My very first steps in Rewriting Theory

Paul-André Melliès

October 24, 2012

Chapter 1

Abstract Rewriting Systems

The original motivation for rewriting is to provide decision procedures for equational theories. The idea is to orient the equations and to rewrite the terms to a canonical form. Two syntactic words are then equal if and only if they are rewritten to the same canonical form. Section 1.1 presents the word problem on monoids and explains how to solve it with Rewriting. Chapter 2 then introduces two historic rewriting systems: the λ-calculus which is not normalising, and the calculus of Petri nets which is not confluent.

1.1 Monoids

A **monoid** is a set M equipped with a law of composition $M \times M \to M$ which is associative, and a unit element. The free monoid on a set Σ, noted Σ^*, is the set of **words** or finite sequences of elements of Σ. Composition in Σ^* is defined by concatenation and unit is the empty word e.

A **presentation** of a monoid M consists of

1. a set Σ of **generators**,
2. a set $\mathcal{R}$ of **relations** $r \approx s$ where r and s are words in Σ^*

such that M is (isomorphic to) the quotient of Σ^* by the congruence associated to $\mathcal{R}$. A **rewriting system** is a presentation $(\Sigma, \mathcal{R})$ where every relation is explictly oriented and considered as a **rewrite rule**.

Example. The monoid M_4 of symmetries of the square[1] is presented by

$$\Sigma = \{\sigma, \tau\} \qquad \text{and} \qquad \mathcal{R} = \{\sigma^4 \approx e, \tau^2 \approx e, \tau\sigma \approx \sigma^3\tau\}$$

[1] In fact, M_4 is not only a monoid, it is a group.

M_4 contains 8 elements which can be enumerated as:

$$e, \sigma, \sigma^2, \sigma^3, \tau, \sigma\tau, \sigma^2\tau, \sigma^3\tau \tag{1.1}$$

One way to decide the equality of two arbitrary words u and v in Σ^* is to orient the relations in $\mathcal{R}$ as follows:

$$\sigma^4 \to e \qquad \tau^2 \to e \qquad \tau\sigma \to \sigma^3\tau$$

and to rewrite the words u and v so as to obtain two words $[u]$ and $[v]$ among the eight **canonical** words of (1.1) ; then to compare the two canonical forms $[u]$ and $[v]$. For instance, if $u = \tau\sigma^2\tau\sigma$ and $v = \tau\sigma\tau$, the series of rewriting steps on u and v:

$$u = \tau\sigma^2\tau\sigma \to \sigma^3\tau\sigma\tau\sigma \to \sigma^6\tau^2\sigma \to \sigma^2\tau^2\sigma \to \sigma^3 = [u]$$

$$v = \tau\sigma\tau \to \sigma^3\tau^2 \to \sigma^3 = [v]$$

shows that $u \approx \sigma^3$ and $v \approx \sigma^3$ in $(\Sigma, \mathcal{R})$, hence that the two symmetries $\tau\sigma^2\tau\sigma$ and $\tau\sigma\tau$ are equal in M_4.

If we try to understand why the procedure for deciding the word equality in M_4 succeeds, we see two reasons:

1. the rewriting procedure terminates: from every word $u \in \Sigma^*$ there is a sequence of rewrite steps to a word in **normal form**, where by normal form we mean a word on which no rewrite rule operates,

2. The normal form computed from a word u is uniquely determined by u. Why? Clearly, the canonical words are the only normal forms of the rewriting system $(\Sigma, \mathcal{R})$. Then, if a word u had *two* different normal forms $[u]_1$ and $[u]_2$ obtained by two different computations $u \to \cdots \to [u]_1$ and $u \to \cdots \to [u]_2$, the two normal forms $[u]_1$ and $[u]_2$ would be canonical *and* equal in M_4, which is impossible.

The properties 1. and 2. are called **normalisation** and **confluence** in the litterature. Next section shows how to formalise them in an abstract setting.

1.2 Abstract Rewriting Systems

An **abstract rewriting system** is a set A with a binary relation $\to$. We write $a \to b$ when $a, b \in A$ are related by $\to$. The relation $\overset{*}{\to}$ is the transitive reflexive closure of $\to$ and $\approx$ is the equivalence relation corresponding to $\to$, or equivalently the symmetric closure of $\overset{*}{\to}$.

Example. The abstract rewriting system $(\Sigma^*, \to)$ interpreting the presentation of M_4 in Section 1.1 consists of the monoid Σ^* considered as a set and the relation $\to$ relating words $uLv \to uRv$ precisely when $L \to R$ is one of the three rewriting rules. The relation $\approx$ is exactly the congruence which quotients Σ^* as M_4.

An element a of $(A, \to)$ is a **normal form** when there is no $b \in A$ such that $a \to b$. We write **nf** the set of normal forms. To every $a \in A$ is associated the set $\mathbf{nf}_a$ defined as

$$\mathbf{nf}_a = \{b \in \mathbf{nf} \mid a \xrightarrow{*} b\}$$

An abstract rewriting system $(A, \to)$ is **normalising** precisely when

$$\forall a \in A, \mathbf{nf}_a \neq \emptyset$$

Exercise*. Prove that the presentation $(\Sigma^*, \to)$ of M_4 is normalising. ■

An abstract rewriting system is **strongly normalising** when there is no infinite sequence

$$a_1 \to a_2 \to \cdots \to a_n \to a_{n+1} \to \cdots$$

Equivalently, the transitive relation $\xrightarrow{*}$ is a strict **well-founded** order.

1.3 Confluence

An abstract rewriting system is **confluent** or **Church-Rosser** when for every $(a, b, c) \in A^3$ such that $b \xleftarrow{*} a \xrightarrow{*} c$, there exists $d \in A$ such that $b \xrightarrow{*} d \xleftarrow{*} c$.

$$\forall (a,b,c) \in A^3 \qquad \begin{array}{ccc} a & \xrightarrow{*} & b \\ {\scriptstyle *}\downarrow & & \\ c & & \end{array} \qquad \Rightarrow \quad \exists d \in A, \qquad \begin{array}{ccc} a & \xrightarrow{*} & b \\ {\scriptstyle *}\downarrow & & \downarrow{\scriptstyle *} \\ c & \xrightarrow[*]{} & d \end{array}$$

Figure 1.1: The confluence or Church-Rosser property

Exercise. Prove that confluence implies that for every $a \in A$, the set $\mathbf{nf}_a$ contains at most one normal form. ■

Confluence may be difficult to prove on a particular rewriting system. We review two cases when confluence derives from very simple properties: strong confluence in the one case, strong normalisation and local confluence in the other case.

1.3.1 Confluence from strong confluence

An abstract rewriting system $(A, \to)$ is **strongly confluent** when for every $(a, b, c) \in A^3$ such that $b \leftarrow a \to c$, there exists $d \in A$ such that $b \to d \leftarrow c$.

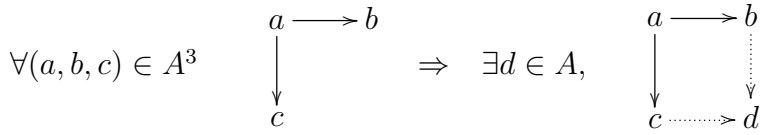


Figure 1.2: The strong confluence property

Proposition 1 *Assume that $(A, \to)$ is strongly confluent. Then $(A, \to)$ is confluent.*

Proof. easy induction on the size of the two rewriting paths defining the confluence diagram. ∎

However, the following exercise shows that the proposition above does not treat the monoidal presentation of M_4.

Exercise. Show that $(\Sigma^*, \to)$ is not strongly confluent. ∎

This leads us to consider a weaker notion of confluence: local confluence.

1.3.2 Confluence from local confluence

An abstract rewriting system $(A, \to)$ is **locally confluent** when for every $(a, b, c) \in A^3$ such that $b \leftarrow a \to c$, there exists $d \in A$ such that $b \overset{*}{\to} d \overset{*}{\leftarrow} c$.

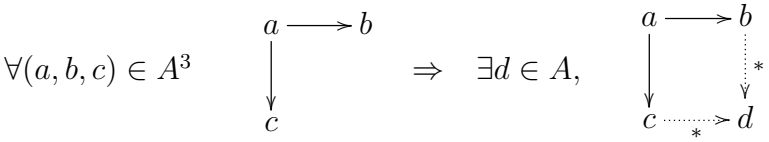


Figure 1.3: The local confluence property

Exercise. Show that $(\Sigma^*, \to)$ is locally confluent. ∎

Unfortunately, local confluence does not imply confluence. An example of a not confluent yet locally confluent rewriting system is the λ-calculus with

surjective pairing, see Klop's survery[2] However, a locally confluent rewriting system $(A, \to)$ is confluent when it verifies a finiteness condition on its reduction sequence.

