Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 338 (2018) 115–131

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Confluence in Probabilistic Rewriting

Alejandro Díaz-Caro[a,b,1] Guido Martínez[c]

[a] *Universidad Nacional de Quilmes*
Roque Sáenz Peña 352. B1876BXD Bernal, Buenos Aires, Argentina

[b] *CONICET-Universidad de Buenos Aires. Instituto de Investigación en Ciencias de la Computación*
Pabellón 1, Ciudad Universitaria. C1428EGA Buenos Aires, Argentina
adiazcaro@icc.fcen.uba.ar

[c] *CIFASIS-CONICET & Universidad Nacional de Rosario*
Ocampo y Esmeralda, S2000EZP. Rosario, Santa Fe, Argentina
martinez@cifasis-conicet.gov.ar

Abstract

Driven by the interest of reasoning about probabilistic programming languages, we set out to study a notion of uniqueness of normal forms for them. To provide a tractable proof method for it, we define a property of *distribution confluence* which is shown to imply the desired uniqueness (even for infinite sequences of reduction) and further properties. We then carry over several criteria from the classical case, such as Newman's lemma, to simplify proving confluence in concrete languages. Using these criteria, we obtain simple proofs of confluence for λ_1, an affine probabilistic λ-calculus, and for Q*, a quantum programming language for which a related property has already been proven in the literature.

Keywords: abstract rewriting system, probabilistic rewriting, confluence

1 Introduction

In the formal study of programming languages, modelling execution via a small-step operational semantics is a popular choice. Such a semantics is given by an *abstract rewriting system* (ARS) which, mathematically, is no more than a binary relation on terms specifying whether one term can rewrite to another. This relation is not required to be a function, and can thus allow for a program to rewrite in two different ways. In such a case, it is important that the different execution paths for a given program reach the same final value (if any); thus guaranteeing that any two determinisations of the semantics (i.e. strategies) coincide on every program.

This correctness property is expressible at the level of relations, and is known as *uniqueness of normal forms* (UN): any two irreducible terms reachable from a

[1] Partially supported by projects ECOS-Sud A17C03 QuCa and PICT 2015-1208.

https://doi.org/10.1016/j.entcs.2018.10.008
1571-0661/© 2018 The Author(s). Published by Elsevier B.V.
This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

common starting point must be equal. For non-trivial languages, such as the λ-calculus, it can be hard to prove this property directly. Fortunately, the property of *confluence* can serve as a proof method for it, since it trivially implies UN and yet its proof tends to be more tractable. This was the approach followed by Church and Rosser in [4, Corollary 2], where UN and confluence were first proven for the λ-calculus in 1936. Nowadays, confluence is widely used to show the adequacy of operational semantics in many kinds of programming languages.

Since some decades ago, there has been growing interest in *probabilistic* programming languages [12, 16] and, in particular, those where reduction itself is stochastic. Properties of such languages have been studied, in particular, from a language-independent rewriting approach [1–3, 15] and for specific calculi [6, 8–11]. For example, taking the former approach, [1] introduces a notion of probabilistic termination of rewriting systems and provides techniques to prove it, extending the classical ones to this probabilistic setting. Following the latter approach, [9] takes a non-deterministic λ-calculus and endows it with probabilistic call-by-value and call-by-name operational semantics (in both small-step and big-step style for each) and proves they simulate each other via CPS translations, following a classic result by Plotkin [21].

Our focus in this paper is the interaction between probabilistic reductions and non-determinism, where the latter comes from different possible reduction choices. Such choices exist, for example, when a given program contains two reducible subexpressions, each of which is probabilistic. In that case, the program can take a step to two different normalised distributions depending on the choice. Given such non-determinism, a natural question to ask is whether the result of a program (which is a distribution of values [16]) is not affected by the strategy, analogously to UN for classical languages. This is precisely the property we set out to study here, developing an associated notion of confluence for it.

While this same property has already been studied in the literature [8, 11], it has only been done for concrete languages, and so a language-independent study was previously lacking. Further, the techniques employed are not immediately applicable to other calculi. There are also other studies of confluence in a probabilistic setting at a more abstract level [3, 15], but such notions are fundamentally different from what we previously described, and of limited use for programming languages.

For a concrete example, let us take a hypothetical language for representing die rolls, where $\Box$ represents an unrolled die which can reduce to any element in $\{1, \dots, 6\}$ with equal probability and '$-$' represents subtraction. For a pair of dice $(\Box, \Box)$, it should be allowed to choose which one to roll first. Rolling the first die can give in any term in the set $\{(i, \Box)\}_{i=1,6}$ with equal probability; and similarly for the second. When continuing the rolls, both alternatives provide the same uniform distribution. However, this could be not so: consider the term $(\lambda x.\ x - x)\ \Box$. If the die is rolled before β-reducing, the result can only be 0, but if one β-reduces first, obtaining $(\Box, \Box)$, the final result can be any number in $\{-5, \dots, 5\}$. We shall later present a similar language and provide conditions to avoid this discrepancy.

Outline. In §2 we introduce probabilistic rewriting and the problem of uniqueness of distributions. In §3 we define a rewriting system over distributions giving rise to our notion of confluence and prove it adequate. In §4 we derive criteria which simplify the task of proving distribution for concrete languages. In §5 we extend some of our results to asymptotically terminating terms. In §6 we prove confluence for two concrete calculi: a simple probabilistic calculus dubbed λ_1 and the quantum lambda calculus Q^* [8]. Finally, in §7 we conclude, give some insights on future directions, and analyse some related work.

2 Probabilistic Rewriting

2.1 Preliminaries

We assume familiarity with abstract rewriting. We adopt the terminology from [4] and call a sequence of expansions followed by a sequence of reductions a *peak*. When the order is reversed, such a sequence is called a *valley*. The property of confluence can then be expressed as "all peaks have a valley". When an ARS $\mathcal{A}$ is confluent, we note it by $\mathcal{A} \models \mathrm{CR}$, and similarly for UN and other properties. The relation R *modulo* E, noted as R/E, is defined for any equivalence relation E as $E \cdot R \cdot E$ [14,20].

We denote by $\mathscr{L}(X)$ the type of finite lists with elements in X, where '[ ]', ':' and '$+\!\!+$' denote the empty list, the list constructor, and list concatenation, respectively. We also use the notation $[a, b, c]$ for $a : b : c : [\,]$.

We define $\mathscr{D}(A) = \mathscr{L}(\mathbb{R}^+ \times A)$, used as an explicit representation of (finitely-supported) distributions. There is no further restriction on $\mathscr{D}(A)$. In particular, any given element might appear more than once, as in $[({}^1\!/_3, a), ({}^1\!/_2, b), ({}^1\!/_6, a)]$. For a point (p, a) of the distribution, p is called the *weight* and a the *element*. The weight of a distribution is defined to be the sum of all weights of its points, and can be any positive real number. When a normalised distribution is required, we use the type $\mathscr{D}_1(A)$, defined as the set of those $d \in \mathscr{D}(A)$ with unit weight. We abbreviate the distribution $[(p_1, a_1), (p_2, a_2), \ldots, (p_n, a_n)]$ by $[(p_i, a_i)]_i$, where n should be clear from context. We also write αD for the distribution obtained by scaling every weight in D by α; that is, $\alpha\,[(p_i, a_i)]_i = [(\alpha p_i, a_i)]_i$.

