A Note on Confluence
in Typed Probabilistic Lambda Calculi*

Rafael Romero

CONICET-Universidad de Buenos Aires.
Instituto de Ciencias de la Computación.
Buenos Aires, Argentina.

lromero@icc.fcen.uba.ar

Alejandro Díaz-Caro

Departamento de Ciencia y Tecnología.
Universidad Nacional de Quilmes.
Bernal, Buenos Aires, Argentina.

CONICET-Universidad de Buenos Aires.
Instituto de Ciencias de la Computación.
Buenos Aires, Argentina.

adiazcaro@icc.fcen.fcen.uba.ar

On the topic of probabilistic rewriting, there are several works studying both termination and confluence of different systems. While working with a lambda calculus modelling quantum computation, we found a system with probabilistic rewriting rules and strongly normalizing terms. We examine the effect of small modifications in probabilistic rewriting, affine variables, and strategies on the overall confluence in this strongly normalizing probabilistic calculus.

1 Introduction

When dealing with probabilistic lambda calculus, we can find two different sources of divergence.

- A *single redex* may reduce in two different ways via a probabilistic reduction.

- A term with *multiple redexes* and no strategy, could be reduced in different ways.

For example, we can consider a lambda calculus extended with a coin $\bigcirc$ reducing to 0 or 1 with probability $\frac{1}{2}$ each. Then, taking just the coin, we are in the first case of divergence. While taking, for example, $(\lambda x.\lambda y.yxx)\bigcirc$, we are in the second case, since we can either beta reduce, or reduce the coin.

There is no point in trying to achieve confluence in the first case: the coin is non-confluent by design. However, we can analyse the branching paths and verify that the probability of reducing to a particular term stays the same, regardless of the reduction sequence. This is what we call *probabilistic confluence*.
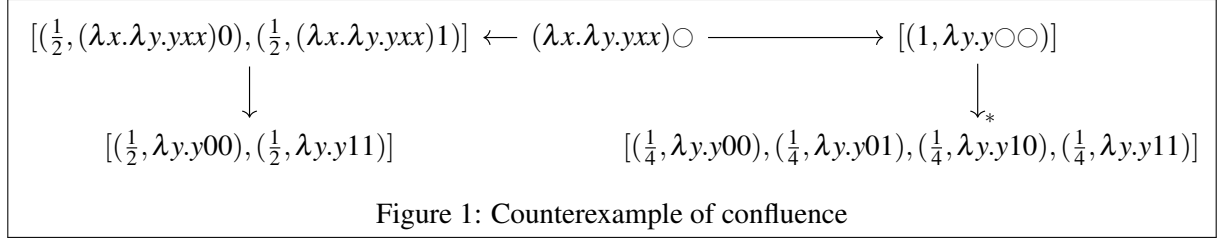
To study this kind of cases, Bournez and Kirchner developed the notion of PARS [3], later refined in [2]. Using the techniques described in [6, 7] we can define the rewriting rules over the distributions.

If we denote by $[(p_1,t_1),\ldots,(p_n,t_n)]$ the probability distribution where t_i has probability p_i, the possible reductions from the previous example are depicted in Figure 1. The resulting distributions are not only different, but also divergent, since in the left branch, the probability to arrive, for example, to $\lambda y.y01$ is 0, while it is one of the possible results in the right branch.

In this short paper we consider a simply typed lambda calculus extended with a coin, and show different possibilities for achieving some sort of confluence, without giving preference to any of them.

In Section 2 we introduce the calculus to be studied, without any restrictions either in the rewriting rules, or in the typing rules. As we argued above, this naive definition is not confluent (cf. Figure 1), unless a strategy is defined (in which case it becomes trivially probabilistically confluent, as will be discussed in Section 3.1). In Section 3.2 we show that we can achieve confluence by internalising the

$$[(\tfrac{1}{2}, (\lambda x.\lambda y.yxx)0), (\tfrac{1}{2}, (\lambda x.\lambda y.yxx)1)] \longleftarrow (\lambda x.\lambda y.yxx)\bigcirc \longrightarrow [(1, \lambda y.y\bigcirc\bigcirc)]$$

$$\downarrow \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow_*$$

$$[(\tfrac{1}{2}, \lambda y.y00), (\tfrac{1}{2}, \lambda y.y11)] \qquad\qquad [(\tfrac{1}{4}, \lambda y.y00), (\tfrac{1}{4}, \lambda y.y01), (\tfrac{1}{4}, \lambda y.y10), (\tfrac{1}{4}, \lambda y.y11)]$$

