Interaction Systems I
The theory of optimal reductions*

Andrea Asperti Cosimo Laneve
Universitá di Bologna INRIA Sophia Antipolis

September 1992, revised February 1994

Abstract

We introduce a new class of higher order rewriting systems, called Interaction Systems (IS's). IS's come from Lafont's (Intuitionistic) Interaction Nets [Lafont 1990] by dropping the linearity constraint. In particular, we borrow from Interaction Nets the syntactical bipartitions of operators into constructors and destructors and the principle of *binary interaction*. As a consequence, IS's are a subclass of Klop's Combinatory Reduction Systems [Klop 1980] where the Curry-Howard analogy still "makes sense". Destructors and constructors respectively corresponds to left and right logical introduction rules, interaction is cut and reduction is cut-elimination.

Interaction Systems have been primarily motivated by the necessity of extending the practice of optimal evaluators for λ-calculus [Lamping 1990, Gonthier *et al.* 1992a] to other computational constructs as conditionals and recursion. In this paper we focus on the *theoretical* aspects of optimal reductions. In particular, we generalize the *family relation* in [Lévy 1978, Lévy 1980], thus defining the *amount of sharing* an optimal evaluator is required to perform. We comfort our notion of family by approaching to it in two different ways (generalizing labeling and extraction in [Lévy 1980]) and proving their coincidence.

The reader is referred to [Asperti and Laneve 1993c] for the paradigmatic description of optimal evaluators of IS's.

1 Introduction

In 1978, Lévy introduced the notion of "redex family" in the λ-calculus with the aim of formalizing the phenomenon of duplication of redexes, thus providing an abstract definition of what could be considered as an optimal implementation of the λ-calculus. Intuitively,

*An extended abstract of this paper is appeared in the proceedings of *Theoretical Aspects of Computer Science 1994*. This work is partially supported by the ESPRIT Basic Research Project 6454 - CONFER.

since redexes in a family are "copies" of a same redex (they have been generated in the same way), it should be possible to *share* their reduction, therefore avoiding any duplication of work.

For a long time, no implementation was able to achieve the theoretical performance fixed by Lévy (see [Field 1990] for a survey), and it is only in recent years that this problem has been finally solved [Lamping 1990, Kathail 1990]. In particular, the graph reduction technique proposed by Lamping, and remarkably simplified in [Gonthier *et al.* 1992a], has put in evidence a restricted set of operators (fan, croissant and bracket) which provide an "optimal" sharing of common subexpressions. The amount of work performed by the sharing operators has been pointed out in [Gonthier *et al.* 1992b], where Lamping's graphs are related to Linear Logic [Girard 1986]. More precisely, in Linear Logic, there is a neat distinction between the *logical* part of the calculus (dealing with the logical connectives) and the *structural* part (dealing with the management of hypotheses). These two different aspects of logical calculi have their obvious correspondent (via Girard's encoding of Intuitionistic Logic into Linear Logic and the Curry-Howard analogy) inside the λ-calculus. From one side we have the *linear* part (the linear λ-calculus), defining the syntactical operators (the arity, the "binding power", etc.) and their interaction; from the other side we have the *structural* part, taking care of the management (sharing) of resources (i.e. of subexpressions). This refinement can be roughly summarized in the equation

$$\boxed{\lambda\text{-calculus} = \text{linear } \lambda\text{-calculus} + \text{sharing}}$$

The main achievement of [Gonthier *et al.* 1992b] was to show that sharing operators provide a "local" and optimal implementation of the structural part of the calculus. This means that Lamping's evaluation style could be smoothly generalized to a very large class of systems, by just replacing the "logical" part of the λ-calculus (i.e., the linear λ-calculus), with an arbitrary linear calculus. The only proviso to respect was to choose a linear calculus *with a strong logical foundation*, since otherwise we could immediately loose the logical analogy underlying all the previous discussion (in particular, the interface between the linear and the structural part seems to be critical). At this point, we had a natural (and, up to our knowledge, unique) candidate for the linear calculus: Lafont's (Intuitionistic) Interaction Nets [Lafont 1990]. The systems corresponding to

$$\boxed{\text{Intuitionistic Interaction Nets} + \text{sharing}}$$

are ours *Interaction Systems*.

In particular, in IS's, we keep the idea of binary interaction and the syntactical bipartition of operators into *constructors* and *destructors*. We only remove the linearity constraint, and add some further conditions imposed by the intuitionistic (functional) framework. In this respect, IS's turn out to be a subclass of Klop's Combinatory Reduction Systems [Klop 1980]. Roughly, they are the subsystems of Combinatory Reduction Systems (CRS's) where the Curry-Howard analogy still "makes sense". This means that we can associate to every IS a suitable "logical" (intuitionistic) system: constructors and destructors respectively correspond to right and left introduction rules, interaction is cut, and computation is cut-elimination.

The main property of an IS as a rewriting system, is its *sequential nature*; in particular, the leftmost outermost reduction always reduces needed redexes. Note that this is one main brick in the theory of optimal reductions [Lévy 1978]: a reduction strategy is optimal if and only if it reduces maximal families of *needed* redexes. For this reason, it is not fair to compare IS's with the full expressive power of CRS's. They can be only reasonably compared with other known sequential subclasses of the latter ones, such as Left Normal CRS's [Klop 1980]. With respect to Left Normal CRS's, IS's have a strong logical status, a deep relation with Interaction Nets, interesting applications to the theory of data types (due to the bipartition of operators into constructors and destructors), and nice perspectives in the typing direction. Moreover, we conjecture that every Left Normal CRS has an optimal encoding as an Interaction System.

The other important brick of the theory of optimality is the formalization of the notion of family of redexes. Take graphical representations of expressions where duplications of equal subexpressions is avoided by means of sharing mechanisms. Then the problem is to formalize the "optimal" amount of sharing, i.e. to define what are the portions of the expression that are copies. Since the notion of optimal sharing is *pretty intuitive*, it is hard to prove the correctness of any formalization. In the case of λ-calculus, Lévy approached the problem by providing several formal definitions, starting from different perspectives, and proving that all the definitions lead to the same notion [Lévy 1978, Lévy 1980]. The generalization of Lévy's work revealed unexpected problems, especially with the notion of copying that underlies the zig-zag and the extraction process [Lévy 1980]. In particular, in the case of the λ-calculus, the correctness of these two approaches seems to be related to two peculiar properties of the β-reduction, which in general do not hold for IS's. Namely:

1. every metavariable is linearly used in the rhs of the rule; in particular, a same metavariable X cannot be nested within itself, as in the metaexpression $X[^X/_x]$.

2. no new forms are introduced in the rhs of the rule.