For reasoning about equivalence of distributions, we define a relation '$\sim$' as the congruence closure of following rules (i.e. the smallest relation satisfying the rules and such that $D \sim D'$ implies $E_1 +\!\!+ D +\!\!+ E_2 \sim E_1 +\!\!+ D' +\!\!+ E_2$, for any E_1, E_2):

$$\textsc{Flip}\ \frac{}{[(p,a),(q,b)] \sim [(q,b),(p,a)]} \qquad \textsc{Join}\ \frac{}{[(p,a),(q,a)] \sim [(p+q,a)]} \qquad \textsc{Split}\ \frac{}{[(p+q,a)] \sim [(p,a),(q,a)]}$$

Note that $\sim$ is symmetric, since JOIN is the inverse of SPLIT and FLIP is its own inverse. We distinguish some subsets of $\sim$ by limiting the rules that may be used. We note by S the congruence closure of SPLIT, by (FJ) the congruence closure of both FLIP and JOIN, and similarly for other subsets.

We call two distributions *equivalent* when they are related by the reflexive-transitive closure of $\sim$, noted '$\approx$'. Two distributions are equivalent, then, precisely when they assign the same total weight to every element, irrespective of order and multiplicity.

Arguably, using such a definition of distributions and equivalence is cumbersome and less clear than using a more semantic one. However, we feel this is outweighed by the degree of rigor attained in later proofs (especially as we found some of them to be quite error-prone). As a secondary benefit, most of our development should be straightforwardly mechanisable, since all equivalence steps are made explicit.

A tangible disadvantage of using lists is that only *finitely*-supported distributions can be represented. This restriction (which is anyway lifted for infinite reduction sequences) is present in other works as well (e.g. [10, 22]) and it does not seem severe for modelling programming languages.

2.2 Probabilistic Abstract Rewriting Systems (PARS)

To model the uncertainty in probabilistic rewriting, we cannot use a simple relation between elements as used in ARS. We must instead relate elements to *distributions* of elements. Further, to model elements such as $(\square, \square)$, we need to be able to relate elements to several such distributions. This motivates the following definition.

Definition 2.1 *A probabilistic abstract rewriting system (PARS) is a pair $(A, \mapsto)$ where A is a set and $\mapsto$ a relation of type $\mathcal{P}(A \times \mathscr{D}_1(A))$ (called the "pointwise evolution relation").*

It should be clear that every ARS is also a PARS by taking "Dirac" distributions (i.e. normalised, single-point distributions). We can provide a simple example of a PARS by extensionally listing $\mapsto$, as is commonly done for ARS.

Example 2.2 Let $\mathcal{A}$ be the PARS given by

$$a \mapsto [(2/3, b), (1/3, c)] \quad a \mapsto [(2/5, a), (3/5, d)] \quad b \mapsto [(1/2, c), (1/2, d)] \quad c \mapsto [(1, d)]$$

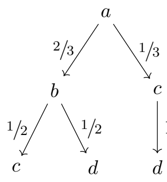
Here, a is the only non-deterministic element. We call d a *terminal* element since it has no successor distributions. More significant examples are presented in §6.

Execution in a PARS is a mixture of non-deterministic and probabilistic choices. The first kind, corresponding to the $\mathcal{P}$ operator, occur when the machine chooses a successor distribution for the current element. The second kind, corresponding to the $\mathscr{D}_1$ operator, is a random choice between the elements of the chosen successor distribution. To model such execution, we use the notion of *computation tree*.

Definition 2.3 *Given a PARS $(A, \mapsto)$, we define the set of its (finite) "computation trees" with root a (noted $\mathcal{T}(a)$) inductively by the following rules. We also sometimes consider infinite computation trees, by taking the coinductively defined set instead.*

$$\frac{}{a \in \mathcal{T}(a)} \qquad \frac{a \mapsto [(p_1, a_1), \ldots, (p_n, a_n)] \qquad t_i \in \mathcal{T}(a_i)}{[a; (p_1, t_1); \ldots; (p_n, t_n)] \in \mathcal{T}(a)}$$

A graphical representation of a tree for the PARS in Example 2.2 is given to the right. A tree in $\mathcal{T}(a)$ represents one possible (uncertain) evolution of the system after starting on a. There is no further assumption about trees: in particular, if an element is expanded many times in a given tree, different successor distributions may be used each time. In other words, we do not assume a "Markovian scheduler" [1,2].



$$\begin{tikzcd}[column sep=small] & & a \arrow[dl, "2/3"'] \arrow[dr, "1/3"] & \\ & b \arrow[dl, "1/2"'] \arrow[dr, "1/2"] & & c \arrow[d, "1"] \\ c & & d & d \end{tikzcd}$$

Collecting the leaves of a tree, along with their accumulated probabilities, gives rise to a normalised list distribution.

Definition 2.4 *The "support" of a tree T is a normalised distribution, defined by:*

$$\begin{aligned} \mathrm{supp}(a) &= [(1,a)] \\ \mathrm{supp}([a;(p_1,t_1);\ldots;(p_n,t_n)]) &= p_1\mathrm{supp}(t_1) +\!\!+ \ldots +\!\!+ p_n\mathrm{supp}(t_n) \end{aligned}$$

When all the leaves of a tree are terminal elements, we call the tree *maximal* (as there is no proper supertree of it). We can now state our property of interest.

Definition 2.5 (UTD) *A PARS $\mathcal{A}$ has "unique terminal distributions" when for every a and $T_1, T_2 \in \mathcal{T}(a)$ maximal, we have* $\mathrm{supp}(T_1) \approx \mathrm{supp}(T_2)$.

We stated before that proving UN (for ARS) directly is usually hard. Since PARS subsume ARS, the same difficulties arise for proving UTD directly. Therefore, we seek a property akin to confluence, providing a compositional and more tractable proof method.

3 Rewriting Distributions and Confluence

To arrive at a notion of confluence we shall first define a rewriting over distributions (Definition 3.4) that is more liberal than that of computation trees (Definition 2.3).

Definition 3.1 *Given a PARS $\mathcal{A} = (A, \mapsto)$, we define the relation $\twoheadrightarrow_P$ (of type $\mathcal{P}(\mathscr{D}(A) \times \mathscr{D}(A))$) (called "parallel evolution") by the rules:*

$$\frac{a \mapsto A \qquad ds \twoheadrightarrow_P ds'}{(p,a) : ds \twoheadrightarrow_P pA +\!\!+ ds'} \qquad \frac{ds \twoheadrightarrow_P ds'}{(p,a) : ds \twoheadrightarrow_P (p,a) : ds'} \qquad \frac{}{[\,] \twoheadrightarrow_P [\,]}$$

Note that without using the first rule, this is just the identity relation on distributions. We note the subset of this relation where the first rule must be used at least once in a step as $\twoheadrightarrow_P^1$, and call it *proper* evolution. The $\twoheadrightarrow_P$ relation can simulate computation trees in this system, since it can be used to rewrite their supports in the sense of Lemma 3.3.