Figure 1: Counterexample of confluence

$$(\lambda x.t)r \to_1 t[r/x] \qquad \text{if } 1 \text{ then } t \text{ else } u \to_1 t \qquad \bigcirc \to_{\frac{1}{2}} 1$$
$$\text{if } 0 \text{ then } t \text{ else } u \to_1 u \qquad \bigcirc \to_{\frac{1}{2}} 0$$

$$\frac{t \to_p r}{\lambda x.t \to_p \lambda x.r} \qquad \frac{t \to_p r}{ts \to_p rs} \qquad \frac{t \to_p r}{st \to_p sr} \qquad \frac{t \to_p r}{\text{if } t \text{ then } u \text{ else } v \to_p \text{if } r \text{ then } u \text{ else } v}$$

$$\frac{u \to_p r}{\text{if } t \text{ then } u \text{ else } v \to_p \text{if } t \text{ then } r \text{ else } v} \qquad \frac{v \to_p r}{\text{if } t \text{ then } u \text{ else } v \to_p \text{if } t \text{ then } u \text{ else } r}$$

Table 1: Rewrite system for $\lambda_\circ$.

probabilistic reductions in the terms. In Section 3.3 we show that we can achieve probabilistic confluence by taking an affine-linear type system. Then, in Section 4, we show that we can relax the type system in an if-then-else branching, obtaining a probabilistic confluence result modulo a computational equivalence.

2   The $\lambda_\circ$ calculus

In this section we present $\lambda_\circ$ (read "lambda coin"), which is the simply typed lambda calculus extended with booleans (1 and 0), an if-then-else construction, and a coin $\bigcirc$. Terms are inductively defined by

$$t := x \mid \lambda x.t \mid tt \mid 1 \mid 0 \mid \text{if } t \text{ then } t \text{ else } t \mid \bigcirc$$

The rewrite system is given in Table 1. The rules $t \to_p r$ mean that t reduces with probability p in one step to r, where the sum of probabilities of reducing one redex is 1. In particular, every non-contextual rule has probability 1, since there is only one rule per redex, except for the coin, which reduces with probability $\frac{1}{2}$ to 0 and probability $\frac{1}{2}$ to 1. If $t \to_{p_1} r_1 \ldots \to_{p_n} r_n$ we may write $t \to^*_{\prod_{i=1}^n p_i} r_n$.

The reduction rules are intentionally as permissive as possible (even the branches of an if-then-else can be reduced) in order to analyse its (lack of) confluence. The type system is given in Table 2.

Strong normalization for this calculus follows trivially from Tait's proof for the simply typed λ-calculus, extended with booleans. The only reduction added is the probabilistic coin toss and it takes at most one step for each operator. Hence, using the rewriting over probabilistic distributions techniques from [6], we only need to show local confluence in order to achieve global confluence altogether. This is an adaptation of Newman's lemma for probabilistic calculi (see [6] for a longer discussion about probabilistic confluence).

Clearly, $\lambda_\circ$ is not confluent, as already seen in the introduction (see Figure 1). It is easy to see that these distributions represent different results.

$$A := \mathbb{B} \mid A \to A$$

$$\frac{}{\Gamma, x : A \vdash x : A} \text{ ax} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B} \to_i \qquad \frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash r : A}{\Gamma \vdash tr : B} \to_e$$

$$\frac{}{\Gamma \vdash 1 : \mathbb{B}} \text{ ax}_1 \qquad \frac{}{\Gamma \vdash 0 : \mathbb{B}} \text{ ax}_0 \qquad \frac{\Gamma \vdash t : \mathbb{B} \quad \Gamma \vdash u : A \quad \Gamma \vdash v : A}{\Gamma \vdash \text{if } t \text{ then } u \text{ else } v : A} \text{ if} \qquad \frac{}{\Gamma \vdash \bigcirc : \mathbb{B}} \text{ ax}_{\bigcirc}$$

Table 2: Type system for $\lambda_{\circ}$.