Indeed, in the paper, we shall provide several counterexamples both to the zig-zag and the extraction in [Lévy 1980] (see Sections 6.2 and 9).

It happens that only the definition of family based on labels generalizes easily to IS's, due to the assumption of binary interaction. On the other hand the generalization of the extraction process is difficult because of the subtle interplay between the substitution process and the creation of forms. For instance, it is possible to create new forms making redexes and, *at the same time*, copying them. The main result of this paper is the proof of the coincidence, in the general case of IS's, of the approaches based on labels and on the extraction process.

It is quite interesting to notice that the labeling is problematic in (left normal) CRS's, since labels have a much richer (tree-like) syntactical structure (redexes consist no more of two forms but of a *tree* of forms). The extraction process, on the other hand, should hold for (left normal) CRS's as well (see the Conclusion). This justifies its interest, despite its complexity.

The structure of the paper is the following. In Section 2 we discuss detailedly the relationships between Interaction Systems, Interaction Nets and Combinatory Reduction Systems:

we shall also provide the main motivations, both theoretical and operational, behind the introduction of this new class of higher order systems. Sections 3 and 4 are devoted to the formal definition of IS's and other standard notions as residuals, standardization, etc. In Section 5 we introduce the family relation by means of the approach based on labels, that is easily generalizable to IS's. This notion of family is comforted in Section 6 by means of another approach, generalizing the extraction relation in [Lévy 1980]. The equivalence of the two approaches is proved in Section 8. In the final section, we shall briefly address the problem of the inadequacy, in the case of IS's, of other standard approaches to families. Appendix A is devoted to the investigation of the *structure* of labels (for IS's), and provides the basis for our original proof of the equivalence between extraction and labeling. Appendices B and C provide the proofs of two technical lemmas needed in Section 8.

In the paper we assume that the reader is acquainted with the Curry-Howard analogy, Gentzen's calculus of sequents and Girard's Proof Nets (see [Girard *et al.* 1989] for a good introduction).

2 The intuitionistic nature of Interaction Systems

In this section we discuss the main motivations behind the definition of Interaction Systems, relating them to Lafont's Interaction Nets (IN's) and Klop's Combinatory Reduction Systems. We shall particularly stress the logical (intuitionistic) nature of IS's, and the possibility to extend the Curry-Howard analogy to this class of systems. To this aim we shall generalize Intuitionistic Logic by considering generic logical connectives (these logics will be named *intuitionistic systems*). While studying logical foundations of IS's, we shall focus on the problem of optimal evaluators of IS's, that provided the original motivation for this work.

The aim of this section is to be introductory. So, as much as possible, we shall avoid statements (and proofs). The reader is referred to [Asperti and Laneve 1994] for a formal discussion.

2.1 Statics

An *intuitionistic system*, in a sequent calculus presentation (*à la* Gentzen) (see [Girard *et al.* 1989]), consists of expressions, named *sequents*, whose shape is $A_1, \cdots A_n \vdash B$ where A_i and B are formulas. Inference rules are partitioned into three groups (in order to emphasize the relationships with IS's, we write rules by assigning terms to proofs):

(**Structural Rules**)

$$(Exchange)\quad \frac{\Gamma,\, x : A,\, y : B,\, \Delta\ \vdash t : C}{\Gamma,\, y : B,\, x : A,\, \Delta\ \vdash t : C}$$

$$(Contraction)\quad \frac{\Gamma,\, x : A,\, y : A,\, \Delta\ \vdash t : C}{\Gamma,\, z : A,\, \Delta\ \vdash t[{}^{z}/_{x}, {}^{z}/_{y}]\ : C} \qquad (Weakening)\quad \frac{\Gamma\ \vdash t : C}{\Gamma,\, z : A\ \vdash t : C}$$

(**Identity Group**)

$$(Identity)\ \frac{}{x : A \vdash x : A} \qquad\qquad (Cut)\ \frac{\Gamma \vdash t : A \qquad \Delta, x : A \vdash t' : B}{\Gamma, \Delta \vdash t'[{}^{t}/_{x}] : B}$$

(**Logical Rules**) These are the "peculiar" operations of the systems, introducing new formulae (i.e. new types) in the proof. The unique new formula introduced by each rule is called the *principal* formula of the inference. If the principal formula is in the rhs of the final sequent, the inference rule is called *right*; it is called *left* otherwise. Right and left introduction rules respectively correspond to *constructors* and *destructors* in IS's. The shape of these rules is:

$$\frac{\Gamma_1,\, \vec{x}^{\,1} : \vec{A}^1 \vdash t_1 : B_1 \quad \cdots \quad \Gamma_m,\, \vec{x}^{\,m} : \vec{A}^m \vdash t_m : B_m \qquad \Delta, z : C \vdash t : D}{\Gamma_1, \cdots, \Gamma_m, \Delta, y : \mathtt{T}_{\mathbf{d}}(\vec{A}^1, B_1, \cdots, \vec{A}^m, B_m, C) \vdash t[{}^{\mathbf{d}(y, \vec{x}^{\,1}.t_1, \cdots, \vec{x}^{\,m}.t_m)}/_{z}] : D}$$

for left introduction rules (destructors), and

$$\frac{\Gamma_1,\, \vec{x}^{\,1} : \vec{A}^1 \vdash t_1 : B_1 \quad \cdots \quad \Gamma_n,\, \vec{x}^{\,n} : \vec{A}^n \vdash t_n : B_n}{\Gamma_1, \cdots, \Gamma_n \vdash \mathbf{c}(\vec{x}^{\,1}.t_1, \cdots, \vec{x}^{\,n}.t_n) : \mathtt{T}_{\mathbf{c}}(\vec{A}^1, B_1, \cdots, \vec{A}^n, B_n)}$$

for right introduction rules (constructors). Above $\mathtt{T}_{\mathbf{d}}(\vec{A}^1, B_1, \cdots, \vec{A}^m, B_m, C)$ and $\mathtt{T}_{\mathbf{c}}(\vec{A}^1, B_1, \cdots, \vec{A}^n, B_n)$ are formulae (types) built up by means of the types they take as arguments and they are equal, provided they correspond to the same logical operator. The contexts Γ_i, Δ have different variable names. We remark that the above rules have the general shape of the so-called multiplicative connectives (implication, conjunction, etc.). As in IN's, there is no means to represent directly additives (disjunction, etc.) (see Section 3.2.1 in [Laneve 1993])

The IS-expressions are exactly those encoding proofs of the above systems (those in the rhs of the lower sequents in the rules) when any typing constraint is removed. For instance, λ-calculus is obtained from an intuitionistic system defining the implication:

$$(\rightarrow\ left)\ \frac{\Delta \vdash t : A \qquad z : B, \Gamma \vdash t' : C}{\Delta, y : A \rightarrow B, \Gamma \vdash t'[{}^{@(y,t)}/_{z}] : C} \qquad (\rightarrow\ right)\ \frac{\Delta, x : A \vdash t : B}{\Delta \vdash \lambda(x.t) : A \rightarrow B}$$

Following Lafont's IN's, we may provide a graphical representation of IS-forms. This will explain some important relations between the arity of the forms and the way they link upper sequents in a proof. In the notation of IN's, a proof of a sequent $A_1, \ldots, A_n \vdash B$ is depicted as a graph with $n+1$ conclusions, n with a negative type (the inputs) and one with a positive type (the output). In particular, an axiom is just a line with one input and one output.