Proposition 2 (Newman) *Assume that $(A, \to)$ is locally confluent and strongly normalising. Then $(A, \to)$ is confluent.*

Proof. Every nonempty set in a well-founded order has a minimal element. Let us call *confluent* any $a \in A$ such that whenever $b \overset{*}{\leftarrow} a \overset{*}{\to} c$ for some b and c, there exists $d \in A$ such that $b \overset{*}{\to} d \overset{*}{\leftarrow} c$. Suppose that the locally confluent and strongly normalising $(A, \to)$ is not confluent. Then the set $N \subset A$ of non confluent elements is nonempty and in particular contains a $\overset{*}{\to}$-minimal element a. In particular there are two elements b and c such that $b \overset{*}{\to} d \overset{*}{\leftarrow} c$ but which cannot be rewritten as $b \overset{*}{\to} d \overset{*}{\leftarrow} c$ for any element $d \in A$.

We reason on the number of rewrite steps in $a \overset{*}{\to} b$ and $a \overset{*}{\to} c$, which is non null in both cases. We suppose without loss of generality that $a \to b' \overset{*}{\to} b$ and $a \to c' \overset{*}{\to} c$. By minimality of $a \in N$ the elements b' and c' are not in N, thus they are confluent. By local confluence and $b' \leftarrow a \to c'$ there exists an element $e \in A$ such that $b' \overset{*}{\to} e \overset{*}{\leftarrow} c'$. By confluence of b' and $b \overset{*}{\leftarrow} b' \overset{*}{\to} e$, there exists $f \in A$ such that $b \overset{*}{\to} f \overset{*}{\leftarrow} e$. Symmetrically, there exists $g \in A$ such that $e \overset{*}{\to} g \overset{*}{\leftarrow} c$. By confluence of e (which follows minimality of $a \in N$), there exists $d \in A$ such that $f \overset{*}{\to} d \overset{*}{\leftarrow} g$. We conclude $b \overset{*}{\to} d \overset{*}{\leftarrow} c$, a contradiction. The strongly normalising and locally confluent $(A, \to)$ is therefore confluent. ∎

Exercise*. Prove that the presentation $(\Sigma^*, \to)$ of M_4 is strongly normalising. Use Newman's lemma to prove that $(\Sigma^*, \to)$ is confluent. Conclude that the word problem in M_4 is decidable. ∎

[2] **Jan Willem Klop.** Term Rewriting Systems. *Handbook of Logic in Computer Science*, Volume 2, in S. Abramsky, Dov M. Gabbay, T.S.E. Maibaum, editors, Oxford Science Publications, 1992.

Chapter 2

Rewriting graphs

We have seen in Section 1.2 how to derive confluence from strong confluence or from local confluence + strong normalisation. We obtain on the way that the presentation of M_4 in Section 1.1 is strongly normalising and confluent, hence decidable. But Rewriting Theory is not only concerned with calculi which are normalising and confluent. We present two calculi in this section: the pure λ-calculus which is confluent but not normalising and the calculus of Petri nets which being non deterministic is not confluent.

2.1 Rewriting graph

A *rewriting graph* $\mathcal{G}$ is a quadruple $(V, E, \partial_0, \partial_1)$ where V is a set of **vertices**, E is a set of **edges**, $\partial_0 : V \longrightarrow E$ and $\partial_1 : V \longrightarrow E$ are total functions indicating the **source** and **target** vertex of any edge. We write $M \xrightarrow{u} N$ when $\partial_0 u = M$ and $\partial_1 u = N$.

A **path** in a graph $\mathcal{G}$ is a sequence

$$f = (M_1, u_1, M_2, ..., M_m, u_m, M_{m+1}) \tag{2.1}$$

where $M_i \xrightarrow{u_i} M_{i+1}$ for every $i \in [1...m]$. The **length** $|f|$ of the path f in (2.1) is m, which is the number of edges the path contains. When $m = 0$, the path (M_1) is said to be **empty** ; we write $(M_1) = \mathbf{id}_{M_1}$. Two paths $(M_1, u_1, ..., u_m, M_m)$ and $(N_1, v_1, ..., v_n, N_n)$ are coinitial (resp. cofinal) when $M_1 = N_1$ (resp. $M_m = N_n$).

2.2 The lambda-calculus

The λ-calculus was invented then promoted by Alonzo Church as the calculus of functional evaluation.

Suppose given an infinite set **var** of **variables**. The set of **λ-trees** is constructed by induction:

1. every variable $x \in \mathbf{var}$ is a λ-tree,

2. if M and N are λ-trees, then MN is a λ-tree,

3. if M is a λ-tree, then $\lambda x.M$ is a λ-tree, where λx is a node λ labelled with any variable $x \in \mathbf{var}$.

The intended meaning of the λ-tree $(\lambda x.M)$ is the function $x \mapsto M$ which maps any argument N to the λ-tree obtained by carefully replacing with N every variable x appearing in M. For example the λ-tree $\lambda x.x$ stands for the identity $x \mapsto x$. Since $\lambda y.y$ also stands for the identity, the two λ-trees $\lambda x.x$ and $\lambda y.y$ should be identified.

Two λ-trees M and N are **α-equivalent**, $M \sim_\alpha N$, when

1. M and N are the same variables, or

2. $M = M_1 M_2$ and $N = N_1 N_2$ with $M_1 \sim_\alpha N_1$ and $M_2 \sim_\alpha N_2$, or

3. $M = \lambda x.M_1$ and $N = \lambda y.N_1$ with $M_1 \sim_\alpha N_2$ where N_2 is the result of replacing every variable y and node λy in N_1 by x and λx respectively.

The relation $\sim_\alpha$ is an equivalence relation whose classes of equivalence are called **λ-terms**. So, a λ-term M is best pictured as the λ-graph obtained from the λ-tree by linking each λx-node to the variables x it binds, and then by removing the variable labels on λ nodes and bound variables. Here, variable means **occurrence** of a variable, and a node λx **binds** (the occurrence of) a variable x in the λ-tree $\lambda x.M$ when $\lambda x.M \sim_\alpha \lambda y.N$ implies that x is replaced by y in N.

The λ-calculus operates on λ-terms. Before we introduce its only reduction rule, the β-rule, we need the operation of **substitution** defined on two λ-terms M and P:

1. $x[x := P] \stackrel{\Delta}{=} P$ and $y[x := P] \stackrel{\Delta}{=} y$ if $x \neq y$,

2. $MN[x := P] \stackrel{\Delta}{=} M[x := P]N[x := P]$,

3. $(\lambda z.M)[x := P] \stackrel{\Delta}{=} \lambda z.(M[x := P])$ if the variable $z \neq x$ does not appear anywhere in P.

It is easy to check that the definition of the λ-term $M[x := P]$ does not depend on the λ-trees M and P but only on the λ-terms they define.

The λ-calculus contains only one reduction rule: the β-rule.

$$(\lambda x.M)P \to M[x := P]$$

The λ-calculus is very expressive. It allows to construct **duplicators** like $\Delta = (\lambda x.xx)$ which duplicates its argument P in two copies:

$$\Delta P \to PP$$

It also allows to construct **erasers** like $(\lambda x.y)$ which does not use its argument P and erases it:

$$(\lambda x.y)P \to y$$

Two duplicators may combine in a λ-term whose computation **loops**:

$$\Delta\Delta = (\lambda x.xx)(\lambda y.yy) \to \Delta\Delta \to \cdots$$

This shows in particular that the λ-calculus is not normalising since $\mathbf{nf}_{\Delta\Delta}$ is empty. This has quite unexpexted consequences. For instance, an eraser like $K = (\lambda x.\lambda y.x)$ eliminates one of the looping procedures:

$$(\lambda x.\lambda y.x)a(\Delta\Delta) \to (\lambda y.a)(\Delta\Delta) \to a$$

This successful reduction path illustrates the fact that some λ-terms like $Ka(\Delta\Delta)$ have a normal form but may be computed for ever. We will see later how to compute successfully any λ-term, thus avoiding all sort of loops.

For the moment, let us only say that despite non termination, Church and Rosser have shown that the λ-calculus is confluent and therefore that a λ-term has at most one normal form. Chapter 4 is devoted to this fundamental result and to its proof.

2.2.1 Petri nets

Petri nets form another important class of Rewriting Systems. A Petri net is a quadruple $N = (P, T, \boldsymbol{pre}, \boldsymbol{post})$ where P is the set of **places** and T is the set of **transitions** of the net. A **state** of the net N is a set of elements of P. To every transition t the functions $\boldsymbol{pre}, \boldsymbol{post} : T \longrightarrow S$ associate a **pre-condition** $\boldsymbol{pre}(t)$ and a **post-condition** $\boldsymbol{post}(t)$ in the set of states S.