The divergence stems from three characteristics of the calculus: (1) Lack of a reduction strategy. (2) Probabilistic reductions. (3) Duplication of variables.

Removing just one of these elements renders the system confluent, however each modification comes with its own trade-off. We will examine each case, one by one, in the following section.

3   Removing the divergence sources

3.1   Defining a strategy

The definition of a strategy is the easiest modification. Choosing a reduction strategy makes all critical pairs disappear, since there is only one possible reduction rule to be applied for each term distribution. For example, in Figure 1 a *call-by-name* strategy would take the right path, where *call-by-value* would take the left one.

Ultimately the choice lies in how to interpret the duplication of variables. Reducing via call-by-name means that the probabilistic event is duplicated. Whereas, a call-by-value strategy duplicates the outcome of said event (see [4] for a discussion on this choice, and even an alternative combining both).

3.2   Internalising the probabilities

Following [5], we can modify the reduction on $\lambda_{\circ}$ to internalise the entire distribution of a term. In this particular case, every reduction has probability 1, and the coin toss deterministically reduces to its probability distribution. We write $t \oplus_p r$ for the probability distribution $[(p,t),(1-p,r)]$, $(t \oplus_p r) \oplus_q s$ for the probability distribution $[(qp,t),(q(1-p)r),(1-q,s)]$, etc. Then, we can consider the rewrite rule $\bigcirc \to 0 \oplus_{\frac{1}{2}} 1$. This idea is common in non-probabilistic settings as well, e.g., [1].

Following this approach brings confluence to the calculus, since every repetition of a probabilistic event rewrites to the same result, its distribution. For example, the two branches of Figure 1 become $(\lambda x.\lambda y.yxx)(0 \oplus_{\frac{1}{2}} 1)$ and $\lambda y.y \bigcirc \bigcirc$, both converging to $\lambda y.y(0 \oplus_{\frac{1}{2}} 1)(0 \oplus_{\frac{1}{2}} 1)$.

Although this is a valid solution, it forces us to consider every possible state of a program at the same time along with its probability of occurrence, making the management of the system more complex. Here we are dealing with a simple coin, but more involved calculi might have several different reductions, each with its own distribution. If not designed correctly, a language that holds every possible state in the probability distribution can easily become too cumbersome to be effective.

$$A := \mathbb{B} \mid A \to A$$

$$\frac{}{\Gamma, x : A \vdash x : A} \text{ ax} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B} \to_i \qquad \frac{\Gamma \vdash t : A \to B \quad \Delta \vdash r : A}{\Gamma, \Delta \vdash tr : B} \to_e$$

$$\frac{}{\Gamma \vdash 1 : \mathbb{B}} \text{ ax}_1 \qquad \frac{}{\Gamma \vdash 0 : \mathbb{B}} \text{ ax}_0 \qquad \frac{\Gamma \vdash t : \mathbb{B} \quad \Delta_1 \vdash u : A \quad \Delta_2 \vdash v : A}{\Gamma, \Delta_1, \Delta_2 \vdash \text{if } t \text{ then } u \text{ else } v : A} \text{ if} \qquad \frac{}{\Gamma \vdash \bigcirc : \mathbb{B}} \text{ ax}_{\bigcirc}$$

Table 3: Affine type system for $\lambda_{\circ}$ (different contexts are considered to be disjoint).

3.3 Affine variables

The last reasonable solution is to restrict duplication. One way to do this is by controlling the appearance of variables at the type system level, with an affine type system, see Table 3.

This type system solves the counterexample from Figure 1, since the considered term has no type in this system. In particular, we can prove the following property:

Lemma 3.1. *If* $r \to_p s$ *then* $t[r/x] \to_p t[s/x]$.