Every logical rule is represented by introducing a new operator in the net (a new labeled node in the graph). The operator has a *principal port*, individuated by an arrow, which is

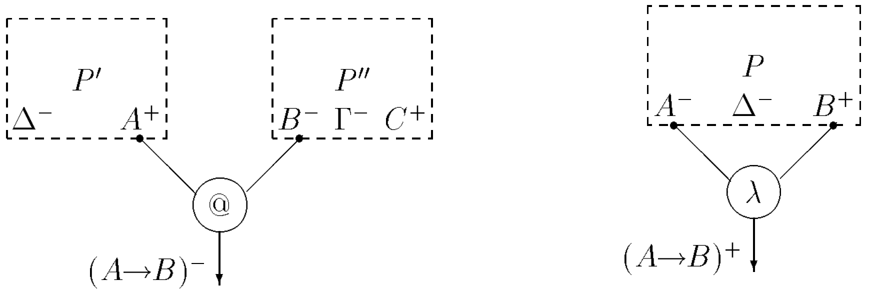


Figure 1: The graphical representation of @ and λ.

associated with the *principal formula* of the logical inference. In Figure 1 we have illustrated the two logical rules for implication.

The principal port of each operator may be either an input or an output. In the first case it corresponds to a new logical assumption in the left hand side of the sequent (as for @), and the operator is a *destructor*; in the second case it corresponds to the right hand side of the sequent (as for λ), and the operator is a *constructor*. The other ports of the agents are associated with the *auxiliary formulae* of the inference rule, that is the distinguished occurrences of the formulae in the upper sequents of the rule. In the two rules above, the auxiliary formulae are A and B.

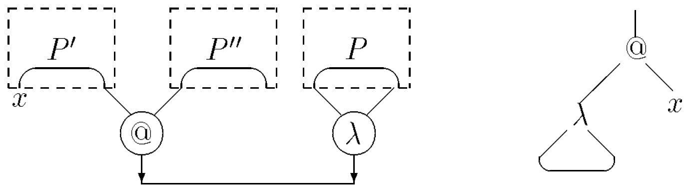
The auxiliary ports of an agent may be either inputs or outputs, independently from the fact that it is a constructor or a destructor. Actually, in the general theory of Interaction Nets, which is inspired by *classical* (linear) logic, there is a complete symmetry between constructors and destructors, and no restriction at all is imposed on the type of the auxiliary ports (in other words, there is a complete symmetry between inputs and outputs). On the contrary, the fact of limiting ourselves to the intuitionistic case, imposes some obvious "functional" constraints.

Note first that auxiliary formulae may come from different upper sequents or from a single one. For instance, the auxiliary ports of @ are in different upper sequents, while those of λ are in a same one. Lafont expresses this fact by defining *partitions* of auxiliary ports. So, in the case of @, A and B are in different partitions, while in the case of λ they are in the same partition. Moreover, the polarity of an auxiliary port is the opposite of the polarity of the conclusion it has to be matched against. Then, the intuitionistic framework imposes the following constraints:

- In every partition there is at most one negative port. If a negative port exists, we shall call it an *input* partition; otherwise it is an *output* partition.

- Every agent has exactly "one output" (functionality). In particular, if the agent is a constructor, the main port is already an output, and all the partitions must be inputs. Conversely, in the case of destructors, we have exactly one output partition among the auxiliary ports, and this partition has to be a singleton.

In Figure 2 we illustrate both the Interaction Net representation of the term $(\lambda z.\, z)x$ and its syntax tree.



(a) (b)

Figure 2: Representations of $(\lambda z.\, z)x$: (a) the Interaction Net, (b) the syntax tree.

Remark 1 *1. The partitions of the auxiliary ports are meant to express a binding mechanism. In case all partitions are singletons (*discrete *case, in [Lafont 1990]) the net is acyclic, corresponding to a first order rewriting system.*

2. The requirement that output partitions should be singletons is not strictly needed from the intuitionistic point of view. However we may assume a stronger point of view, namely to regard every agent *as a functional (single output) operator. Since this approach remarkably simplifies the concrete syntax, we have finally adopted it.*

Note that the problem only concerns destructors because the output of constructors is the principal port, that is always unique. The main example of agent which is captured by assuming the "weak" constraint (one output partition) but not from our "strong" constraint (one singleton *output partition) is the "contraction" operator (i.e. the operator of duplication). Indeed, it is a destructor with a single output partition* $\{+,+\}$*. Another example is provided by the operator of "dereliction" (erasing), whose output partition is empty. Since we are going to add these "structural" operators as an integrated part of our systems, the loss of generality is largely compensated by the gain in clarity.*

2.2 Dynamics

The main theorem of sequent calculi is that stating the redundancy of cut-rules, (i.e. every proof can be effectively turned into a cut-free one). From the operational point of view, this gives dynamics, since it equips the logical system with a rewriting systems on proofs. The rewriting rules must be specified *a priori*: a proof ending into a cut can be simplified into another one by means of some mechanism that is characteristic of that cut.

In order to ensure the possibility of eliminating *all* cuts, the cut-elimination process (term rewriting) must *terminate*. In general, this property does not hold in IS's. We just have a general correspondence between IS's and systems with an *intuitionistic nature*, but only *a posteriori* we may actually check if a particular system holds good "logical" properties (cut-elimination, subformula property, etc.).

Obviously, we could proceed the other way round, imposing some sufficient conditions on IS's (typing, first of all) to establish a tighter relation with logic. This is surely an interesting subject, but it is out of the scope of this paper. So, we shall merely focus on the *dynamic* aspects of cut-elimination, without worrying about termination and removing any typing discipline.

Foremost, there are several kinds of cuts. The most interesting case is the *logical cut*, i.e. a cut between two dual (left-right) introduction rules for the same principal formula. In all the other cases (structural cuts), Intuitionistic Systems "eliminate" the cut by lifting it in the premise(s) of one of the rules preceding the cut (that becomes the last rule of the new proof). As in the case of λ-calculus, these kind of cuts are essentially "unobservable" in IS's (they are dealt with in the metalevel definition of substitution). So, let us concentrate on logical cuts only.