To every Petri net is associated a rewriting relation $\to$ between its states. This relation describes the possible transformations of the net through time. In fact, this leads to the definition of an abstract rewriting system $(A, \to)$ where:

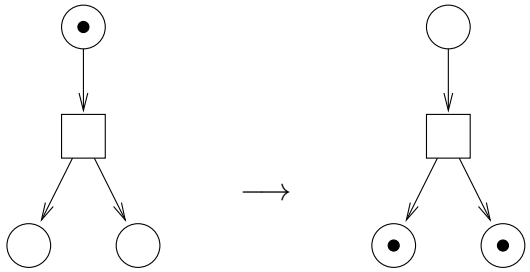


Figure 2.1: An example of rewriting step $s \to s'$

1. A is the set of states of N,

2. $s \to s'$ when there exists a set s_0 such that $s = s_0 \uplus \boldsymbol{pre}(t)$ and $s' = s_0 \uplus \boldsymbol{post}(t)$, letting $\uplus$ be the disjoint set union.

There is a standard graphical representation of Petri nets. Places are represented as circles and transition by boxes. The function t is represented by oriented arcs between circles and boxes, so that there are as many arcs from p to t (and from t to q) as there are occurrences of p in $\boldsymbol{pre}(t)$ (and of q in $\boldsymbol{post}(t)$). Every state of N is represented by the corresponding tagging of **tokens** in the places, see figure 2.2.1 for an illustration. Many familiar concepts of computer science may be expressed using Petri nets, like

1. **concurrency**, when two unrelated transitions t_1 and t_2 fire $s \to s_1$ and $s \to s_2$. Unrelated means that $\boldsymbol{pre}(t_1) \cap \boldsymbol{pre}(t_2)$ is empty. Observe that in that case there is a diamond diagram $s_1 \to s'$ and $s_2 \to s'$ for some state s'.

2. **causality** when a token in p is fired to a place q which enables a transition t, see figure 2.2.1,

3. **conflicts** when two different transitions have a source place or a target place in common, see figure 2.2.1.

Other interesting configurations are presented in figure 2.2.1.

Exercises

In the following exercises, $(A, \to)$ denotes an abstract rewriting system.

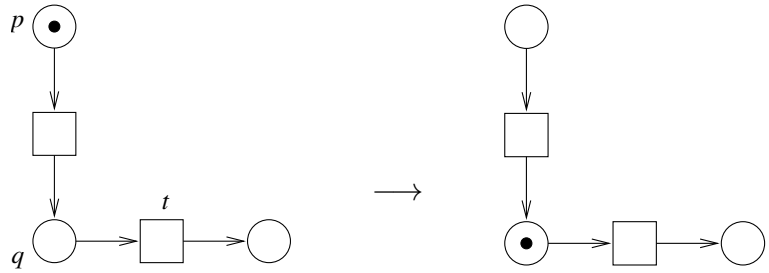


Figure 2.2: An example of causality

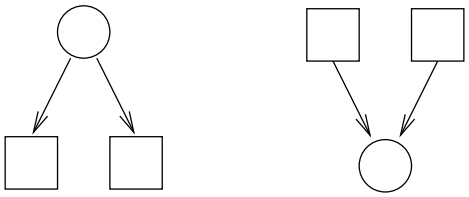


Figure 2.3: Forward and backward conflicts

Exercise. Assume that $(A, \to)$ verifies the following property: for every M, P, Q such that $P \leftarrow M \stackrel{*}{\to} Q$, there exists an element $N \in A$ such that $P \stackrel{*}{\to} N \stackrel{*}{\leftarrow} Q$. Show that $(A, \to)$ is confluent. ■

Exercise. Let $(A, \stackrel{\leq 1}{\to})$ be the abstract rewriting system constructed from $(A, \to)$ by replacing $\to$ by its reflexive closure $\stackrel{\leq 1}{\to}$. Show that $(A, \stackrel{\leq 1}{\to})$ is confluent if and only if $(A, \to)$ is confluent. Deduce from proposition 2 that an abstract rewriting system $(A, \to)$ is confluent when for every M, P, Q such that $P \leftarrow M \to Q$ there exists $N \in A$ such that $P \stackrel{\leq 1}{\to} N \stackrel{\leq 1}{\leftarrow} Q$. ■

Exercise. A **linear** λ-term is a λ-term whose λ-nodes bind one variable exactly. Show that N is linear when M is linear and β-reduces to N. The linear λ-calculus is the λ-calculus restricted to linear λ-terms. Prove that the linear λ-calculus is confluent. ■

Exercise. Same exercise with the affine λ-calculus. An **affine** λ-term is a λ-term whose λ-nodes bind at most one variable. ■

Exercise. Show that a Petri net defines a confluent system $(A, \to)$ when it does not contain any backward or forward conflict. ■

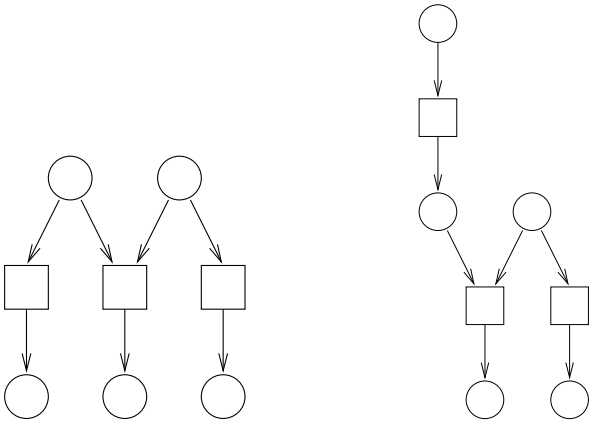


Figure 2.4: Symmetric and asymmetric confusion

Chapter 3

Canonical confluence in concurrent systems

The λ-calculus was proved confluent by two logicians, Church and Rosser, who established that way in 1936 the consistency of their new-born calculus. Later investigations by Curry [Cur 58], Hindley [Hin 69, Hin 78] and especially Lévy [Lév 78] have led to interpret confluence as the existence of pushouts in a category of rewriting paths suitably quotiented. We call *canonical* confluence that kind of universal confluence.

Sections 3.1 and 3.2 introduce the setting of conflict-free graph where confluence is interpreted as a pushout property in the category of paths modulo "permutation". The theory is adapted to the λ-calculus in the next chapter.

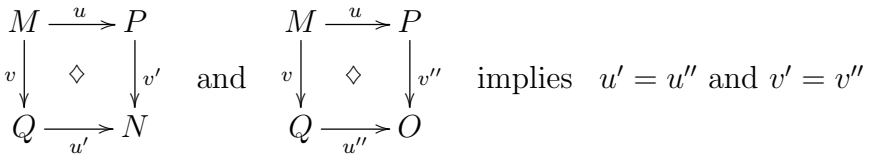
3.1 Concurrent graphs

Definition 1 *A* **pointed** *graph is a graph which contains for every vertice $M \in V$ a specific edge $\emptyset_M : M \longrightarrow M$. This edge $\emptyset_M$ is called the* **point** *of M.*

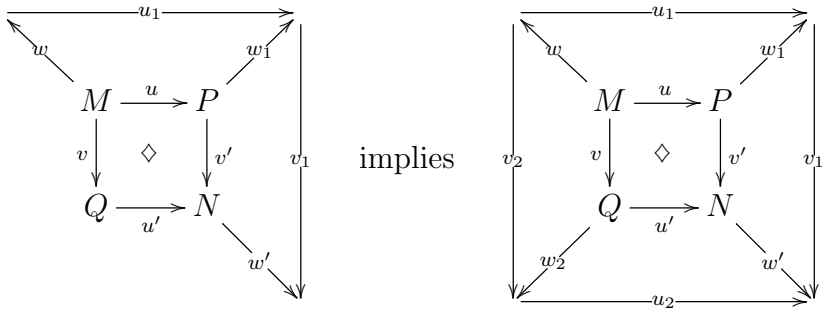
A **concurrent** graph [HL 79, Plo 80] is a pointed graph $\mathcal{G}$ equipped with a symmetric relation $\diamondsuit$ between paths of $\mathcal{G}$, such that for every edges u, u', u'', u_1 and v, v', v'', v_1 and w, w', w_1 and paths f, g:

1. if $f \diamondsuit g$, then f, g are coinitial, cofinal, and $|f| = |g| = 2$,

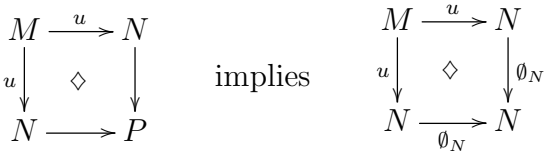
2. if $u; v' \diamondsuit v; u'$ and $u; v'' \diamondsuit v; u''$ then $v' = v''$ and $u' = u''$,



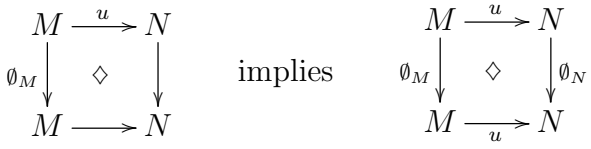
3. if $u; v' \diamondsuit v; u'$ and $w; u_1 \diamondsuit u; w_1$ and $w_1; v_1 \diamondsuit v'; w'$, then there exists edges w_2, v_2, u_2 such that $w; v_2 \diamondsuit v; w_2$ and $w_2; u_2 \diamondsuit u'; w'$ and $u_1; v_1 \diamondsuit v_2; u_2$,



4. if $u : M \longrightarrow N$ and $u; v \diamondsuit u; v$, then $v = \emptyset_N$,



5. if $u : M \longrightarrow N$ and $u; v \diamondsuit \emptyset_M; u'$ then $u = u'$ and $v = \emptyset_N$.