Definition 3.2 *We call a relation $\to$ "compositional" when, if $D_1 \to E_1$ and $D_2 \to E_2$, then $\alpha D_1 +\!\!+ \beta D_2 \to \alpha E_1 +\!\!+ \beta E_2$ for all $\alpha, \beta \in \mathbb{R}^+$.*

Lemma 3.3 *If $T \in \mathcal{T}(a)$, then* $[(1,a)] \twoheadrightarrow_P^* \mathrm{supp}(T)$

Proof. First, note that $\twoheadrightarrow_P$ is compositional. The result then follows by induction on T, using compositionality. □

We now define an ARS over distributions, combining both parallel evolution and equivalence steps. Our definition of confluence for a PARS $\mathcal{A}$ is then simply the usual confluence of that relation.

Definition 3.4 *Given a PARS $\mathcal{A} = (A, \mapsto)$, we define an associated ARS* $\mathrm{Det}(\mathcal{A})$ *(called the "determinisation" of $\mathcal{A}$) over the set $\mathscr{D}(A)$ by the relation $\twoheadrightarrow = (\twoheadrightarrow_P \cup \approx)$.*

Definition 3.5 (Distribution confluence) *We say a PARS $\mathcal{A}$ is "distribution confluent" (or simply "confluent") when* $\mathrm{Det}(\mathcal{A})$ *is confluent in the classical sense.*

Reduction in $\mathrm{Det}(\mathcal{A})$ is more liberal than the expansion of trees, since it allows for "partial" evolutions. Indeed, if $a \mapsto D$, then $[(1, a)] \twoheadrightarrow [(1/2, a), (1/2, a)] \twoheadrightarrow 1/2 D +\!\!\!+ [(1/2, a)]$. Nevertheless, Lemma 3.7 shows that its confluence implies UTD.

Lemma 3.6 *If D_1 is terminal and $D_1 \twoheadrightarrow^* D_2$, then $D_1 \approx D_2$ and D_2 is terminal.*

Proof. It is clear, from the definition of $\twoheadrightarrow_P$, that if D_1 is terminal and $D_1 \twoheadrightarrow_P D'$, then $D_1 = D'$ (that is, exactly equal). The result then follows by induction on the number of steps, and the transitivity and reflexivity of $\approx$. □

Lemma 3.7 *If $\mathcal{A} \models \mathrm{CR}$, then $\mathcal{A} \models \mathrm{UTD}$.*

Proof. Take $T_1, T_2 \in \mathcal{T}(a)$ maximal. We have from Lemma 3.3 that $\mathrm{supp}(T_2) \twoheadleftarrow^* [(1, a)] \twoheadrightarrow^* \mathrm{supp}(T_1)$. By confluence, there must exist C such that $\mathrm{supp}(T_2) \twoheadrightarrow^* C \twoheadleftarrow^* \mathrm{supp}(T_1)$. Since T_1, T_2 are maximal, their supports are terminal. Then, from Lemma 3.6, we get that $\mathrm{supp}(T_2) \approx C \approx \mathrm{supp}(T_1)$, as needed. □

Furthermore, beyond UTD, distribution confluence implies that diverging computations (with no terminal distribution) can also be joined. As a consequence of that, confluence gives a neat method for proving the consistency of the equational theory induced by $\twoheadrightarrow$, as long as two distinct terminal elements exist.

Lemma 3.8 *If D_1, D_2 are terminal distributions and $\mathcal{A}$ is confluent, then $D_1 \twoheadleftarrow\!\!\twoheadrightarrow^* D_2$ if and only if $D_1 \approx D_2$.*

Proof. The way back is trivial, so we detail the way forward. From confluence (repeatedly), D_1 and D_2 must have a common reduct. The result then follows from Lemma 3.6. □

Then, if a and b are distinct terminal elements, it follows that $\twoheadrightarrow$-convertibility is a consistent theory as $[(1, a)] \not\approx [(1, b)]$. Summarizing, in a confluent PARS, reasoning about equivalence of programs is simplified and there is a strong consistency guarantee about convertibility, much like in the classical case.

Readers familiar with rewriting modulo equivalence [14, 20] may wonder why we are not studying confluence modulo $\approx$, or the stronger Church-Rosser property modulo $\approx$. It turns out both of these are too strong for our purposes. The following system:

$$a \mapsto [(1/2, a), (1/2, b)]$$

should undoubtedly be considered confluent due to being deterministic; but in it we can form the following diagram

$$[({}^1\!/\!{}_2, a), ({}^1\!/\!{}_2, b)] \twoheadleftarrow_P [(1, a)] \approx [({}^1\!/\!{}_3, a), ({}^2\!/\!{}_3, a)]$$

which cannot be closed by $\twoheadrightarrow_P^* \cdot \approx \cdot \twoheadleftarrow_P^*$, due to a factor of 3 present in one side and not the other. Thus this system is neither confluent modulo $\approx$ nor CR modulo $\approx$, yet it is clearly distribution confluent. We therefore study the strictly weaker distribution confluence, which is strong enough for our purposes and easier to reason about.

4 Proving Distribution Confluence

4.1 Introduction

In the previous section, we introduced our definition of confluence and argued about its correctness. For it to be useful in practice, it should also be amenable to be proven. In this section we provide several simplified criteria for this task, obtaining analogues to the most usual methods for proving classical confluence.

Since distribution confluence is simply the classical confluence of $\twoheadrightarrow$, all existing classical criteria (e.g. the diamond property or Newman's lemma) are valid without modification. However, they are not very useful. One issue is that one needs to consider all distributions (instead of single elements) and the presence of equivalence steps in the peaks. Also, $\mathrm{Det}(\mathcal{A})$ is never strongly (or even weakly) normalising, so Newman's lemma is useless here. With respect to the diamond property, consider a system with $a \mapsto D$, then the following reductions are possible:

$$[(1, a)] \twoheadleftarrow [({}^1\!/\!{}_2, a), ({}^1\!/\!{}_2, a)] \twoheadrightarrow {}^1\!/\!{}_2 D +\!\!+ [({}^1\!/\!{}_2, a)]$$

and these two distributions cannot in general be joined in a single step, even if a has no other successor distribution.

Thus, a priori, it seems as if distribution confluence is even harder to prove than classical confluence. To relieve that, we shall prove various syntactic lemmas about the $\twoheadrightarrow$ relation, allowing us to decompose it into more manageable forms. We then show how we can limit our reasoning to Dirac distributions, ignore equivalence steps in the peaks and allow to use them freely in the valleys. Lastly, we carry over classical criteria for confluence into our setting, such as the aforementioned diamond property and Newman's lemma.