Let us consider the case of implication. Here is the typical situation:

$$\dfrac{\dfrac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda(x.t) : A \rightarrow B} \qquad \dfrac{\Delta \vdash t' : A \qquad y : B, \Theta \vdash t'' : C}{\Delta, z : A \rightarrow B, \Theta \vdash t''[{}^{@(z,t')}/_{y}] : C}}{\Gamma, \Delta, \Theta \vdash t''[{}^{@(z,t')}/_{y}][{}^{\lambda(x.t)}/_{z}] : C}$$

The elimination of the above cut consists in introducing two other cuts (see [Girard *et al.* 1989]). The rewritten proof is:

$$\dfrac{\Delta \vdash t' : A \qquad \dfrac{\Gamma, x : A \vdash t : B \qquad y : B, \Theta \vdash t'' : C}{\Gamma, x : A, \Theta \vdash t''[{}^{t}/_{y}] : C}}{\Gamma, \Delta, \Theta \vdash t''[{}^{t}/_{y}][{}^{t'}/_{x}] : C}$$

Note that this meta-operation on proofs induces a rewriting rule in the underlying IS, which is, in this case, β-reduction. Indeed, the equation $t''[{}^{@(z,t')}/_{y}][{}^{\lambda(x.t)}/_{z}] = t''[{}^{t}/_{y}][{}^{t'}/_{x}]$, implies β-reduction.

Remark 2 *The main property of the above cut-elimination is the* locality*: i.e., let L and R be the left and right sequent in the cut-rule, then the process just works on the "interfaces" of the proofs ending into the premises of L and R, considering as "black boxes" the proofs themselves.*

In general, besides the *locality property*, we want to avoid the *creation of new hypotheses* in the final sequent of rewritten proof, i.e. cut elimination cannot create new free variables. This has two implications:

1. the variables bound by L or R (i.e. the auxiliary formulae of the rules) must be suitably filled in (typically with cuts or introducing *new* rules binding them);

2. if a new axiom is introduced by the rewriting, then the hypothesis in the lhs must be consumed (with a cut or by binding it via a logical rule) inside the rewritten proof.

According to statics, a cut in the syntax of IS's corresponds to a term of the following kind:

$$\mathbf{d}(\mathbf{c}(\vec{x}^{\,1}.\, X_1, \cdots, \vec{x}^{\,m}.\, X_m), \cdots, \vec{x}^{\,n}.\, X_n).$$

The X_i represent the proofs ending into the upper sequents of L and R (the "black boxes" above). Following the previous discussion, in the right hand sides of an IS-rewriting rule, we may either

- introduce new variables with axioms or with weakenings (denoted by x),

or, starting from proofs $H_1, \cdots, H_n$ that have been already built up, we may

- exploit proofs X_i, provided we fill the bound hypothesis (notation $X[^{H_1}/_{x_1}, \cdots, ^{H_n}/_{x_n}]$)
- or introduce a new logical rule (denoted as $\mathbf{f}(\vec{x}^{\,1}.\, H_1, \cdots, \vec{x}^{\,n}.\, H_n)$).

The other logical rules (contractions, cuts) are visible in the concrete syntax of IS's under copying or interactions. Finally, the syntactical constraint reflecting the logical absence of new hypotheses is: right hand sides of rules must be closed expressions.

Example 1 (Naturals) *We have two constructors* `0` *and* `succ`*, respectively associated with the following right introduction rules:*

$$(nat, right_0) \quad \vdash \mathtt{0} : nat \qquad\qquad (nat, right_S) \quad \frac{\Delta, \vdash n : nat}{\Delta \vdash \mathtt{succ}(n) : nat}$$

A typical destructor is `add`*.*

$$(nat, left_{add}) \quad \frac{\Delta \vdash p : nat \qquad \Gamma, y : nat \vdash t : A}{\Delta, \Gamma,\ x : nat \vdash t[^{\mathtt{add}(x,p)}/_y] : A}$$

where A can be any type. The following is an example of cut:

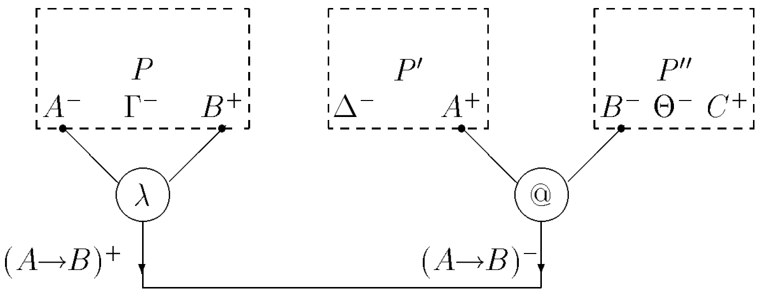
$$\frac{\vdash \mathtt{0} : nat \qquad \dfrac{\Delta \vdash p : nat \qquad y : nat, \Theta \vdash t : A}{\Delta, x : nat, \Theta \vdash t[^{\mathtt{add}(x,p)}/_y] : A}}{\Delta, \Theta \vdash t[^{\mathtt{add}(x,p)}/_y][^{\mathtt{0}}/_x] : A}$$

that is simplified into:

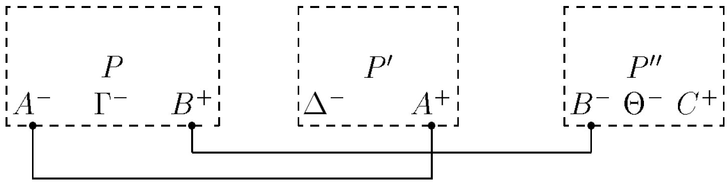
$$\frac{\Delta \vdash p : nat \qquad\qquad y : nat, \Theta \vdash t : A}{\Delta, \Theta \vdash t[^p/_y] : A}$$

The above elimination induces the IS-rule $\mathtt{add}(\mathtt{0}, X) \to X$*, corresponding to the equation* $t[^{\mathtt{add}(x,p)}/_y][^{\mathtt{0}}/_x] = t[^p/_y]$*. Notice that the rule, unlike intuitionistic proofs, has no typing constraint.*

Back to Interaction Nets, the proof specifying a cut of implication is illustrated by the following interaction between λ and @:



and the simplification can be depicted by replacing the two agents @ and λ by the connections below (that are new cuts):

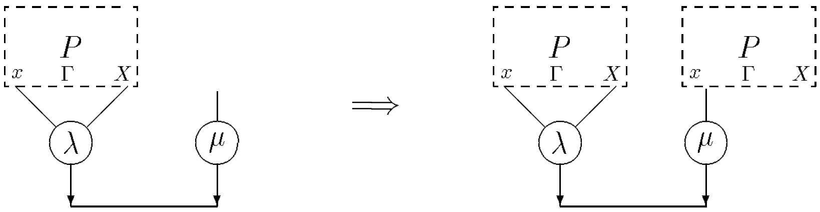


We remark that, in the above reduction, just the connections of the *interface* of the dashed boxes changes, whilst the contents of the dashed boxes is not affected at all. This is a general property IN's rules: every IN-rewriting rule behaves *linearly* w.r.t. the interface of the agents involved in the reduction and the only effect of the reduction rule is to change the connections of ports in the interface, possibly by introducing new operators. This is the main difference of IS's w.r.t. IN's, since in the first ones this linearity constraint is dropped.