We associate a category $[\mathcal{G}]$ to any concurrent graph $(\mathcal{G}, \diamondsuit)$:

1. its objects are the vertices of $\mathcal{G}$,
2. its morphisms are the paths of $\mathcal{G}$ modulo the binary relation $\diamondsuit$ and the equalities $\emptyset_M = (M)$.

So, two paths $f, g : M \longrightarrow N$ are identified in $[\mathcal{G}]$ precisely when $f \simeq g$, where $\simeq$ is the smallest equivalence relation containing $\diamondsuit$ and both:

1. verifying $f; \emptyset_P; g \simeq f; g$ whenever $M \xrightarrow{f} P \xrightarrow{g} N$,

2. verifying $d; f; e \simeq d; g; e$ whenever $M \xrightarrow{d} P \xrightarrow{f,g} Q \xrightarrow{e} N$ and $f \simeq g$.

We associate to every path $f : M \longrightarrow N$ in $\mathcal{G}$ the corresponding morphism $[f] : M \longrightarrow N$ in $[\mathcal{G}]$. So,

$$[f] = [g] \iff f \simeq g$$

Since u' and v' are uniquely determined by u and v when $u; v' \diamondsuit v; u'$, we introduce the notation u/v for u' and v/u for v'. So, every edge $u : M \longrightarrow N$ defines a partial function $v \mapsto v/u$ from M-edges to N-edges, where by **P-edge** we mean an edge outgoing P. In particular, $\emptyset_M/u$ is $\emptyset_N$ if defined.

Exercise. Show that a concurrent graph is characterised by its pointed graph and the partial functions $v \mapsto v/u$. ■

By composition, every path $f : M \longrightarrow N$ defines a function $u \mapsto u/f$ from M-edges to N-edges. Formally, $(u \mapsto u/f)$ is defined as the identity when f is empty and as $(u \mapsto u/g) \circ (u \mapsto u/v)$ when $f = v; g$.

Exercise. Show that the two partial functions $(u \mapsto u/f)$ and $(u \mapsto u/g)$ are equal when $f \simeq g$. ■

Definition 2 (residual of a path after an edge) *Every edge $u : M \longrightarrow N$ defines a partial function $f \mapsto f/u$ from M-paths to N-paths. The path f/u is defined*

- *as $\mathbf{id}_M/u = \mathbf{id}_N$ when $f = \mathbf{id}_M$,*
- *as $(v; g)/u = (v/u); g/(u/v)$ when $f = v; g$, when v/u and u/v and $g/(u/v)$ are all defined.*

Definition 3 (residual of a path after a path) *Every path $f : M \longrightarrow N$ defines a partial function $h \mapsto h/f$ from M-paths to N-paths. The path h/f is defined*

- *as $h/\mathbf{id}_M = h$ when $f = \mathbf{id}_M$,*
- *as $h/(v; g) = (h/v)/g$ when $f = v; g$ and h/v and $(h/v)/g$ are all defined.*

Exercise*. Assume that two M-paths f and g are such that f/g is defined. Show that g/f is also defined, and that $f; (f/g) \simeq g; (g/f)$. Show also that $f \simeq f'$ and $g \simeq g'$ implies that f'/g' is defined and that $f/g \simeq f'/g'$. ■

3.2 Conflict-free graphs

A **conflict-free** graph is a concurrent graph in which for every two coinitial edges u and v, there exist cofinal edges u' and v' such that $u; v' \diamondsuit v; u'$. Equivalently, it is a concurrent graph in which every edge $u : M \longrightarrow P$ defines a total function $v \mapsto v/u$ from M-edges to N-edges.

Exercise. Assume that f, g are two coinitial paths in a conflict-free graph. Show that f/g is always defined, and that $f/f = \mathbf{id}_{\partial_1 f}$. ■

Exercise. Assume that $(\mathcal{G}, \diamondsuit)$ is a conflict-free graph. Show that $(u; f)/u$ is always defined and equal to f. ■

Proposition 3 *Assume that $(\mathcal{G}, \diamondsuit)$ is a conflict-free graph. Then every morphism in the category $[\mathcal{G}]$ is epi.*

Proof. Recall that a morphism $f : X \longrightarrow Y$ in a category $\mathcal{C}$ is **epi** when for every two morphisms g, h, $g \circ f = h \circ f$ implies $g = h$.

We only prove that every $\simeq$-class $[u]$ of an edge $u : M \longrightarrow N$ is epi. The proposition follows by composition of epis. Suppose that $F \circ [u] = G \circ [u]$ in $[\mathcal{G}]$. By construction of $[\mathcal{G}]$, there are paths f and g such that $[f] = F$ and $[g] = G$. Moreover, the two paths f and g compose with u, so that in fact $F \circ [u] = [u; f]$ and $G \circ [u] = [u; g]$.

We deduce that $f \simeq g$ from $u; f \simeq u; g$ and conclude. From $(u; f)/u = f$ and $u; f \simeq u; g$, we deduce that $(u; g)/u$ is defined and $(u; g)/u \simeq f$. But $(u; g)/u = g$ and we conclude. ■

Let $f : M \longrightarrow P$ and $g : M \longrightarrow N$ be two paths. We write $f \sqsubseteq g$ when there exists a path $f' : P \longrightarrow N$ such that $f; f' \simeq g$.

Proposition 4 *Assume an edge $u : M \longrightarrow P$ and a path $f : M \longrightarrow N$ in a conflict-free graph $(\mathcal{G}, \diamondsuit)$. Then $u \sqsubseteq f$ is equivalent to $u/f = \emptyset_N$.*

Proof. $u \sqsubseteq f$ means that there is a path $g : P \longrightarrow N$ such that $u; g \simeq f$. In particular $u/f = u/(u; g)$. But $u/(u; g) = (u/u)/g = \emptyset_P/g = \emptyset_N$, and we conclude that $u \sqsubseteq f$ implies $u/f = \emptyset_N$.

Conversely, we prove by induction on $|f|$ that $u/f = \emptyset_N$ implies $u \sqsubseteq f$. If $f = (M)$, then $u/f = u$ and we conclude. Otherwise, suppose that $f = v; g$, that $u/f = \emptyset_N$ and that g verifies the property. Let $w = u/v$. That $u/f = \emptyset_N$ means that $w/g = \emptyset_N$, therefore that $w \sqsubseteq g$. So, there exists a path h such that $w; h \simeq g$, hence $u; (v/u); h \simeq v; w; h \simeq v; g \simeq f$. So, $u \sqsubseteq f$. We conclude by induction. ■

Proposition 5 *Assume that u and v are two M-edges in a conflict-free graph $(\mathcal{G}, \diamondsuit)$. Then $[v/u] \circ [u]$ and $[u/v] \circ [v]$ form a pushout diagram in $[\mathcal{G}]$.*

Proof. Suppose that $F : P \longrightarrow N$ and $G : Q \longrightarrow N$ verify $F \circ [u] = G \circ [v]$ in $[\mathcal{G}]$. There exists paths f and g in $\mathcal{G}$ such that $[f] = F$ and $[g] = G$ and $u; f \simeq v; g$. By proposition 4 $v/(u; f) = \emptyset_N$ and $u/(v; g) = \emptyset_N$. In particular, $(v/u)/f = \emptyset_N = (u/v)/g$. By proposition 4 again, $v/u \sqsubseteq f$ and $u/v \sqsubseteq g$. This means that there exists two paths h_1 and h_2 such that $(v/u); h_1 \simeq f$ and $(u/v); h_2 \simeq g$. Let us write $H_1 = [h_1]$ and $H_2 = [h_2]$ and establish that $H_1 = H_2$ using the equalities

$$H_1 \circ [v/u] \circ [u] = F \circ [u] = G \circ [v] = H_2 \circ [u/v] \circ [v]$$

and epiness of $[v/u] \circ [u] = [u/v] \circ [v]$.