4.2 Decomposing the $\twoheadrightarrow$ relation

Since both $\twoheadrightarrow_P$ and $\approx$ are reflexive, $(\twoheadrightarrow_P \cup \approx)^*$ coincides with $(\twoheadrightarrow_P / \approx)^*$. Thus, since confluence is a property over the reflexive-transitive closure of a relation, it suffices to study the confluence of $\twoheadrightarrow_P / \approx$, where equivalence steps do not have a "cost", but are pervasive (as in rewriting modulo equivalence).

Given the precise syntactic definition for both relations, we can prove by analysing the reductions that any step of $\twoheadrightarrow_P/\!\approx$ can be made by first splitting, then evolving, and then joining back elements, as Lemma 4.3 states. We first introduce the following notion of commutation.[2]

Definition 4.1 (Sequential commutation) *We say that a relation R "commutes over" S when $S \cdot R \subseteq R \cdot S$, and note it as $R \dashv S$. The property can be expressed by the diagram on the right. Intuitively, it means that R can be "pushed" before S.*

$$\begin{tikzcd}
 & \cdot \arrow[dl, "S"'] \arrow[dr, dashed, "R"] & \\
\cdot \arrow[dr, "R"'] & & \cdot \arrow[dl, dashed, "S"] \\
 & \cdot &
\end{tikzcd}$$

A key property of sequential commutation is that if $R \dashv S$, then $(R \cup S)^* = R^* \cdot S^*$. It is also preserved when taking the n-fold composition (i.e. "n steps") or reflexive-transitive closures on each side. We now prove some commutations relating evolution and equivalence steps (the last one needs some "administrative" steps).

Lemma 4.2 *We have $\twoheadrightarrow_P \dashv (FJ)^*$; $S \dashv (FJ)^*$ and $\twoheadrightarrow_P \cdot S \subseteq S \cdot \twoheadrightarrow_P \cdot (FJ)^*$.*

Proof. By induction on the shape of the reductions. □

Lemma 4.3 *The relations $(\twoheadrightarrow_P/\!\approx)$ and $S^* \cdot \twoheadrightarrow_P \cdot (FJ)^*$ coincide.*

Proof. The backwards inclusion is trivial, so we detail only the forward direction. By making use of the second commutation in Lemma 4.2 we get that $\approx\, = S^* \cdot (FJ)^*$. Thus, we need to show $S^* \cdot (FJ)^* \cdot \twoheadrightarrow_P \cdot S^* \cdot (FJ)^* \subseteq S^* \cdot \twoheadrightarrow_P \cdot (FJ)^*$. The proof then proceeds by using the other two commutations to reorder the relations. □

Further, this equivalence extends to n-fold compositions and therefore to the reflexive-transitive closure.

Lemma 4.4 *The relations $(\twoheadrightarrow_P/\!\approx)^n$ and $S^* \cdot \twoheadrightarrow_P^n \cdot (FJ)^*$ coincide.*

Proof. By induction on n, and using the previous lemma and commutations. □

4.3 Simplifying diagrams

With the previous decompositions, we can now prove a very generic result about diagram simplification with a specific root D, which then easily generalizes to the whole system.

Definition 4.5 *We say a pair of relations (γ, δ) "closes" another pair (α, β) "on a" if whenever $b \leftarrow_\alpha a \rightarrow_\beta c$ then there exists d such that $b \rightarrow_\gamma d \leftarrow_\delta c$. The diagram for the property can be seen on the right. When this occurs for all a, we simply say "(γ, δ) closes (α, β)". Note that $\rightarrow$ is confluent precisely when $(\rightarrow^*, \rightarrow^*)$ closes $(\rightarrow^*, \rightarrow^*)$.*

$$\begin{tikzcd}
 & a \arrow[dl, "\alpha"'] \arrow[dr, "\beta"] & \\
b \arrow[dr, dashed, "\gamma"'] & & c \arrow[dl, dashed, "\delta"] \\
 & d &
\end{tikzcd}$$

Definition 4.6 *We call a relation $\rightarrow$ "local", when if $\alpha D_1 +\!\!\!+ \beta D_2 \rightarrow E$, then there exist E_1, E_2 such that $E = \alpha E_1 +\!\!\!+ \beta E_2$ and $D_i \rightarrow E_i$. (Note that $\twoheadrightarrow_P$ and S are local).*

[2] Note that this is not the usual notion of commuting relations, defined as $S^{-1} \cdot R \subseteq R \cdot S^{-1}$, which is a symmetric property and could be described as "parallel".

Theorem 4.7 *Let α, β be local relations and γ, δ compositional relations. If $(\gamma/\!\approx, \delta/\!\approx)$ closes (α, β) for the Dirac distributions of D, then $(\gamma/\!\approx, \delta/\!\approx)$ closes $(S^* \cdot \alpha \cdot (FJ)^*, S^* \cdot \beta \cdot (FJ)^*)$ for D.*

Proof. We give a sketch of the proof, more details can be found in [17]. We need to close $(S^* \cdot \alpha \cdot (FJ)^*, S^* \cdot \beta \cdot (FJ)^*)$. First, note that closing $(S^* \cdot \alpha, S^* \cdot \beta^*)$ is enough since we can revert the $(FJ)^*$ steps with $(FS)^*$ steps. Now, since α and β are local, $S^* \cdot \alpha$ and $S^* \cdot \beta$ are as well. Thus, we can limit ourselves to closing the Dirac distributions of D, and combine the reductions since γ, δ are compositional. We now need to close $(S^* \cdot \alpha, S^* \cdot \beta)$ when starting from some $[(1, a)]$. Note that the left (right) branch is then of the form $p_1 D_1 +\!\!+ \ldots +\!\!+ p_n D_n$ $(q_1 E_1 +\!\!+ \ldots +\!\!+ q_m E_m)$, where a reduces via α (β) to each D_i (E_j). We can apply our hypothesis to get a $C_{i,j}$ closing each D_i, E_j. By first splitting each branch appropriately, we can close them in $p_1 q_1 C_{1,1} +\!\!+ \ldots +\!\!+ p_1 q_m C_{1,m} +\!\!+ \ldots +\!\!+ p_n q_m C_{n,m}$. □

From this theorem, we get as corollaries several simplified criteria for confluence, applicable at the level of a particular distribution or to the whole system.