Let us see an example. An interesting destructor for the operator of λ-abstraction is the recursion operator μ. The destructor μ interacts with λ according to the following rewriting rule:

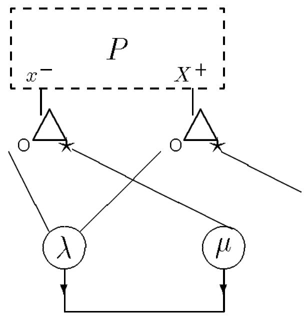
$$\mu(\lambda(x.X)) \;\rightarrow\; X[{}^{\mu(\lambda(x.X))}/_{x}]$$

Note that in the right hand side we use two times the body of the λ; so the rule is not linear. Graphically we have something of the following kind:



where the part of graph corresponding to X has been duplicated. Moreover, all the free variables in P (those marked with Γ in the picture above) must be suitably shared. This means that the subterm X has to be considered as a global entity (a *box*, in Girard's terminology), and we loose the possibility to express the rewriting rule as a local operation interacting with the rest of the world only through an interface.

However the locality of rewriting rules can still be recovered by *implementing* IS's through IN's: we have just to use Lamping's sharing operators. This is the main idea underlying the optimal implementation in [Asperti and Laneve 1993a]. For instance, we can repair the above reduction rule by replacing the graph in the rhs with the following one:



where we have used two sharing operators (*fans*). These operations must be regarded as "auxiliary" agents, which live together with the agents defined by the logical rules. Of course, suitable rules must be provided in order to manage the interactions of auxiliary agents with all the other ones and between themselves.

Remark 3 *Right hand sides of IS's rules are actually more generous than rhs's of IN's rules. In particular Lafont forbids the presence of redexes inside the schema representing the rhs ("optimization" constraint in [Lafont 1990]). This constraint does not simplify in any way the theory of optimality, so we prefer to drop it in IS's.*

2.3 Back from Interaction Systems

So, we have shown that every intuitionistic system can be encoded by an IS. In order to show the *vice versa*, we must solve the problem of the lack of a type discipline of IS's (as in the case of the pure λ-calculus). One possibility is to generalize intuitionistic systems by dropping any type discipline. Instead of investigating this (straightforward) generalization, we have pursued an alternative approach in [Asperti and Laneve 1994], which is more useful in the perspective of optimal implementations. We have implemented any IS into *pure nets*, a type-free variant of Girard's Proof Nets, that have been studied in [Danos 1990, Malacaria and Regnier 1991, Regnier 1991] with the aim of encoding pure λ-calculus. The implementation, on one hand, confirms the logical foundation of IS's and, on the other hand, provides the intermediate step towards the *optimal* implementation in [Asperti and Laneve 1993a, Laneve 1993]. We refer the interested reader to [Asperti and Laneve 1994] for a detailed discussion.

2.4 Discrete Interaction Systems

An interesting subclass of IS's is the discrete one, i.e. those systems without any binding feature. In this case the rewriting rules have the following general shape:

$$\mathbf{d}(\mathbf{c}(X_1, \cdots, X_m), \cdots, X_n) \;\rightarrow\; t$$

where t is an element of the term algebra over $\{X_1, \cdots, X_n\}$.

Example 2 *A typical example of discrete IS is* Primitive Recursion. *We have only two constructors* `0` *and* `succ`. *Composition of two functions* $\mathbf{f}(-)$ *and* $\mathbf{g}(-)$ *is obviously defined as* $\mathbf{f}(\mathbf{g}(-))$*, and definition by primitive recursion has already the right form, indeed:*

$$\begin{array}{l} \mathbf{d}(\mathtt{0},\, X) \;\rightarrow\; \mathbf{h}(X) \\ \mathbf{d}(\mathtt{succ}(X), Y) \;\rightarrow \mathbf{f}(X,\, Y,\, \mathbf{g}(X, Y)) \end{array}$$

For instance, we may define addition and multiplication:

$$\begin{array}{l} \mathtt{add}(\mathtt{0},\, X) \;\rightarrow\; X \\ \mathtt{add}(\mathtt{succ}(X),\, Y) \;\rightarrow\; \mathtt{succ}(\mathtt{add}(X,\, Y)) \\[1ex] \mathtt{mult}(\mathtt{0},\, X) \;\rightarrow\; \mathtt{0} \\ \mathtt{mult}(\mathtt{succ}(X),\, Y) \;\rightarrow\; \mathtt{add}(Y,\, \mathtt{mult}(X,\, Y)) \end{array}$$

Example 3 *Discrete IS's are Turing complete. A simple way to prove it is by coding Combinatory Logic (CL). Recall that CL is defined by the two rules:* $\mathbf{S}xyz \rightarrow xz(yz)$ *and* $\mathbf{K}xy \rightarrow x$*. These rules do not fit with the rewriting rules of IS's; we must break them down in more atomic steps, namely:*

$$\begin{array}{lcl} @(\mathbf{S},\, X) \;\rightarrow\; \mathbf{S}_1(X) & \qquad & @(\mathbf{K},\, X) \;\rightarrow\; \mathbf{K}_1(X) \\ @(\mathbf{S}_1(X),\, Y) \;\rightarrow\; \mathbf{S}_2(X,\, Y) & & @(\mathbf{K}_1(X),\, Y) \;\rightarrow\; X \\ @(\mathbf{S}_2(X,\, Y),\, Z) \;\rightarrow\; @(@(X,\, Z),\, @(Y,\, Z)) & & \end{array}$$

That is, we have five constructors $\mathbf{S}$, $\mathbf{S}_1(-)$, $\mathbf{S}_2(-,-)$, $\mathbf{K}$, $\mathbf{K}_1(-)$ *and only one destructor* $@$*: they provide the obvious "implementation" of CL.*

W.r.t. optimality, the main property of discrete IS's is that it is not difficult to implement optimal sharing (as in every first order system). Indeed, since we do not have bindings (i.e. the net is acyclic), we never introduce *fan-outs* in the reduction. This implies that we do not even need control operators to match fan-ins and fan-outs.