To prove that $[v/u] \circ [u]$ and $[u/v] \circ [v]$ form the pushout diagram of $[u]$ and $[v]$, we only need to check the uniqueness of $H = H_1 = H_2$ such that $H \circ [v/u] = F$ and $H \circ [u/v] = G$. But this is a direct consequence of epiness of $[v/u]$ and $[u/v]$. We conclude. ∎

Proposition 6 (Lévy) *Assume that $(\mathcal{G}, \diamondsuit)$ is a conflict-free graph. Then its category $[\mathcal{G}]$ has pushouts.*

Proof. By proposition 5, every diagram (3.1) is a pushout.

$$\begin{array}{ccc} M & \xrightarrow{[u]} & P \\ {\scriptstyle [v]}\downarrow & & \downarrow{\scriptstyle [v/u]} \\ Q & \xrightarrow[[u/v]]{} & N \end{array} \tag{3.1}$$

We use the categorical pasting lemma, see [Mac 71] for information, to tile and close any diagram $P \xleftarrow{F} M \xrightarrow{G} Q$ into a pushout diagram. ∎

Exercises

Exercise. Show that the category $[\mathcal{G}]$ associated to a concurrent graph $(\mathcal{G}, \diamondsuit)$ contains no iso except the identities. ∎

Exercise. A **normal form** in a pointed graph is a vertex P whose only P-edge is $\emptyset_P$. Show that two paths f and g from a vertex M to a normal form P are always $\simeq$-equivalent in a concurrent graph $(\mathcal{G}, \diamondsuit)$. ∎

Exercise. Construct the conflict-free graphs of the linear λ-calculus and of the affine λ-calculus. ∎

Exercise*. Construct the concurrent graph of a Petri net. When is it a conflict-free graph? ■

Exercise. A subgroup H of a group G is **normal** when $aH = Ha$ for every $a \in G$. A group is **simple** when it contains no normal subgroup except $\{e\}$ and itself.

Show that groups form a conflict-free graph where an edge $G \longrightarrow U$ indicates that U is (isomorphic to) a normal subgroup of G and that G/U is simple. Deduce the Jordan-Hölder theorem that two normal towers $G_1 \supset G_2 \cdots \supset \{e\}$ and $H_1 \supset H_2 \cdots \supset \{e\}$ are equivalent when $G_1 = H_1$ and every G_i/G_{i+1} (resp. H_i/H_{i+1}) is simple and non trivial. ■

Chapter 4

Canonical confluence in the λ-calculus

Proposition 6 is remarkable and conveys hope for a similar interpretation of the λ-calculus as a conflict-free graph. This would imply the existence of pushouts in its category of paths modulo (some adequate notion of) permutation equivalence. Unfortunately, the framework of conflict-free graph appears to be quite restrictive at this point. Indeed, a satisfactory notion of permutation in the λ-calculus should validate diagrams of that form:

$$\begin{array}{ccc} \Delta(Ia) & \xrightarrow{\quad I \quad} & \Delta a \\ {\scriptstyle \Delta_1}\downarrow & & \downarrow{\scriptstyle \Delta_2} \\ (Ia)(Ia) & \xrightarrow[I_1]{} a(Ia) & \xrightarrow[I_2]{} aa \end{array} \tag{4.1}$$

which mirrors the duplication by $\Delta = (\lambda x.xx)$ of its argument Ia. [The λ-term $I = (\lambda x.x)$ is the identity]. Clearly, the diagram (4.1) does not relate rewriting paths of length 2 and therefore cannot be considered as a permutation $f \diamondsuit g$ in a conflict-free graph.

This chapter is devoted to a trick to interpret diagrams like (4.1) as diagrams in a conflict-free graph. The trick is to consider a (conflict-free) graph where $I_1; I_2$ appears as an edge instead of a path of length 2. Edges in this new graph are called "multi-edges" and are presented next.

4.1 Lévy graphs

A **Lévy graph** is a graph $\mathcal{L}$ and for every edge $u : M \longrightarrow N$ a relation $[\![u]\!]$ between M-edges and N-edges. Every Lévy graph is supposed to verify four axioms that we introduce after the necessary definitions.

Let $\mathcal{L}$ be a Lévy graph. We extend the notation $[\![-]\!]$ to paths by defining

$$[\![u_1;\cdots;u_k]\!] \stackrel{\Delta}{=} [\![u_1]\!] \circ \cdots \circ [\![u_k]\!]$$

An edge v is a **residual** of u through f when $u[\![f]\!]v$. We define the set $u[\![f]\!]$ as $\{v | u[\![f]\!]v\}$, the set of residuals of u though f.

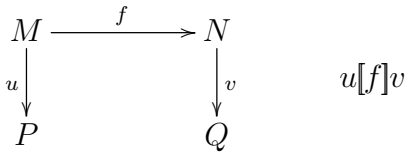


Figure 4.1: The edge v is a residual of u through the path f

A **multi-edge** in $\mathcal{L}$ is a couple (U, M) consisting of a vertex M and a finite set U of M-edges. We often write U_M instead of (U, M). We extend the relation $[\![f]\!]$ to multi-edges by declaring

$$V_M[\![f]\!]W_N$$

when $f : M \longrightarrow N$ and $W = \{w | \exists v \in V, v[\![f]\!]w\}$. Observe that we have the equivalence

$$U_M[\![u_1;\cdots;u_k]\!]V_N \iff U_M[\![u_1]\!] \circ \cdots \circ [\![u_k]\!]V_N$$

We use the notation $(U, M) \xrightarrow{u} (V, P)$ to express that $u : M \longrightarrow P$ is an edge element of U, and $(U, M)[\![u]\!](V, P)$. A **development** of a multi-edge U_M is a path

$$M = M_1 \xrightarrow{u_1} M_2 \xrightarrow{u_2} \cdots \xrightarrow{u_{i-1}} M_i \xrightarrow{u_i} M_{i+1}$$

such that

$$(U, M) \xrightarrow{u_1} (U_2, M_2) \xrightarrow{u_2} \cdots \xrightarrow{u_{i-1}} (U_i, M_i) \xrightarrow{u_i} (\emptyset, M_{i+1})$$

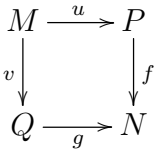
The four properties every Lévy graph $\mathcal{L}$ must verify are:

1. for every two coinitial edges u, v: $u[\![v]\!]$ is finite,
2. for every edge u, $u[\![u]\!] = \emptyset$,
3. there is no infinite sequence

$$(U_1, M_1) \xrightarrow{u_1} (U_2, M_2) \xrightarrow{u_2} \cdots \xrightarrow{u_{i-1}} (U_i, M_i) \xrightarrow{u_i} \cdots$$

of multi-edges (U_i, M_i) and edges $u_i \in U_i$,

4. for every vertex M and two different M-edges $u : M \longrightarrow P$ and $v : M \longrightarrow Q$, there exist developments f and g of $(v[\![u]\!], P)$ and $(u[\![v]\!], Q)$ respectively such that $[\![f]\!] = [\![g]\!]$.



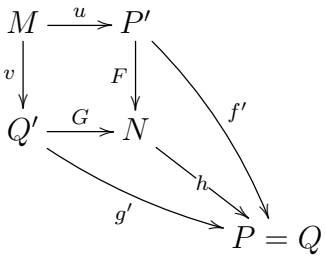
Axioms 1 and 3 together imply that every multi-edge has at least one development. Axioms 3 and 4 together imply that all these developments are equivalent in the following strong sense.

Proposition 7 *Assume that U_M is a multi-edge and that $f : M \longrightarrow P$ and $g : M \longrightarrow Q$ are developments of U_M. Then $P = Q$ and $[\![f]\!] = [\![g]\!]$.*

Proof. By induction on the length of the longest development of U_M. This longest path exists by König's lemma and finiteness of every multi-edge. The proposition is verified when $U_M = \emptyset_M$. Suppose now that $f = u; f'$ and $g = v; g'$. If $u = v : M \longrightarrow N$, then $U_M[\![u]\!]V_N$ for $V = \{v | \exists w \in U, w[\![u]\!]v$, and f' and g' are developments of this multi-edge V_N. In particular, f' and g' are cofinal and since $[\![f']\!] = [\![g']\!]$ we have

$$[\![f]\!] = [\![u; f']\!] = [\![u]\!] \circ [\![f']\!] = [\![u]\!] \circ [\![g']\!] = [\![u; g']\!] = [\![g]\!]$$

Suppose now that $u : M \longrightarrow P'$ and $v : M \longrightarrow Q'$ are different. In that case, there exist two cofinal developments $F : P' \longrightarrow N$ and $G : Q' \longrightarrow N$ of $(v[\![u]\!], P')$ and $(v[\![u]\!], Q')$ respectively such that $[\![u; F]\!] = [\![v; G]\!]$. In particular, if W_N denotes the residual $U_M[\![u]\!][\![F]\!]$ of $U_M[\![u]\!]$ through F, it is also the residual $U_M[\![v]\!][\![G]\!]$ of $U_M[\![v]\!]$ through G. So, letting h be a development of W_N, the two paths $F; h$ and $G; h$ are developments of $U_M[\![u]\!]$ and $U_M[\![v]\!]$ respectively, as are f' and g'. So, $F; h$ and f' are cofinal by induction hypothesis on $U_M[\![u]\!]$, as are similarly $G; h$ and g'. Hence $P = Q$.