Criterion 4.8 (Dirac confluence) *If for every element a of D and distributions E, F such that $E \twoheadleftarrow^*_P [(1, a)] \twoheadrightarrow^*_P F$ there is a C such that $E \twoheadrightarrow^* C \twoheadleftarrow^* F$, then D is confluent.*

Proof. A corollary of Theorem 4.7, taking $\alpha = \beta = \gamma = \delta = \twoheadrightarrow^*_P$. □

Criterion 4.9 (Semi-confluence) *If for every element a of D and distributions E, F such that $a \mapsto E$ and $[(1, a)] \twoheadrightarrow^*_P F$ there is a C such that $E \twoheadrightarrow^* C \twoheadleftarrow^* F$, then D is semi-confluent for $\twoheadrightarrow_P/\!\approx$.*

Proof. A corollary of Theorem 4.7, taking $\alpha = \twoheadrightarrow_P$ and $\beta = \gamma = \delta = \twoheadrightarrow^*_P$. □

Criterion 4.10 (Diamond property) *If for every element a of D and distributions E, F such that $E \leftarrowtail a \mapsto F$ there is a C such that $E \twoheadrightarrow_{P/\approx} C \twoheadleftarrow_{P/\approx} F$, then D has the diamond property for $\twoheadrightarrow_P/\!\approx$.*

Proof. A corollary of Theorem 4.7, taking $\alpha = \beta = \gamma = \delta = \twoheadrightarrow_P$. □

Note that in all these criteria, we need not consider any equivalence in the peak, and can use them freely in the valley, both before and after evolving. Also, proving any of these criteria for every element a entails the confluence of the system.

Another common tool for proving confluence is switching the relation to another one with equal reflexive-transitive closure (and thus an equivalent confluence) but which might be easier to analyse. For distribution confluence, a similar switch is allowed, slightly simplified by Lemma 4.12.

Definition 4.11 *Given $\mapsto_1$ and $\mapsto_2$ over the same set A, if for every $a \mapsto_1 D$, we have $[(1, a)] \twoheadrightarrow^*{}_2\, D$ we say that $\mapsto_1$ is* simulated by *$\mapsto_2$.*

Lemma 4.12 *If $\mapsto_1$ is simulated by $\mapsto_2$, then $\twoheadrightarrow_1 \subseteq \twoheadrightarrow^*{}_2$. If both relations simulate each other, then $\twoheadrightarrow^*{}_1 = \twoheadrightarrow^*{}_2$, and their confluences are equivalent.*

Proof. The first part follows by case analysis on the reduction $\twoheadrightarrow_1$. The second part is trivial. $\square$

4.4 *Newman's lemma*

Newman's lemma [18] states that, for a strongly normalising system, local confluence and confluence are equivalent, yet we have remarked previously that $\twoheadrightarrow_P$ is never a strongly normalising relation. To get an analogue to Newman's lemma, we thus provide a specialized notion of strong normalisation.

Definition 4.13 *A infinite sequence D_i such that $D_1 \to D_2 \to D_3 \to \cdots$ is called an "infinite $\to$-chain" (of root D_1).*

Definition 4.14 (SN) *We call a distribution D "strongly normalising" when there is no infinite $\twoheadrightarrow^1_P$-chain of root D.*[3] *We call a PARS strongly normalising when every distribution is strongly normalising.*

There are indeed systems which do satisfy this requirement, and it is intuitively what one would expect. Now a probabilistic analogue to Newman's lemma can be obtained, following a proof style very similar to that of [13].

Definition 4.15 (LC) *We say that a distribution D is "locally confluent" when $E \twoheadleftarrow^1_P D \twoheadrightarrow^1_P F$ implies that there exists C such that $E \twoheadrightarrow^* C \twoheadleftarrow^* F$.*

Note that strong normalisation over Dirac distributions implies it for all distributions, and likewise for local confluence.

Lemma 4.16 (Newman's) *If $\mathcal{A} \models \mathrm{LC}$ and $\mathcal{A} \models \mathrm{SN}$, then $\mathcal{A} \models \mathrm{CR}$.*

Proof. We shall prove, by well-founded induction over $\twoheadrightarrow^1_P$, that every distribution is confluent. For a given distribution, it suffices to show that that any peak of proper evolutions can be closed by $\twoheadrightarrow^*$. Then, by Corollary 4.8 (and since $\twoheadrightarrow^*_P = \twoheadrightarrow^{1*}_P$), confluence follows. We want to close a diagram of shape $E \twoheadleftarrow^{1*}_P D \twoheadrightarrow^{1*}_P F$. If either of the branches is zero steps long, then we trivially conclude. If not, we can form the diagram on the right, completing the proof by local confluence and the induction hypotheses for E' and F'. $\square$

$$\begin{tikzcd}
& & D \arrow[dl, two heads, "P^1"'] \arrow[dr, two heads, "P^1"] & & \\
& E' \arrow[dl, two heads, "P^{1*}"'] \arrow[dr, dashed, two heads, "*"] & \text{LC} & F' \arrow[dl, dashed, two heads, "*"'] \arrow[dr, two heads, "P^{1*}"] & \\
E \arrow[dr, dashed, two heads, "*"'] & \text{IH} & C \arrow[dl, dashed, two heads, "*"] & & F \arrow[ddll, dashed, two heads, "*"] \\
& C' \arrow[dr, dashed, two heads, "*"'] & & \text{IH} & \\
& & C'' & &
\end{tikzcd}$$

5 Limit Distributions

In classical abstract rewriting, an element can either be non-normalising, weakly normalising or strongly normalising (corresponding to the situations where it *will not*, *may*, and *will* normalise, respectively). In probabilistic rewriting, the story is not as simple. Consider the following PARS, where b is a terminal element:

$$a \mapsto [(1/2, a), (1/2, b)]$$

[3] Note that infinite $(\twoheadrightarrow^1_P/\approx)$-chains always exist because of partial evolution.

Is a normalising? One could say "no" since, indeed, it does not have a finite maximal computation tree, as there is always some probability for the system to be in the non-terminal a state. However, such a probability will be made arbitrarily small by taking sufficient steps, and the distribution $[(1, b)]$ is reached *in the limit*. In this case a is called *almost surely terminating* [1]. Certainly, a desirable fact is that almost-surely-terminating elements have a unique final distribution. We will prove that distribution confluence guarantees such uniqueness.

We first introduce a notion of distance between *mathematical distributions*, i.e. normalised functions of type $A \to [0, 1]$. The reason we move away from list distributions is to allow for infinitely-supported limit distributions. We note with $[\![D]\!]$ the mathematical distribution obtained from the list distribution D (with the expected definition). We also extend definitions over mathematical distributions to list distributions by applying $[\![-]\!]$ where appropriate.

Definition 5.1 *Given D, E mathematical distributions, we define the distance between them as $d(D, E) = \sum_{a \in A} |D(a) - E(a)|$.*

Definition 5.2 (Limit of a sequence) *Given an infinite sequence of mathematical distributions $D_0, D_1, \ldots$ we say that L is a limit for the sequence if for every $\varepsilon > 0$, there exists $N > 0$ such that for all $i \geq N$, $d(D_i, L) < \varepsilon$.*

Note that this distance is the L^1 distance and the definition of limit is the usual one for metric spaces. It is then well known that limits for a given sequence are unique. We are interested in limits composed of terminal elements, representing a distribution of values. For that, the following definition is useful.

Definition 5.3 *For a mathematical distribution D, we define its "liveness" as the sum of weights for non-terminal elements. That is,* $\mathrm{Liv}(D) = \sum_{a \in \mathrm{dom}(\mapsto)} D(a)$

Note that the liveness of a list distribution cannot increase by evolution, and that $\mathrm{Liv}(D) = 0$ iff D is terminal. Moreover, since the normalised part of a distribution cannot evolve, liveness provides an upper bound on the possible distance to be attained by evolution, as the following lemma states.