There is an extremely important point to be understood here. Since it is easy to provide an optimal implementation of discrete IS's, that are already Turing Complete, one may wonder what is the interest to consider higher order systems, where the correct handling of sharing becomes much more difficult. For instance, in the case of λ-calculus, one could compile a λ-term M in a term M' of CL, and then reduce M' in an optimal way.

The problem is that the optimal reduction of M' has nothing to do with optimality in the λ-calculus. Let us consider the following example. We assume the usual encoding of λ-abstraction in CL, namely:

$$
\begin{aligned}
[\![\lambda x.\, x]\!] &= \mathbf{SKK} \\
[\![\lambda x.\, M]\!] &= \mathbf{K}M \qquad\qquad \text{if } x \notin \mathtt{var}(M) \\
[\![\lambda x.\, MN]\!] &= \mathbf{S}([\![\lambda x.\, M]\!])\,([\![\lambda x.\, N]\!])
\end{aligned}
$$

Consider a term $M = (\lambda x.\, x(x(y)))(\lambda z.AB)$. If you translate M in CL and you reduce it as a discrete IS according to the rules above, you end up in the situation illustrated in Figure 3. Note now that the subterm $\mathbf{S}_2([\![\lambda z.\, A]\!], [\![\lambda z.\, B]\!])$ is the translation of $\lambda z.AB$, so

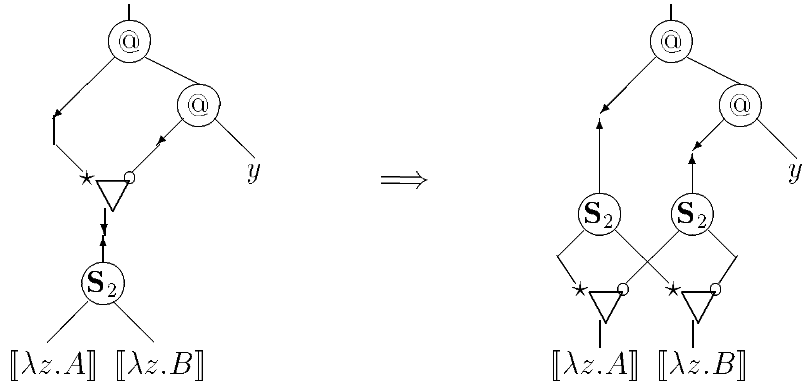


Figure 3: An example with combinators

the duplication of $\mathbf{S}_2(-,-)$ corresponds to a duplication of the application AB. If AB was a redex, this is eventually duplicated. For instance, the optimal reduction of $\mathtt{n}(\mathtt{m}I)x$ (where $\mathtt{n}$ and $\mathtt{m}$ are Church integers), once translated in CL, takes a number of steps which grows as $\mathtt{n} * \mathtt{m}$, while an optimal implementation of the λ-calculus should take a time proportional to $\mathtt{n} + \mathtt{m}$.

The previous example depends from the particular encoding of λ-abstraction in CL (but not from our IS-implementation of CL, which is optimal). However, the problem seems to be completely general: when we translate a λ-term M in CL-term M', there will be applications in M which are not translated by applications in M', but coded as arguments of $\mathbf{S}$. When one of these $\mathbf{S}$ is shared, it must be duplicated, that implies a blind, possibly unoptimal duplication of the application.

3 The syntax

Let us introduce IS's in a more formal way. An Interaction System is defined by a *signature* Σ and a set of *rewriting rules* R.

(**The signature**) The signature Σ consists of a denumerable set of *variables* and a set of *forms*. The set of forms is partitioned into two disjoint sets Σ^+ and Σ^-, representing *constructors* (ranged over by $\mathbf{c}$) and *destructors* (ranged over by $\mathbf{d}$). Variables will be ranged over by x, y, z, $\cdots$, possibly indexed. Vectors of variables will be denoted by $\vec{x}^{\,i}$ where i is the length of the vector (often omitted). Vectors of length 1 will be represented directly by the variable name.

Each form can work as a binder. This means that in the arity of the form we must specify not only the number of arguments but also, for each argument, the number of variables it is supposed to bind. Thus, the *arity* of a form $\mathbf{f}$, is a finite (possibly empty) sequence of naturals (and not, as usual, a natural!). Moreover, we have the constraint that the arity of every destructor $\mathbf{d} \in \Sigma^-$ has a leading 0 (i.e., it cannot bind over its first argument). The reason for this restriction is that, in Lafont's notation [Lafont 1990], at the first argument we find the *principal port* of the destructor, that is the (unique) port where we will have interaction.

Expressions, ranged over by $t, t_1, \cdots$, are inductively generated by the two rules below:

a. every variable is an expression;

b. if $\mathbf{f}$ is a form of arity $k_1 \cdots k_n$ and $t_1, \cdots, t_n$ are expressions then $\mathbf{f}(\vec{x}^1_{k_1}.\, t_1, \cdots, \vec{x}^n_{k_n}.\, t_n)$ is an expression.

Free and bound occurrences of variables are defined in the obvious way. As usual, we will identify terms up to renaming of bound variables (α-conversion).

(The rewriting rules) Rewriting rules are described by using schemas or *metaexpressions*. A metaexpression is an expression built up also with *metavariables*, ranged over by X, Y, $\cdots$, possibly indexed. Metaexpressions will be denoted by $H, H_1 \cdots$.

A *rewriting rule* is a pair of metaexpressions, written $H_1 \to H_2$, where H_1 (the left hand side of the rule, shortened into lhs) has the following format

$$\mathbf{d}(\mathbf{c}(\vec{x}^1_{k_1}.\, X_1, \cdots, \vec{x}^m_{k_m}.\, X_m), \cdots, \vec{x}^n_{k_n}.\, X_n)$$

and $i \neq j$ implies $X_i \neq X_j$ (*left linearity*). The arity of $\mathbf{d}$ is $0k_{m+1} \cdots k_n$ and that of $\mathbf{c}$ is $k_1 \cdots k_m$.

The right hand side (rhs) H_2 is every *closed* metaexpression (every *new* variable $x, y, \cdots$ must be bound by a *new* form), whose metavariables are already in the lhs and built up by the following syntax

$$H \ ::= \ x \ \mid \ \mathbf{f}(\vec{x}^{\,1}_{a_1}.\, H_1, \cdots, \vec{x}^{\,j}_{a_j}.\, H_j) \ \mid \ X_i[{}^{H_1}/_{x^i_1}, \cdots, {}^{H_{k_i}}/_{x^i_{k_i}}]$$

The expression $X[{}^{H_1}/_{x_1}, \cdots, {}^{H_n}/_{x_n}]$ denotes a meta-operation of substitution, as in the λ-calculus.