By induction hypothesis again, $[\![f']\!] = [\![F; h]\!]$ and $[\![g']\!] = [\![G; h]\!]$, so

$$[\![f]\!] = [\![u]\!] \circ [\![F]\!] \circ [\![h]\!] = [\![u; F]\!] \circ [\![h]\!] = [\![v; G]\!] \circ [\![h]\!] = [\![v]\!] \circ [\![G; h]\!] = [\![v]\!] \circ [\![g']\!] = [\![g]\!]$$

This conludes the proof by induction. ∎

This proposition allows to extend $[\![-]\!]$ to multi-edges by writing $V_M[\![U_M]\!]W_P$ precisely when $f : M \longrightarrow P$ is a development of U_M and $W = V[\![f]\!]$. We say in that case that W_P is the residual of V_M through U_M, and write $W_P = U_M[\![V_M]\!]$. This extension paves the way to the definition of a conflict-free graph $\widehat{\mathcal{L}} = (\mathcal{G}, \diamondsuit)$ associated to $\mathcal{L}$:

1. its vertices are the vertices of $\mathcal{L}$,
2. its edges are the multi-edges of $\mathcal{L}$,
3. its points $\emptyset_M : M \longrightarrow M$ are the multi-edges $\emptyset_M$,
4. the relation $\diamondsuit$ relates two paths f and g precisely when they are of the form $U_M; V_M[\![U_M]\!]$ and $V_M; U_M[\![V_M]\!]$

$$\begin{array}{ccc} M & \xrightarrow{U} & P \\ {\scriptstyle V}\downarrow & \diamondsuit & \downarrow{\scriptstyle V[\![U]\!]} \\ Q & \xrightarrow[U[\![V]\!]]{} & N \end{array}$$

Exercise. Show that the couple $(\mathcal{G}, \diamondsuit)$ constructed above is a conflict-free graph. ∎

Note in particular that the edge $W_P = U_M/V_M$ in $(\mathcal{G}, \diamondsuit)$ is precisely the multi-edge $U_M[\![V_M]\!]$.

4.2 Lévy permutation equivalence

In Section 3.2, we presented the category $[\mathcal{G}]$ associated to any conflict-free graph $(\mathcal{G}, \diamondsuit)$. We have just seen in the previous section how to construct a conflict-free graph $\widehat{\mathcal{L}}$ from any Lévy graph $\mathcal{L}$. By composition, there is a category $[\widehat{\mathcal{L}}]$ which, by proposition 6, has pushouts. But can $[\widehat{\mathcal{L}}]$ be directly defined from $\mathcal{L}$? The answer is positive, and we present the solution here.

Assume $\mathcal{L}$ is a Lévy graph and $\heartsuit$ relates two paths $u; f$ and $v; g$ precisely when $u : M \longrightarrow P$ and $v : M \longrightarrow Q$ are two different coinitial edges and f and g are respective developments of $(v[\![u]\!], P)$ and $(u[\![v]\!], Q)$.

$$\begin{array}{ccc} M & \xrightarrow{u} & P \\ {\scriptstyle v}\downarrow & \heartsuit & \downarrow{\scriptstyle f} \\ Q & \xrightarrow[g]{} & N \end{array}$$

Let $[\mathcal{L}]$ be the category

1. whose objects are the vertices of $\mathcal{L}$,
2. whose morphisms are the paths modulo $\heartsuit$.

So, two paths $f, g : M \longrightarrow N$ are identified in $[\mathcal{L}]$ precisely when $f \equiv g$, where $\equiv$ is the smallest equivalence relation containing $\heartsuit$ and such that $d; f; e \equiv d; g; e$ whenever $M \xrightarrow{d} P \xrightarrow{f,g} Q \xrightarrow{e} N$ and $f \equiv g$.

The relation $\equiv$ is called **Lévy permutation equivalence** in the litterature. We associate to every path $f : M \longrightarrow N$ in $\mathcal{L}$ the corresponding morphism $[f] : M \longrightarrow N$ in $[\mathcal{L}]$. So,

$$[f] = [g] \iff f \equiv g$$

The construction of $[\mathcal{L}]$ is justified by the following proposition.

Proposition 8 *The category $[\mathcal{L}]$ is isomorphic to $[\widehat{\mathcal{L}}]$.*

Proof. Let $F : [\mathcal{L}] \longrightarrow [\widehat{\mathcal{L}}]$ transport any morphism $[u]$ for an edge $u : M \longrightarrow N$ in $\mathcal{L}$ to the morphism $[\{u\}_M]$ in $[\widehat{\mathcal{L}}]$. This functor is well-defined because $u_1; \cdots; u_m \heartsuit v_1; \cdots; v_n$ in $\mathcal{L}$ implies in $\widehat{\mathcal{L}}$ that:

$$\{u_1\}; \cdots; \{u_k\} \simeq \{u_1, v_1\} \simeq \{v_1\}; \cdots; \{v_n\}$$

Now, let $G : [\widehat{\mathcal{L}}] \longrightarrow [\mathcal{L}]$ transport any morphism $[U_M]$ for an edge $U_M : M \longrightarrow N$ in $\widehat{\mathcal{L}}$ to $[f]$ in $[\mathcal{L}]$, where f is a development of the multi-edge U_M. This morphism is well-defined by proposition 7. We could leave the proof that $G \circ F = \mathbf{id}_{[\mathcal{L}]}$ and $F \circ G = \mathbf{id}_{[\widehat{\mathcal{L}}]}$ to the reader, but we carry it out in full. Observe that we only have to check the identities on morphisms of the form $[u]$ where u is an edge of $\mathcal{L}$ or $\widehat{\mathcal{L}}$. For an edge $u : M \longrightarrow N$ in $\mathcal{L}$, $F[u] = \{u\}_M$ and therefore $G \circ F[u]$ is the $\equiv$-class which contains the developments of the multi-edge $\{u\}_M$. But u is a development of $\{u\}_M$, hence $G \circ F[u] = [u]$. For an edge $U_M : M \longrightarrow N$ in $\widehat{\mathcal{L}}$, $G[u]$ is the $\equiv$-class of the developments of U_M, hence $F \circ G[u]$ is the $\simeq$-class containing the paths $\{u_1\}_{M_1}; \cdots; \{u_k\}_{M_k}$ such that $M = M_1 \xrightarrow{u_1} M_2; \cdots; M_k \xrightarrow{u_k} N$ develops U_M. We only have to show that in that case

$$U_M \simeq \{u_1\}_{M_1}; \cdots; \{u_k\}_{M_k}$$

But this is the consequence of the equivalence $W_P \simeq \{w\}_P; W_P[w]$ when $w : P \longrightarrow Q$ is element of W, and a straightforward induction argument. ∎

The important consequence of proposition 8 is that the category $[\mathcal{L}]$ has all pushouts. In particular, two paths $M \xrightarrow{f,g} N$ to a normal form N are equal modulo $\equiv$. This uniqueness property plays a central role in the search for normalising computational strategies in Chapter V.

Exercise. Prove the assertion that in a Lévy graph every two paths $M \longrightarrow N$ to a normal form N are equal modulo $\equiv$. ∎

4.3 The λ-calculus

The whole chapter has been devoted to graph-theoretic descriptions of Rewriting systems, but we have not taken the time yet to translate the λ-calculus into a Lévy graph $\mathcal{L}_\lambda$. We do this now.

The vertices of the Lévy graph $\mathcal{L}_\lambda$ are the λ-terms. Its edges $u : M \longrightarrow N$ called **β-redexes** are the triples $u = (M, o, N)$ where o is the occurence of a β-pattern $(\lambda x.P)Q$ in $M = C[(\lambda x.P)Q]$ such that $N = C[P[x := Q]]$. Here C is a **context**, that is λ-tree on the set $\mathbf{var} + \{-\}$ of variables. Every occurrence of $-$ in C is called a **hole**. So, in our case the only hole $-$ of C is at occurrence o, and a λ-tree can be seen as context with no holes. Every context C defines a function $M \mapsto C[M]$ from λ-terms to λ-terms, defined as follows:

1. $-[M] \stackrel{\Delta}{=} M$ and for $x \in \mathbf{var}$, $x[M] \stackrel{\Delta}{=} x$,
2. $(\lambda x.C)[M] \stackrel{\Delta}{=} \lambda x.(C[M])$,
3. $CC'[M] \stackrel{\Delta}{=} C[M]C'[M]$.