Lemma 5.4 *If $D \twoheadrightarrow^* E$, then $d(D, E) \leq 2 \cdot \mathrm{Liv}(D)$.*

Proof. By Lemma 4.4 there exist D' and E' such that $D \approx D' \twoheadrightarrow_P^* E' \approx E$. Because of the equivalences, it suffices to show the result for D' and E'. Assume, without loss of generality, that $D' = D_l +\!\!+ D_t$, where all elements of D_l are not terminal, and all those of D_t are. Since parallel evolution is local and terminal elements cannot evolve, we have that $E' = E'' +\!\!+ D_t$ for some E''. Then, $d(D', E')$ is simply $d(D_l, E'')$. Note that $\mathrm{Liv}(D')$ is the weight of D_l and of E''. Since distance is bounded by total weight, it follows that it is at most $2 \cdot \mathrm{Liv}(D') = 2 \cdot \mathrm{Liv}(D)$. □

Now, we can extend our notion of uniqueness of terminal distributions to limit distributions of terminal elements, accounting for an infinite sequence of reductions.

Definition 5.5 (ULD) *A PARS $\mathcal{A}$ has "unique limit distributions" when for every D that is the root of two infinite $\twoheadrightarrow$-chains E_i and F_j with respective limits E_∞ and F_∞ terminal distributions, then $E_\infty = F_\infty$.*

Lemma 5.6 *If* $\mathcal{A} \models \mathrm{CR}$, *then* $\mathcal{A} \models \mathrm{ULD}$.

Proof. Take $\varepsilon > 0$. By the definition of limit, we know there are i, j such that $d(E_i, E_\infty) < {}^{\varepsilon}/_{6}$ and $d(F_j, F_\infty) < {}^{\varepsilon}/_{6}$. Since E_∞ and F_∞ are terminal, $\mathrm{Liv}(E_i)$ and $\mathrm{Liv}(F_j)$ must be less than ${}^{\varepsilon}/_{6}$. The distributions E_i and F_j are reachable by a finite amount of $\twoheadrightarrow$ steps, so by confluence there exists a distribution C such that $E_i \twoheadrightarrow^* C \twoheadleftarrow^* F_j$. From Lemma 5.4, we get that $d(E_i, C) < {}^{\varepsilon}/_{3}$ and likewise for F_j. From these four bounds and the triangle inequality we get that $d(E_\infty, F_\infty) < \varepsilon$. Since this is the case for any positive ε, $d(E_\infty, F_\infty)$ must be exactly 0, and therefore $E_\infty = F_\infty$. □

$$\begin{tikzcd}
 & & D \arrow[dl, two heads, "*"'] \arrow[dr, two heads, "*"] & & \\
 & E_i \arrow[dl, dotted, two heads, "{[0,\varepsilon/6)}"', "*"] \arrow[dr, two heads, "{[0,\varepsilon/3)}", "*"'] & & F_j \arrow[dl, two heads, "{[0,\varepsilon/3)}"', "*"] \arrow[dr, dotted, two heads, "{[0,\varepsilon/6)}", "*"'] & \\
E_\infty & & C & & F_\infty
\end{tikzcd}$$

6 Case Studies

6.1 An affine probabilistic λ-calculus: λ_1

In our introductory example, we used the term $(\lambda x.\ x - x)\ \square$ as an example of a non-confluent computation. There seem to be three ingredients needed for this failure of confluence of a term $(\lambda x.M)N$: (1) x appears free more than once in M (2) N has a non-Dirac terminal distribution (3) both call-by-name and call-by-value reductions are possible.

In this section we define a probabilistic λ-calculus, dubbed λ_1, that prevents the combination of these three features by providing two kinds of abstractions, one restricting duplication and one restricting evaluation order.[4] We show λ_1 to be confluent (by a diamond property), giving evidence that little more than affinity of probabilistic arguments is required to achieve a confluent probabilistic programming language.

The calculus is heavily based on the one defined in [23]. The set of *pre*-terms is given by the following grammar

$$M, N ::= x \mid MN \mid \lambda x.M \mid \lambda !x.M \mid !M \mid M \oplus_p N$$

where the main novelty is the probabilistic choice operator $\oplus_p$, for any real number p in the open interval $(0, 1)$. For an abstraction $\lambda x.M$, x must be *affine* in M, that is, M can have at most one free occurrence of x. If there is exactly one such occurrence, we say x is *linear* in M. Banged abstractions ($\lambda!$) have no such restriction. Affinity is enforced by a well-formedness judgment, whose definition is straightforward and which we thus omit. We work only with well-formed pre-terms, which form the set of terms. For the sake of brevity, given a term M and distribution $D = [(p_i, N_i)]$ we use the notation MD to represent the distribution $[(p_i, MN_i)]$, and similarly for all other syntactic constructs.

The operational semantics is provided as a PARS in Fig. 1. Terms of the form $!M$ do not reduce and are called *thunks*. A banged abstraction can only β-reduce

[4] For more expressivity, a third kind without either restriction, but forbidding probabilistic arguments, could be added. We do not deem this as interesting for the scope of this paper.

$$\text{R-}\beta\ \frac{}{(\lambda x.M)N \mapsto [(1, M[N/x])]} \qquad \text{R-}\beta!\ \frac{}{(\lambda !x.M)!N \mapsto [(1, M[N/x])]}$$

$$\text{R-}\oplus\ \frac{}{M \oplus_p N \mapsto [(p, M), (1-p, N)]}$$

$$\text{R-}\oplus\text{-L}\ \frac{M \mapsto D}{M \oplus_p N \mapsto D \oplus_p N} \qquad \text{R-}\oplus\text{-R}\ \frac{N \mapsto D}{M \oplus_p N \mapsto M \oplus_p D} \qquad \text{R-}\textsc{AppL}\ \frac{M \mapsto D}{MN \mapsto DN}$$

$$\text{R-}\textsc{AppR}\ \frac{N \mapsto D}{MN \mapsto MD} \qquad \text{R-}\lambda\ \frac{M \mapsto D}{\lambda x.M \mapsto \lambda x.D} \qquad \text{R-}\lambda!\ \frac{M \mapsto D}{\lambda !x.M \mapsto \lambda !x.D}$$

Fig. 1. Full semantics for λ_1

when applied to a thunk. This effectively implies that banged abstractions follow a fixed strategy (which is, morally, call-by-value until the argument is reduced to a thunk and call-by-name afterwards).[5]

To prove the diamond property for λ_1, we first need two substitution lemmas. When $D = [(p_i, a_i)]_i$, we write $D[M/x]$ for the distribution $[(p_i, a_i[M/x])]_i$. Similarly, $M[D/x]$ denotes $[(p_i, M[a_i/x])]_i$.