Notice that this property is not true in the unrestricted $\lambda_{\circ}$. For example, while $\bigcirc \to_{\frac{1}{2}} 1$, we have $(\lambda y.yxx)[\bigcirc/x] = \lambda y.y\bigcirc\bigcirc \to_{\frac{1}{4}}^* \lambda y.y11$.

The drawback in this approach is clear, there is a loss in expressivity. Of course, in some cases, this restriction is a desirable quality. For example, in quantum computing it may serve to avoid cloning qubits, a forbidden operation in quantum mechanics.

4 Computational confluence with a sub-affine type system

The solution considered in Section 3.3 seems quite extreme. In particular, using the same variable in different branches of an if-then-else construction does not actually duplicate it, since only one of those branches will remain. However, changing the rule if from Table 3 to if$_s$ given by

$$\frac{\Gamma \vdash t : \mathbb{B} \quad \Delta \vdash u : A \quad \Delta \vdash v : A}{\Gamma, \Delta \vdash \text{if } t \text{ then } u \text{ else } v : A} \text{ if}_s$$

breaks confluence anyway. We call this calculus "sub-affine". Consider the following example. Let $\text{not} = \lambda x.\text{if } x \text{ then } 0 \text{ else } 1$, then

Note that terms in both distributions are in normal form. The two paths are syntactically divergent, however the resulting programs share the same behaviour under the same inputs. If we were to apply the resulting abstractions to 0 or to 1, both paths would yield $[(1/2,0);(1/2,1)]$. Therefore, these distributions are semantically confluent, they are not the same terms but they *represent* the same function.

We can formalise this notion for the sub-affine calculus as follows. Let $C := \Diamond \mid Cv$ be an elimination context, where $\Diamond$ is called "placeholder" and v is a normal closed term. We write $C\langle t \rangle$ for $C[t/\Diamond]$. Notice that $C\langle t \rangle$ is a term. We say that C is an elimination context of A, written C^A, if for all $\vdash t : A$, we have $\vdash C\langle t \rangle : \mathbb{B}$. That is, it applies t until it reaches the basic type $\mathbb{B}$.

The computational equivalence is defined as follows.

Definition 4.1. Let $D_1 = [(p_i, t_i)]_i$ and $D_2 = [(q_j, r_j)]_j$ be two distributions of terms, all closed of type A. Then, we say that these distributions are computationally equivalent (notation $D_1 \equiv D_2$) if for all C^A we have $C\langle t_i \rangle \to_{u_{ik}}^* b_{ik}$ and $C\langle r_j \rangle \to_{s_{jh}}^* c_{jh}$ such that the b_{ik} and c_{jh} are in normal form, and $[(p_i u_{ik}, b_{ik})]_{ik} \sim [(q_j s_{jh}, c_{jh})]_{jh}$, where $\sim$ denotes the equality on distributions.

The previous definition means that two distributions are computationally equivalent if by applying the resulting terms to all the possible inputs, they produce the same probability distribution of results. Notice that the definition is not assuming confluence.
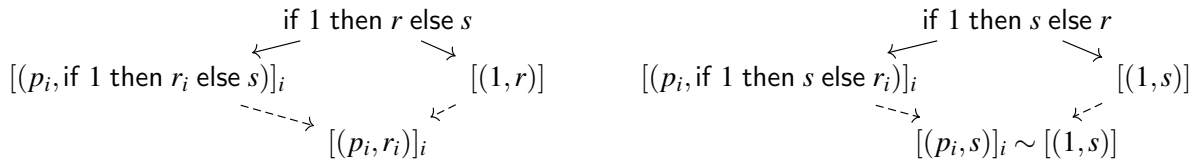
Then, we can prove that the confluence modulo computational equivalence of the sub-affine calculus (Theorem 4.4). We need the following two lemmas, which follow by straightforward induction.