Exercise. Assume that C is a λ-tree and show that $C[M]$ does not depend on M and is the λ-term associated to C (= its class modulo α-equivalence). ■

We return to the construction of $\mathcal{L}_\lambda$ and associate to every edge $u : M \longrightarrow N$ a total function $\mathcal{F}_u : o_N \mapsto \mathcal{F}_u(o_N)$ from N-occurrences to M-occurrences:

1. let $(\lambda x.M)P \stackrel{u}{\longrightarrow} M[x := P]$. The function $\mathcal{F}_u$ maps an occurrence o in $M[x := P]$ to $1;1;o$ when o is an occurrence of M not labelled by x and to $2;o'$ when there is an occurrence o_x of M labelled by x such that $o = o_x; o'$.
2. let $u = (M, o_u, N) : C[(\lambda x.m)p] \longrightarrow C[m[x := p]]$ where C's hole has occurrence o, and $v = ((\lambda x.m)p, \epsilon, m[x := p])$. Then $\mathcal{F}_u(o) \stackrel{\Delta}{=} o_u; \mathcal{F}_v(o')$ when $o = o_u; o'$ for some occurrence o', and $\mathcal{F}_u(o) = o$ otherwise.

It will be useful in the sequel to observe that $\mathcal{F} : \mathcal{L}^{op} \longrightarrow \mathbf{Set}$ defines a **presheaf** from $\mathcal{L}$ considered as a free category. For M a λ-term, $\mathcal{F}_M$ is simply the set of M's occurrences, and $\mathcal{F}_f : \mathcal{F}_N \longrightarrow \mathcal{F}_M$ is defined as $\mathcal{F}_{u_1} \circ \cdots \circ \mathcal{F}_{u_k}$ for a path $f = u_1; \cdots; u_k : M \longrightarrow N$.

Observe also that two occurrences o in N and $\mathcal{F}_u(o)$ in M have the same label. So, for $v = (M, o, P)$ an M-edge and $w = (N, o', Q)$ an N-edge, we simply write $v[\![u]\!]w$ when $\mathcal{F}_u(o') = o$. The relation $[\![u]\!]$ then defines a partial function from N-edges to M-edges.

Proposition 9 *Assume that M, P, Q are λ-terms such that $x \in \mathbf{var}$ is not free in Q and $x \neq y$. Then $M[x := P][y := Q] = M[y := Q][x := P[y := Q]]$.*

Proof. By induction on M. ∎

Proposition 10 *$\mathcal{L}_\lambda$ is a Lévy graph.*

Proof. $\mathcal{L}_\lambda$ clearly verifies the two first axioms of Lévy graphs: the set $u[\![v]\!]$ is finite for u and v coinitial, and $u[\![u]\!]$ is always empty. That axiom 3 holds in $\mathcal{L}_\lambda$ is an important and famous normalisation result on the λ-calculus, called the **finite development** lemma. The proof which is too complicated to carry out here is presented in Chapter IV.

For the moment, we prove that axiom 4 holds in $\mathcal{L}_\lambda$. Let $u = (M, o, P)$ and $v = (M, o', Q)$ be two distinct M-edges. Suppose that o and o' are disjoint in the sense that none precedes the other. Then, $u[\![v]\!]$ and $v[\![u]\!]$ are singletons $\{u'\}$ and $\{v'\}$ respectively, where $u' = (P, o', N_1)$ and $v' = (Q, o, N_2)$ for some λ-terms N_1 and N_2. It is not difficult to see that $N_1 = N_2$ and that $[\![u]\!] \circ [\![v']\!] = [\![v]\!] \circ [\![u']\!]$.

Now, we treat the case when u's occurrence o precedes v's occurrence o'. Let us consider first the case when o is the root occurrence ϵ. Then $M = (\lambda x.M_1)P_1$ and $u = (M, \epsilon, M_1[x := P_1])$ while v occurs in M_1 or in P_1. In both cases, $M = C[(\lambda y.M_2)P_2]$ and $v : M \longrightarrow C[M_2[y := P_2]]$ for a context C whose unique hole is at occurrence o'. Here we ask that $x \neq y$ and that C does not contain any bound variable x, so that in particular $C[M][x := P] = C[x := P][M[x := P]]$ for any λ-term M and P. We also ask that y does not appear free in P_1.

If v occurs in P_1, then $u[\![v]\!]$ is a singleton $\{u'\}$ and $v[\![u]\!] = \{v_1, ..., v_k\}$ contains as many β-redexes as there as occurrences of x in M_1. We have the following diagram for f a development of $v[\![u]\!]$ and $C = (\lambda x.M)C'$.

$$\begin{array}{ccc} (\lambda x.M_1)C'[(\lambda y.M_2)P_2] & \xrightarrow{\quad u \quad} & M_1[x := C'[(\lambda y.M_2)P_2]] \\ \Big\downarrow{\scriptstyle v} & & \Big\downarrow{\scriptstyle f} \\ (\lambda x.M_1)C'[M_2[y := P_2]] & \xrightarrow{\quad u' \quad} & M_1[x := C'[M_2[y := P_2]]] \end{array}$$

We leave to the reader the proof that $\mathcal{F}_u \circ \mathcal{F}_f = \mathcal{F}_v \circ \mathcal{F}_{u'}$ which implies that $[\![u; f]\!] = [\![v; u']\!]$.

If v occurs in M_1, then $u[\![v]\!]$ and $v[\![u]\!]$ are singletons $\{u'\}$ and $\{v'\}$ which

form the following diagram for $C = (\lambda x.C')P_1$ and $C'' = C'[x := P_1]$:

$$\begin{array}{ccc}
(\lambda x.C'[(\lambda y.M_2)P_2])P_1 & \xrightarrow{u} & C''[(\lambda y.M_2[x := P_1])P_2[x := P_1]] \\
\downarrow{\scriptstyle v} & & \downarrow{\scriptstyle v'} \\
(\lambda x.C'[M_2[y := P_2]])P_1 & & C''[M_2[x := P_1][y := P_2[x := P_1]] \\
\downarrow{\scriptstyle u'} & & \\
C''[M_2[y := P_2][x := P_1]] & &
\end{array}$$

But proposition 9 insures that $M_2[y := P_2][x := P_1] = M_2[x := P_1][y := P_2[x := P_1]$ and therefore that the diagram closes. We leave to the reader the proof that $\mathcal{F}_u \circ \mathcal{F}'_v = \mathcal{F}_v \circ \mathcal{F}'_u$ which implies that $[\![u; v']\!] = [\![v; u']\!]$.

Now, when the β-redex $u = (M, o, N)$ is not at the root ϵ, then $M = C[\underline{M}]$ for some context C whose only hole is at occurrence o. Letting v's occurrence o' be $o; o''$, the $\underline{M}$-edges $\underline{u}$ at occurrence ϵ and $\underline{v}$ at occurrence o'' verify the property 4 of Lévy graphs. But the property is preserved by substituting the λ-term $\underline{M}$ in the context C, so property 4 is verified on u and v. We conclude. ■

The argument above implies that $\mathcal{F}$ is a functor from $[\mathcal{L}_\lambda]^{op}$ to **Set**. Also, note that $\mathcal{F}$ verifies the following nice property.

Exercise*. Show that $\mathcal{F} : [\mathcal{L}_\lambda]^{op} \longrightarrow$ **Set** preserves pullbacks. ■

This preservation property means that in any pushout diagram of $[\mathcal{L}]$:

$$\begin{array}{ccc}
M & \xrightarrow{f} & P \\
{\scriptstyle g}\downarrow & & \downarrow{\scriptstyle g'} \\
Q & \xrightarrow[f']{} & N
\end{array}$$

the sets $\mathcal{F}_N$ and $\{(p, q) | p \in \mathcal{F}_P, q \in \mathcal{F}_Q, \mathcal{F}_f(p) = \mathcal{F}_g(q)\}$ are one-to-one.

Exercises

Exercise. Show that every two paths $M \xrightarrow{f,g} N$ to a normal form in $\mathcal{L}_\lambda$ are equal modulo $\equiv$. Compare the various paths implementing $I(Ia) \longrightarrow a$ and $Ka(\Delta\Delta) \longrightarrow a$. ■

Bibliography

[ACCL 90] M. Abadi, L. Cardelli, P.-L. Curien, J.-J. Lévy, "*Explicit substitutions*". Conférence Principle Of Programming Languages, 1990.

[Bar 80] H.-P. Barendregt, "*The Lambda Calculus: Its Syntax and Semantics*". North Holland, 1985.

[Ber 79] G. Berry, "*Modèles complètement adéquats et stables des lambda-calculs typés*" Thèse de Doctorat d'Etat, Université Paris VII, 1979.

[Bou 85] G. Boudol, "*Computational semantics of term rewriting systems*". Algebraic methods in Semantics, Maurice Nivat and John C. Reynolds (eds), Cambridge University Press, 1985.

