov model for probabilistic concurrent constraint programming. In *APPIA-GULP-PRODE'98, Joint Conference on Declarative Programming*, 1998.
18. D. Pierro and Wiklicky. Probabilistic concurrent constraint programming: Towards a fully abstract model. In *MFCS: Symposium on Mathematical Foundations of Computer Science*, 1998.
19. D. Pierro and Wiklicky. Concurrent constraint programming: Towards probabilistic abstract interpretation. In *2nd International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP'00)*, 2000.
20. W. Rudin. *Real and Complex Analysis, 3rd edition.* McGraw Hills, USA, March 1987.
21. V. Subrahmanian. Probabilistic databases and logic programming. In *International Conference on Logic Programming*, volume 2237 of *Lecture Notes in Computer Science*, page 10. Springer-Verlag, 2001.