Lemma 6.1 *If $M \mapsto D$, then $M[N/x] \mapsto D[N/x]$.*

Proof. By induction on $M \mapsto D$. □

Lemma 6.2 *If $M \mapsto D$, and x is linear in N, then $N[M/x] \mapsto N[D/x]$.*

Proof. By induction on the well-formedness of N. □

Lemmas 6.1 and 6.2 are analogous to both statements of [23, Lemma 3.1]. Armed with both, we can prove the following theorem, which implies the diamond property.

Theorem 6.3 *If $D \leftarrowtail M \mapsto E$ then there exist C, C' such that $D \twoheadrightarrow_P C$ and $E \twoheadrightarrow_P C'$ with $C \approx C'$.*

Proof. By induction on the shape of $M \mapsto D$ and $M \mapsto E$. □

By this theorem and Corollary 4.10 we conclude that λ_1 is confluent, and thus enjoys both UTD and ULD.

6.2 A quantum λ-calculus: Q^*

The Q^* calculus [8] is a quantum programming language which models measurement, an inherently probabilistic operation. Reduction occurs between *configurations*, which are terms coupled with a quantum state, and which we will not detail further. Its semantics does not fix a strategy and, as λ_1, is also based on [23].

[5] We are thus adopting "surface reduction" only since "internal reductions" [23] hinder confluence.

Quantum variables in Q^* are linear (and not affine), so they cannot be duplicated nor discarded, as per the no-cloning [25] and no-erasure [19] properties of quantum physics. Reduction steps are paired with a *label* describing the reduction (e.g. which qubit was measured).

The authors prove a property called *strong confluence* which asserts that any two maximal (possibly infinite) computation trees with a common root have an equivalent normalized support (that is, the normalized part of both support distributions are equivalent) and, further, that any normal form appears in an equal amount of leaves on both trees.

As they note, this property is quite strong, and not enjoyed by the classical λ-calculus. Consider the term $(\lambda x.\lambda y.y)\Omega$. It has an infinite computation tree without any normal form (as the term reduces to itself in call-by-value) and also a finite tree with a single $\lambda y.y$ leaf, which of course does not have an equivalent normalized support. Note that the Q^* well-formedness judgment rejects this term as it is not linear.

To prove strong confluence, a crucial lemma called *quasi-one-step confluence* is proved, which is morally a diamond property but with slightly different behaviours according to the reductions taken. Reductions are distinguished between two sets, $\mathcal{N}$ and $\mathcal{K}$, and measurements of the form $\mathbf{meas}_r$. We will not describe these sets nor Q^*'s semantics (its full description is found in [8]), and will merely state the lemma. The notation $C \to^p_\alpha D$ means "C reduces to D with probability p via the label α"; and $C \to^p_\mathcal{N} D$ means $C \to^p_\alpha D$ for some $\alpha \in \mathcal{N}$ (idem $\mathcal{K}$).

Lemma 6.4 (Quasi-one-step Confluence for Q^* [7, Proposition 4]) *Let C, D, E be configurations and $C \to^p_\alpha D$, $C \to^s_\beta E$, then:*

- *If $\alpha \in \mathcal{K}$ and $\beta \in \mathcal{K}$, then either $D = E$ or there is F with $D \to^1_\mathcal{K} F$ and $E \to^1_\mathcal{K} F$.*
- *If $\alpha \in \mathcal{K}$ and $\beta \in \mathcal{N}$, then either $D \to^1_\mathcal{N} E$ or there is F s.t. $D \to^1_\mathcal{N} F$ and $E \to^1_\mathcal{K} F$.*
- *If $\alpha \in \mathcal{K}$ and $\beta = \mathbf{meas}_r$, then there is F with $D \to^s_{\mathbf{meas}_r} F$ and $E \to^1_\mathcal{K} F$.*
- *If $\alpha \in \mathcal{N}$ and $\beta \in \mathcal{N}$, then either $D = E$ or there is F with $D \to^1_\mathcal{N} F$ and $E \to^1_\mathcal{N} F$.*
- *If $\alpha \in \mathcal{N}$ and $\beta = \mathbf{meas}_r$, then there is F with $D \to^s_{\mathbf{meas}_r} F$ and $E \to^1_\mathcal{N} F$.*
- *If $\alpha = \mathbf{meas}_r$ and $\beta = \mathbf{meas}_q$ (with $r \neq q$), then there are $t, u \in [0,1]$ and an F such that $pt = su$, $D \to^t_{\mathbf{meas}_q} F$ and $E \to^u_{\mathbf{meas}_r} F$.*

From this lemma, the fact that there are no infinite $\mathcal{K}$ sequences, and a "probabilistic strip lemma", the authors prove strong confluence [8, Theorem 5.4].

For distribution confluence, a simpler proof can be obtained. After modelling Q^* as a PARS (roughly using successor distribution per label) we can readily reinterpret Lemma 6.4 to prove the diamond property for it (by Corollary 4.10). From this result, distribution confluence follows, and therefore also uniqueness of both terminal and limit distributions. Notably, neither the normalisation requirement for $\mathcal{K}$ nor the probabilistic strip lemma are needed for this fact.

Our obtained distribution confluence is similar, but neither weaker nor stronger than strong confluence. It is not weaker as distribution confluence guarantees that divergences of computations without any normal form can be joined, which strong confluence does not. It is also not stronger as it implies nothing of limit distributions that are not terminal, while it follows from strong confluence that they must coincide in their normalized part.[6]

7 Conclusions

We have studied the problem of showing that a probabilistic operational semantics is not affected by the choice of strategy. For this purpose, we provided a definition of confluence for probabilistic systems by defining a classical relation over distributions. We showed our property of distribution confluence to be appropriate as, in particular, it implies a uniqueness of terminal distributions, both for finite and infinite reductions, and gives an equational consistency guarantee.

We believe this development demonstrates that distribution confluence provides a reasonable "sweet spot" for proving the correctness of probabilistic semantics, as it provides the expected guarantees about execution while allowing tractable proofs. Concretely, the provided proofs for λ_1 and Q^* are in line with what one would expect for linear calculi.

The proof about Q^* also partially answers the conjecture posed in [8, Section 8] ("any rewriting system enjoying properties like Proposition 4 [our Lemma 6.4] enjoys confluence in the same sense as the one used here") positively. The answer is partial since distribution confluence is not strictly equivalent.

Looking ahead, there are several interesting directions to explore. Firstly, a study of confluence dealing with *terms* (and not just abstract elements) should provide more insights applicable to concrete languages. For terms, we expect concepts such as orthogonality to be of interest. Secondly, as a generalization, it seems possible to take distribution weights from any mathematical field and not only the positive reals. Even if interpreting such systems is not obvious, most of our results would hold: interestingly, we have not once assumed that weights actually represent probabilities. Finally, a quantitative notion of confluence could also be explored, where a distribution is considered confluent if any divergence of it can be joined "up to ε"; in particular, obtaining useful simplified criteria for said property seems difficult.