Lemma 4.2. *Let $y \notin \mathsf{FV}(r)$, then $t[q/y][r/x] = t[r/x][q[r/x]/y]$.*   □
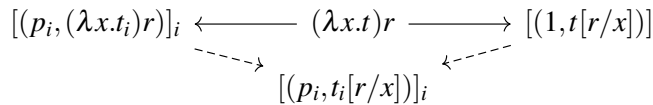
Lemma 4.3. *Let t reduce to the distribution D, then $t[r/x]$ reduces to $D[r/x]$.*   □

Theorem 4.4 (Computational confluence). *Let $\vdash t : A$ in the sub-affine calculus. If t reduces to the distribution D_1 and to the distribution D_2, then $D_1 \equiv D_2$.*

Proof. We only consider the six critical pairs, four derived from the if-then-else construction and two from the classical lambda calculus. Non-critical pairs are trivially probabilistically confluent.

$$
\begin{array}{ccc}
 & \text{if } 1 \text{ then } r \text{ else } s & \\
 \swarrow & & \searrow \\
[(p_i, \text{if } 1 \text{ then } r_i \text{ else } s)]_i & & [(1, r)] \\
 \dashrightarrow & & \dashleftarrow \\
 & [(p_i, r_i)]_i &
\end{array}
\qquad
\begin{array}{ccc}
 & \text{if } 1 \text{ then } s \text{ else } r & \\
 \swarrow & & \searrow \\
[(p_i, \text{if } 1 \text{ then } s \text{ else } r_i)]_i & & [(1, s)] \\
 \dashrightarrow & & \dashleftarrow \\
 & [(p_i, s)]_i \sim [(1, s)] &
\end{array}
$$

The symmetrical cases with 0 close in a similar way. The fifth critical pair closes by Lemma 4.3.

$$
[(p_i, (\lambda x.t_i)r)]_i \longleftarrow (\lambda x.t)r \longrightarrow [(1, t[r/x])]
$$
$$
\dashrightarrow \quad [(p_i, t_i[r/x])]_i \quad \dashleftarrow
$$

The last critical pair requires a more thorough analysis.

$$
[(p_i, (\lambda x.t)r_i)]_i \longleftarrow (\lambda x.t)r \longrightarrow [(1, t[r/x])]
$$
$$
\dashrightarrow \quad [(p_i, t[r_i/x])]_i
$$
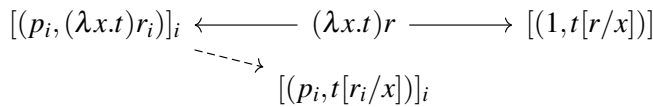
We must prove that for all C^A and for all i, there exists b_{ij} with $C\langle t[r_i/x] \rangle \to_{q_{ij}}^* b_{ij}$ and c_k with $C\langle t[r/x] \rangle \to_{s_k}^* c_k$ such that $[p_i q_{ij}, b_{ij}]_{ij} \sim [s_k, c_k]_k$. It is enough to take only one path to $[p_i q_{ij}, b_{ij}]_{ij}$ and to $[s_k, c_k]_k$ according to the definition of $\equiv$. If $x \notin \mathsf{FV}(t)$, then $C\langle t[r/x] \rangle = C\langle t \rangle = C\langle t[r_i/x] \rangle$. If x appears once in t, $C\langle t[r/x] \rangle \to_{p_i} C\langle t[r_i/x] \rangle$. If x appears more than once in t, it appears at most 2^n times (where n is the number of if-then-else constructions), so we can proceed by induction on n.