Finally, the set of rewriting rules must be *non-ambiguous*, i.e. there exists at most one rewriting rule for every pair $\mathbf{d}$-$\mathbf{c}$.

Example 4 *(***The λ-calculus***) We have a destructor @ of arity 00, and a constructor λ of arity 1. The only rewriting rule is β-reduction:*

$$@(\lambda(x.X), Y) \ \to \ X[{}^{Y}/_{x}].$$

Example 5 *Interesting Interaction Systems can be defined by enriching the* λ*-calculus with other rules. For instance, it is possible to extend the* λ*-calculus with an* if-then-else *operator* $\natural$. *Let* $\mathbf{T}$ *and* $\mathbf{F}$ *be two (new) constructors of arity* ε, *and* $\natural$ *be a destructor of arity* 000. *The rules for the conditional are described by the following interactions between* $\natural$ *and* $\mathbf{T}$ *or* $\mathbf{F}$*:*

$$\natural(\mathbf{T}, X, Y) \to X$$

$$\natural(\mathbf{F}, X, Y) \to Y$$

Note that we do not have any interaction between $\natural$ *and* λ.

A less trivial example is provided by the recursion operator μ. *We may describe it as a destructor of arity* 0, *interacting with* λ *as follows:*

$$\mu(\lambda(x.X)) \to X[^{\mu(\lambda(y.X[^y/_x]))}/_x]$$

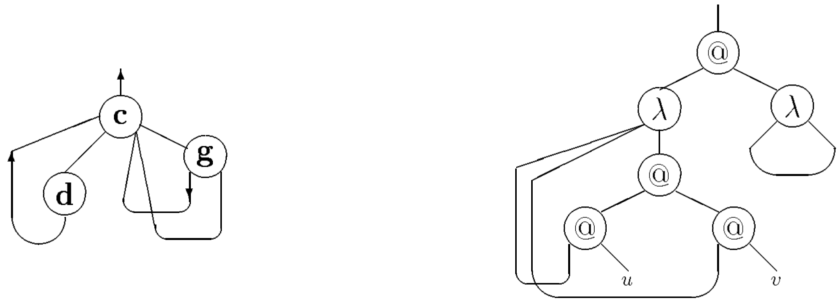
Note that the μ *and the* λ *are different from those in the lhs: the latter ones are responsible of the* creation *of the first ones.*

3.1 Graphical representations

As discussed in Section 2, IS-expressions have graphical representations that are reminiscent of IN's. In particular, a form $\mathbf{f}$ of arity $k_1 \dots k_n$ is represented as a node of name $\mathbf{f}$ with $1 + \sum_{i=1}^{n} p_i$ ports (edges); $p_i = k_i + 1$ is the i-th *partition* of $\mathbf{f}$. The i-th partition represents the connections between $\mathbf{f}$ and its i-th argument M. In the i-th partition, one connection is with the root of M, and k_i with the variables bound by $\mathbf{f}$: the latter ones will be called *bound* ports of the partition. Edges coming into bound ports are called *bound edges*; an edge which is not bound is said *free*. Observe in Figure 4 below that bound variables correspond to (bound) ports of the form $\mathbf{f}$. Thus our graphs are cyclic and represent terms up-to α-conversion.

The unique port which does not belong to a partition is called the output port of $\mathbf{f}$. All the forms, have a *principal port*, which is drawn with an outgoing arrow (the arrow is omitted when it is clear from the context). The other entries are called *auxiliary ports* (see [Lafont 1990]). In the case of a constructor, the principal port coincides with the output port. In the case of a destructor, the principal port is the unique edge in the first partition (recall that the arity of the first argument of a destructor is eventually 0, so this partition is a singleton and does not have bound ports).

For example the graphical representation of $\mathbf{c}(\langle x_1, x_2\rangle.\,\mathbf{d}(x_1),\, y.\,\mathbf{g}(y, y))$ is illustrated in Figure 4.(a). Variables which are not bound will be depicted as dangling edges whose ends are labeled by the name of the variables. This is the case for the variables u and v in the λ-term $(\lambda x.(xu)(xv))(\lambda y.y)$ depicted in Figure 4.(b). We recall that, in this way, several edges may have a common end, due to the multiple occurrence of a free variable in an expression.



(a) (b)

Figure 4: Graphical representations of expressions

4 Preliminaries (residuals and standardization)

The aim of this section is to fix the notation, rather than to provide an accurate account of the properties of IS's. We refer to [Aczel 1978, Klop 1980] for all the details.

We shall denote *redexes* through the *access path* to the corresponding subexpression, defined as follows:

- ε (the empty string) is the access path of the root;
- if u is the access path of $\mathbf{f}(\vec{x}^{\,1}.\,t_1, \cdots, \vec{x}^{\,n}.\,t_n)$ then $u \cdot i$ is the access path of t_i.

Reductions will be represented by the access paths of the corresponding redex in bold or by $t \xrightarrow{u} t'$, when we want to emphasize the initial and final expressions. Sequences of reductions, called *derivations*, will be ranged over by $\sigma, \rho, \cdots$. Again the notation $t \xrightarrow{\sigma}\!\!\!\!\rightarrow t'$ will emphasize initial and final expressions. We denote with $\underline{\emptyset}$ the empty derivation.

The notation $v/\mathbf{u}$ stays for the *residuals* of the redex v after the contraction $\mathbf{u}$. A way to formalize residuals of a redex v w.r.t. $t \xrightarrow{u} t'$ is by marking v and taking all the marked redexes in t'. Notice that $v/\mathbf{u}$ is in general a (possibly empty) set of redexes. See [Lévy 1980, Laneve 1993] for an alternative definition of residuals.

The operation of residual can be extended to derivations as follows (σ and ρ are supposed to begin at the same term):

$$\sigma/\rho \; = \; \begin{cases} \underline{\emptyset} & \textit{if } \sigma = \underline{\emptyset} \\ (\sigma'/\rho); v/(\rho/\sigma') & \textit{if } \sigma = \sigma'; \mathbf{v}. \end{cases}$$

Notice that each step of σ/ρ is a contraction of a *set* of redexes. The well-definedness of σ/ρ follows by the Parallel Moves Lemma (see [Lévy 1980] for the λ-calculus and [Klop 1980] for the generalization to Combinatory Reduction Systems). The notation for the reduction firing an empty set of redexes is $\underline{\emptyset}$, the same as for empty derivations (they must be considered as equal). We denote by $\sigma \sqcup \rho$ the derivation $\sigma; \rho/\sigma$.

Two properties are essential in the following: Church-Rosser and existence of a standard derivation. These properties hold in a *strong* form, equating both the final terms and the derivations leading to them. The definition of the equivalence of derivation is the following.