7.1 Related work

In [6], similar definitions of evolution and confluence are introduced. A subtle yet key difference with ours is that equivalent distributions are identified and there is no partial evolution. This means that evolution is not compositional; and in fact the diamond property over Dirac distributions does not extend over to all distributions. This was wrongly stated in an early version [5] of the paper and subsequently fixed.

In [11], a notion of confluence is defined and proven for an extension of λ_q [24] (a quantum λ-calculus) with measurements. The proposed property is basically

[6] It also does not imply the equality between the amount of leaves on each tree. This can in fact be recovered by removing the Split and Join rules and (straightforwardly) deriving criteria following §4. One can then conclude not only that there is the same amount but that they are paired with the same weights.

a confluence on computation trees, and the one we study in this paper is strictly weaker, yet sufficient for UTD and consistency.

In [8], already amply discussed, the introduced property is a strong confluence over maximal trees (either finite or infinite) which is very related, but neither weaker nor stronger than distribution confluence.

Finally, in [3] and subsequently in [15], a property of confluence is defined and studied over probabilistic rewriting systems which do not contain any non-determinism (i.e. where $\mapsto$ is a partial function). This is a very different notion of confluence, dealing with punctual final results instead of distributions, and with different applications. In this setting, distribution confluence trivially holds.

Acknowledgement

We thank Exequiel Rivas for reviewing early versions of this work and for many interesting discussions. We also thank both the anonymous reviewers and LSFA attendees for constructive and encouraging comments.

References

[1] Bournez, O. and F. Garnier, *Proving positive almost-sure termination*, in: J. Giesl, editor, *Proceedings of the 16th International Conference on Term Rewriting and Applications (RTA'05)*, Lecture Notes in Computer Science **3467** (2005), pp. 323–337.

[2] Bournez, O. and M. Hoyrup, *Rewriting logic and probabilities*, in: R. Nieuwenhuis, editor, *Rewriting Techniques and Applications*, Lecture Notes in Computer Science **2706**, 2003, pp. 61–75.

[3] Bournez, O. and C. Kirchner, *Probabilistic Rewrite Strategies. Applications to ELAN*, in: S. Tison, editor, *Proceedings of the 13th International Conference on Rewriting Techniques and Applications (RTA'02)*, Lecture Notes in Computer Science **2378** (2002), pp. 252–266.

[4] Church, A. and J. B. Rosser, *Some properties of conversion*, Transactions of the American Mathematical Society **39** (1936), pp. 472–482.

[5] Dal Lago, U., C. Faggian, B. Valiron and A. Yoshimizu, *The geometry of parallelism: Classical, probabilistic, and quantum effects*, arXiv:1610.09629v2 (2016).

[6] Dal Lago, U., C. Faggian, B. Valiron and A. Yoshimizu, *The geometry of parallelism: Classical, probabilistic, and quantum effects*, in: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, POPL 2017 (2017), pp. 833–845.
URL `http://doi.acm.org/10.1145/3009837.3009859`

[7] Dal Lago, U., A. Masini and M. Zorzi, *Confluence results for a quantum lambda calculus with measurements*, arXiv:0905.4567, extended version of [8].

[8] Dal Lago, U., A. Masini and M. Zorzi, *Confluence results for a quantum lambda calculus with measurements*, in: B. Coecke, P. Panangaden and P. Selinger, editors, *Proceedings of the 6th International Workshop on Quantum Physics and Logic (QPL'09)*, Electronic Notes in Theoretical Computer Science **270.2** (2011), pp. 251–261.

[9] Dal Lago, U. and M. Zorzi, *Probabilistic operational semantics for the lambda calculus*, RAIRO Theoretical Informatics and Applications **46** (2012), pp. 413–450.

[10] Di Pierro, A., C. Hankin and H. Wiklicky, *Probabilistic λ-calculus and quantitative program analysis*, Journal of Logic and Computation **15** (2005), pp. 159–179.

[11] Díaz-Caro, A., P. Arrighi, M. Gadella and J. Grattage, *Measurements and confluence in quantum lambda calculi with explicit qubits*, in: B. Coecke, I. Mackie, P. Panangaden and P. Selinger, editors, *Proceedings of the Joint 5th International Workshop on Quantum Physics and Logic and 4th Workshop on Developments in Computational Models (QPL/DCM'08)*, Electronic Notes in Theoretical Computer Science **270.1** (2011), pp. 59–74.

[12] Gordon, A., T. A. Henzinger, A. Nori and S. Rajamani, *Probabilistic programming*, in: J. Herbsleb, editor, *Proceedings of the 36th International Conference on Software Engineering. Session on Future of Software Engineering*, FOSE'14 (2014), pp. 167–181.

[13] Huet, G., *Confluent reductions: Abstract properties and applications to term rewriting systems: Abstract properties and applications to term rewriting systems*, Journal of the ACM **27** (1980), pp. 797–821.

[14] Jouannaud, J.-P. and H. Kirchner, *Completion of a set of rules modulo a set of equations*, in: *Proceedings of the 11th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '84 (1984), pp. 83–92.

[15] Kirkeby, M. and H. Christiansen, *Confluence and convergence in probabilistically terminating reduction systems*, in: *Proceedings of the 27th International Symposium on Logic-Based Program Synthesis and Transformation*, LOPSTR '17 (2017), pp. 33–46.

[16] Kozen, D., *Semantics of probabilistic programs*, Journal of Computer and System Sciences **22** (1981), pp. 328–350.

[17] Martínez, G., "Confluencia en sistemas de reescritura probabilista," Master's thesis, Universidad Nacional de Rosario, Argentina (2017), available at `https://dcc.fceia.unr.edu.ar/~gmartinez/tesina`.

[18] Newman, M. H. A., *On theories with a combinatorial definition of "equivalence"*, Annals of Mathematics **43** (1942), pp. 223–243.

[19] Pati, A. K. and S. L. Braunstein, *Impossibility of deleting an unknown quantum state*, Nature **404** (2000), pp. 164–165.

[20] Peterson, G. E. and M. E. Stickel, *Complete sets of reductions for some equational theories*, Journal of the ACM **28** (1981), pp. 233–264.

[21] Plotkin, G. D., *Call-by-name, call-by-value and the λ-calculus*, Theoretical Computer Science **1** (1975), pp. 125–159.

[22] Selinger, P. and B. Valiron, *A lambda calculus for quantum computation with classical control*, in: P. Urzyczyn, editor, *Proceedings of the 7th International Conference on Typed Lambda Calculi and Applications (TLCA'05)*, Lecture Notes in Computer Science **3461** (2005), pp. 354–368.

[23] Simpson, A., *Reduction in a linear lambda-calculus with applications to operational semantics*, in: J. Giesl, editor, *Proceedings of the 16th International Conference on Term Rewriting and Applications (RTA'05)*, Lecture Notes in Computer Science **3467** (2005), pp. 219–234.

[24] van Tonder, A., *A lambda calculus for quantum computation*, SIAM Journal on Computing **33** (2004), pp. 1109–1135.

[25] Wootters, W. K. and W. H. Zurek, *A single quantum cannot be cloned*, Nature **299** (1982), pp. 802–803.