[Chu 36] A. Church, J.-B. Rosser, "*Some properties of conversion*". Transactions of the American Mathematical Society Volume 39, pages 472—482, 1936.

[Chu 41] A. Church, "*The calculi of lambda conversion*", Princeton University, 1941.

[CK 95] D. Clark, R. Kennaway, "*Some properties of non-orthogonal Graph Rewriting* To appear as part of the Elsevier Electronic Notes on Computer Science, 1995.

[CHR 92] P.-L. Curien, T. Hardin, A. Ríos, "*Strong Normalization of Substitutions*". Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 629, 1992.

[Cur 58] H.-B. Curry, R. Feys, "*Combinatory Logic*". North Holland Volume 1, 1958.

[DeB 72] N. De Bruijn, "*Lambda-calculus Notation with Nameless Dummies, a Tool for Automatic Formula Manipulation*". Indag. Mat. Volume 34, 1972.

[FK 72] P.J. Freyd, G.M. Kelly, "*Categories of continuous functors, I*". Journal of Pure and Applied Algebra 2, pp 169—191, 1972.

[Gir 95] J. Y. Girard, "*Linear Logic: its syntax and semantics*" in "Advances in Linear Logic", edited by Jean-Yves Girard, Yves Lafont aet Laurent Regnier, Cambridge University Press, 1995.

[GLM 92] G. Gonthier, J.-J. Lévy, P.-A. Mellies, "*An abstract standardisation theorem*". Seventh Annual IEEE Symposium on Logic In Computer Science, August 1992.

[HaLé 92] T. Hardin, J.-J. Lévy, "*Confluence Properties of Weak and Strong Calculi of Explicit Substitutions*". Rapport de recherche INRIA Rocquencourt 1617, février 1992.

[Hin 69] R. Hindley, "*An abstract form of the Church Rosser theorem*". Journal of Symbolic logic Volume 34, Number 4, December 1969.

[Hin 78] R. Hindley, "*Reductions of residuals are finite*". Transactions of the American Mathematical Society Volume 240, June 1978.

[Hin 86] J.R. Hindley, J.P. Seldin, "*Introduction to Combinators and λ-calculus*". London Mathematical Society, Students Texts 1, Cambridge University Press, 1986.

[Hue 80] G. Huet, "*Confluent Reductions : Abstract Properties and Applications to Term Rewriting Systems*". Journal of the Association for Computing Machinery vol. 27, No 4 (1980), 797–821.

[HL 79] G. Huet, J.-J. Lévy, "*Call by Need Computations in Non-Ambiguous Linear Term Rewriting Systems*". Rapport de recherche INRIA 359, 1979.

[HL 91] G. Huet, J.-J. Lévy, "*Computations in orthogonal rewriting systems*". In J.-L. Lassez and G. D. Plotkin, editors, *Computational Logic; Essays in Honor of Alan Robinson*, pages 394–443. MIT Press, 1991.

[Hyl 73] J.M.E. Hyland, "*A simple proof of the Church-Rosser theorem*", Type Script, Oxford University.

[Klo 80] J.W. Klop, "*Combinatory Reduction Systems*". Thèse de l'Université d'Utrecht, Pays-Bas (1980).

[Klo 92] J.W. Klop, "*Term Rewriting Systems*" Handbook of Logic in Computer Science, Volume 2, in S. Abramsky, Dov M. Gabbay, T.S.E. Maibaum, editors, Oxford Science Publications, 1992.

[KM 89] J.W. Klop, A. Middeldorp, "*Sequentiality in Orthogonal Term Rewriting Systems*". Report Cs-R8932, Centre for Mathematics and Computer Science, Amsterdam, Pays-Bas.

[KOR 93] Jan Willem Klop, Vincent van Oostrom, Femke van Raamsdonk, "*Combinatory Reduction Systems: introduction and survey*". in "A Collection of Contributions in Honour of Corrado Böhm", M. Dezani-Ciancaglini, S. Ronchi Della Rocca, and M. Venturini Zilli editors, Theoretical Computer Science, volume 121, pages 279–308, 1993.

[KB 70] D.E. Knuth, P.B. Bendix, "*Simple word problems in universal algebras*", In J. Leech, editor, Computational Problems in Abstract Algebra, page 263–297, Pergamon Press, 1970.

[Kru 60] J. B. Kruskal, "*Well quasi orderings, the tree theorem, and Vázsonyi's conjecture*", Trans. America Math. Soc. 95, 210—225, 1960.

[Laf 95] Yves Lafont, "*From proof-nets to interaction nets*", in "Advances in Linear Logic", edited by Jean-Yves Girard, Yves Lafont aet Laurent Regnier, Cambridge University Press, 1995.

[Lév 78] Jean-Jacques Lévy, "*Réductions correctes et optimales dans le λ-calcul*". Thèse de Doctorat d'Etat, Université Paris VII, 1978.

[Lév 80a] Jean-Jacques Lévy, "*Optimal reductions in the lambda-calculus*". In J.P. Seldin and J.R. Hindley editors, "To H.B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism", Academic Press. 1980.

[Lév 80b] Jean-Jacques Lévy, "*Computation as Deduction*". Course note 7 at CMU.

[Mac 71] Sauders Mac Lane, "*Categories for the working mathematician*", Springer Verlag, 1971.

[MM 92] Sauders Mac Lane, Ieke Moerdijk, "*Sheaves in Geometry and Logic — a first introduction to topos theory*", Springer-Verlag, 1992.

[Mel 95] Paul-André Melliès, "*Typed Lambda-Calculi with Explicit Substitutions may not terminate*". Proceedings of the 2nd Conference on Typed Lambda Calculi and Applications, Edinburgh, LNCS 902, pp. 328-334, Springer, 1995.

[Mel 95b] Paul-André Melliès, "Braids as an orthogonal system". Manuscript, 1995.

[Mel 97] Paul-André Melliès, "A factorisation theorem in rewriting theory". Proceedings of the 7th Conference on Category Theory and Computer Science, Santa Margherita Ligure, LNCS 1290, pp. 49-68, Springer, 1997.

[Mel 98] Paul-André Melliès, "A stability theorem in rewriting theory". Proceedings of the 14th Symposium on Logic in Computer Science, Indianapolis, 1998.

[Mil 91] R. Milner, "*The polyadic pi-calculus:a tutorial*" Research Report ECS-LFCS-01-180, Laboratory for Fondations of Computer Science, Edinburgh University, 1991.

[Mil 92] R. Milner, "*Action structures*" Research Report LFCS-92-249, Laboratory for Fondations of Computer Science, Edinburgh University, 1992.

[Mil 94] R. Milner, "*Action calculi, or syntactic action structures*" Proceedings of MFCS'93, Lecture Notes in Computer Science, Springer, 1993.

[Ned 73] R. P. Nederpelt, "*Strong Normalisation in a typed lambda calculus with lambda structured types*" in R. P. Nederpelt, J. H. Geuvers, R. C. de Vrijer Editors, "Selected Papers on Automath", Studies in Logic, North Holland, volume 133, 1994.

[New 42] M.H.A. Newman, "On theories with a combinatorial definition of "equivalence"" Annals of Mathematics, 43, Number 2, pages 223–243.

[NW 95] M. Nielsen, G. Winkel, "Models for concurrency", in *Handbook of Logic in Computer Science*, Abramsky, Gabbay, Maibaum editors, Oxford Science, 1995.

[O'D 77] M. O'Donnell, "*Computing in Systems Described by Equations*". LNCS No 58, Eds G. Goos and J. Hartmanis, Springer, 1977.

[Oos 94] V. van Oostrom, "*Confluence for Abstract and Higher-Order Rewriting*", Thèse de l'Université Libre d'Amsterdam, Pays-Bas (1994).

[Plo 80] G. Plotkin, "Church-Rosser categories". Manuscript, 1980.

[Ram 96] F. van Raamsdonk, "*Confluence and Normalisation for Higher-Order Rewriting*", Thèse de l'Université Libre d'Amsterdam, Pays-Bas (1996).

[Rev 85] G. Revesz, "Axioms for the theory of lambda-conversion", *S.I.A.M. Journal of Computation*, 14(2):373-382, mai 1985.

[Sch 65] D. E. Schroer, "*The Church-Rosser theorem*", Phd thesis, Cornell University, 1965.

[Sta 89] E. W. Stark, "Concurrent transition systems", *J. Theoretical Computer Science*, volume 64, number 3, may 1989.

[Wad 71] C.P. Wadsworth, "*Semantics and Pragmatics of the Lambda Calculus*". Oxford Universtity, 1971.

[Zan 93] H. Zantema, "*Termination of term rewriting by interpretation*". Proceedings of CTRS'93, Lecture Notes in Computer Science 656, 1993.