- If $n = 1$, then there is one if-then-else, say if c then s_1 else s_2 and x appears both in s_1 and in s_2. Therefore, there are three cases:
    - c is 0 or 1, then, for $j = 0$ or $j = 1$, we have $C\langle t[r/x]\rangle \rightarrow_1^* C\langle s_j[r/x]\rangle \rightarrow_{p_i}^* C\langle s_j[r_i/x]\rangle$. Notice that $C\langle t[r_i/x]\rangle \rightarrow_1 C\langle s_j[r_i/x]\rangle$.
    - $c \rightarrow_{q_j}^* j$ with $j = 0, 1$, so we have $C\langle t[r/x]\rangle \rightarrow_{q_j}^* C\langle s_j[r/x]\rangle \rightarrow_{p_i}^* C\langle s_j[r_i/x]\rangle$. That is, we arrived to the distribution $[(q_j p_i, s_j[r_i/x])]_{ij}$. Notice that since $C\langle t[r_i/x]\rangle \rightarrow_{q_j} C\langle s_j[r_i/x]\rangle$, we arrive to the same distribution in the right branch of this critical pair.
    - c is an open term, hence not reducing to a constant 0 or 1. In such a case, since $t[r/x]$ is a closed term, it means that the if-then-else construction in t is under a lambda-abstraction, therefore, we must beta-reduce first, either with the argument in t, or with an external argument given by the context C. We can repeat the same process until we get one of the previously treated cases (that is, at some point, the if-then-else becomes a redex).

- If $n > 1$, we proceed as the previous case, reducing the if-then-else first, which reduces n and so the induction hypothesis applies. □

5 Conclusion

In this paper we have analysed the different possibilities to transform a simple probabilistic calculus into a (probabilistically / computationally) confluent calculus. The main contribution on this note is Theorem 4.4, which proves that we can relax the affinity restriction on "non-interfering" paths. This technique has been considered in the first author master's thesis [8] to prove the computational confluence of the quantum lambda calculus λ_ρ [5].

References

[1] S. Alves, B. Dundua, M. Florido & T. Kutsia (2018): *Pattern-based calculi with finitary matching*. *Logic Journal of the IGPL* 26, pp. 203–243, doi:10.1093/JIGPAL/jzx059.

[2] O. Bournez & F. Garnier (2005): *Proving Positive Almost-sure Termination*. In J. Giesl, editor: *Term Rewriting and Applications (RTA 2005)*, *Lecture Notes in Computer Science* 3467, Springer, pp. 323–337, doi:10.1007/978-3-540-32033-3_24.

[3] O. Bournez & C. Kirchner (2002): *Probabilistic Rewrite Strategies. Applications to ELAN*. In S. Tison, editor: *Rewriting Techniques and Applications (RTA 2002)*, *Lecture Notes in Computer Science* 2378, Springer, pp. 252–266, doi:10.1007/3-540-45610-4_18.

[4] U. Dal Lago, G. Guerrieri & W. Heijltjes (2020): *Decomposing Probabilistic Lambda-Calculi*. In J. Goubault-Larrecq & B. König, editors: *Foundations of Software Science and Computation Structures (FoSSaCS 2020)*, *Lecture Notes in Computer Science* 12077, Springer, pp. 136–156, doi:10.1007/978-3-030-45231-5_8.

[5] A. Díaz-Caro (2017): *A lambda calculus for density matrices with classical and probabilistic controls*. In B.-Y. E. Chang, editor: *Programming Languages and Systems (APLAS 2017)*, *Lecture Notes in Computer Science* 10695, Springer, pp. 448–467, doi:10.1007/978-3-319-71237-6_22. Available at arXiv:1705.00097.

[6] A. Díaz-Caro & G. Martínez (2018): *Confluence in probabilistic rewriting*. In S. Alves & R. Wassermann, editors: *Logical and Semantic Frameworks with Applications (LSFA 2017)*, *Electronic Notes in Teoretical Computer Science* 338, Elsevier, pp. 115–131, doi:10.1016/j.entcs.2018.10.008.

[7] C. Faggian (2019): *Probabilistic Rewriting: Normalization, Termination, and Unique Normal Forms*. In H. Geuvers, editor: *Formal Structures for Computation and Deduction (FSCD 2019)*,

Leibniz International Proceedings in Informatics (LIPIcs) 131, Schloss Dagstuhl, pp. 19:1–19:25, doi:10.4230/LIPIcs.FSCD.2019.19.

[8] R. Romero (2020): *Una extensión polimórfica para los λ-cálculos cuánticos $λ_ρ$ y $λ_ρ^\circ$*. Master's thesis, Universidad de Buenos Aires, Argentina. Available at `http://dc.sigedep.exactas.uba.ar/media/academic/grade/thesis/presentacion.pdf`.