Definition 1 *The* permutation equivalence *(notation $\equiv$) is the least equivalence relation satisfying the axioms:* $\mathbf{U} \sqcup \mathbf{V} \equiv \mathbf{V} \sqcup \mathbf{U}$ *and if* $\sigma \equiv \rho$ *then* $\tau; \sigma; \tau' \equiv \tau; \rho; \tau'$*.*

Theorem 1 *(***Church-Rosser***: Klop [Klop 1980]) For every pair σ and ρ of coinitial derivations (in Left Normal CRS's), $\sigma; \rho/\sigma \equiv \rho; \sigma/\rho$ holds.*

Theorem 2 *(***Standardization Theorem***, Klop [Klop 1980]) Every derivation σ (in a Left Normal CRS's) is permutation equivalent to the unique one firing always the leftmost-outermost redex fired by σ.*

Given a derivation $t \stackrel{\sigma}{\longrightarrow\!\!\!\!\rightarrow}$, let $\mathtt{lmc}(\sigma)$ be the *leftmost-outermost* redex in t contracted along σ. The *standard* derivation satisfying the above theorem is $\mathbf{v}_0; \mathbf{v}_1; \cdots$, where v_i are defined through the following procedure:

$$\begin{array}{ll} \sigma_0 = \sigma & v_0 = \mathtt{lmc}(\sigma) \\ \sigma_{i+1} = \sigma_i / v_i & v_{i+1} = \mathtt{lmc}(\sigma_{i+1}) \end{array}$$

Finally, we shall also use a smooth generalization of the notion of residuals. The residuals of a form $\mathbf{f}$ in t after the reduction $t \stackrel{u}{\longrightarrow} t'$ are obtained by marking $\mathbf{f}$ in t and taking all the marked $\mathbf{f}$ in t'.

5 The family relation and the labeling

The notion of family relates those redexes that are copies of a same redex along a derivation. For example, take $t = \Delta(\mathbf{I}z)$ ($\Delta = \lambda x.xx$ and $\mathbf{I} = \lambda x.x$). Then the reduction $t \longrightarrow (\mathbf{I}z)(\mathbf{I}z)$ duplicates the innermost redex in t. In this case, we say that the two redexes in the final expression are in the same *family*. In spite of the intuitive nature of this notion, the formalization is not easy.

For instance, the above example suggests that two redexes are in the same family provided they are residuals of a common redex. This isn't a necessary condition. Consider the λ-term $t = \Delta(\mathbf{II}z)$. There are two redexes in t: the outermost u and the innermost $\mathbf{II}$, that we call v. If u is performed before v, we get the intermediate expression $t' = (\mathbf{II}z)(\mathbf{II}z)$, where the two redexes $\mathbf{II}$ are both residuals of v. Contracting (the residuals of) v in t' yields the expression $t'' = (\mathbf{I}z)(\mathbf{I}z)$ having two redexes r' and r'' *that are not residuals of any redex in* t'. However, intuitively, these newly created redexes are "copies" of a same redex: an easy way to recognize this is to perform v as first reduction, obtaining $\Delta(\mathbf{I}z)$ (let r be the innermost redex). By contracting the (unique) residual of u we get t'', and we may now observe that r' and r'' are both residuals of r. That is, we have found an alternative, permutation equivalent derivation, where the two redexes r' and r'' are residuals of a common redex.

The purpose of the notion of family is pretty operational: we could imagine an evaluator of the (functional) language which copes with the phenomenon of duplication by *keeping shared* subexpressions *as much as possible* (i.e. the sharing it performs is *optimal*). Notice that, due to the sharing, such an evaluator reduces at each computational step several redexes of the corresponding "flattened" syntactical expression. The idea is that the evaluator always reduces exactly all the redexes which are in the same family. Due to the informality of the notion of optimal sharing, we can hardly prove the correctness of any optimal evaluator. The approach to the formalization of the notion of optimal sharing is to start from several different perspectives, and to prove that all these approaches lead to the same formal notion.

For this reason, Lévy proposed three formal approaches to the family relation, and proved their equivalence: the labeling, the extraction process and the zig-zag relation. Since the zig-zag cannot be generalized to the case of IS's (see Section 9) we will focus in this section and in the following one on the first two approaches.

We recall that the problem of implementing optimal reductions has been recently solved by Lamping [Lamping 1990], using a complex graph reduction technique.

5.1 The labeled Interaction Systems

The *labeled* Interaction Systems extend the IS's by marking all *points of contact* between operators, and keeping information about these connections along the reduction (labels dynamically grow during a reduction). Thus, labels implicitly provide a trace of the history of the creation of each redex.

In order to avoid some annoying problems concerning the associativity of the concatenation operator over labels, our presentation will be slightly different (but equivalent) w.r.t. Lévy's one. In particular, we shall delay the concatenation of labels until it is required for firing a redex.

Definition 2 *Let $L = \{a, b, \cdots\}$ be a countable set of* atomic labels. *The set* $\mathbf{L}$ *of* labels, *ranged over by α, $\beta, \cdots$ is defined by the following rules:*

$$L \quad | \quad \alpha\beta \quad | \quad (\alpha)_s$$

where s is a string of positive integers. The operation of concatenation $\alpha\beta$ will be assumed associative. *The set* $\mathbf{L}_p$ *of* proper labels *contains exactly all those labels $\ell \in \mathbf{L}$ such that ℓ is atomic or $\ell = (\ell')_s$, for some s. The set $\mathbf{L}_p$ will be ranged over by α, $\beta, \cdots$.*

Given a label ℓ, `at`*(ℓ) is the set of all its* atomic *(sub)labels.*

The idea behind the following labeling is the same of that of λ-calculus: marking with the degree of the redex any "point of contact" created by the rule. Let us come to the formal definition. Every IS (Σ, R) can be turned in a free way into a (*labeled*) CRS (Σ^L, R^L).

The forms of Σ^L are those in $\Sigma \cup \mathbf{L}$ with the arity of every $\alpha \in \mathbf{L}$ equal to 0. If

$$\mathbf{d}(\mathbf{c}(\vec{x}^{\,1}.X_1, \cdots, \vec{x}^{\,m}.X_m), \cdots, \vec{x}^{\,n}.X_n) \;\rightarrow\; H$$

is a rule in R then, for every i and for every i-tuple $\alpha_1, \cdots \alpha_i$, the rule

$$\mathbf{d}(\alpha_1(\cdots(\alpha_i(\mathbf{c}(\vec{x}^{\,1}.X_1, \cdots, \vec{x}^{\,m}.X_m)\cdots), \cdots, \vec{x}^{\,n}.X_n) \;\rightarrow\; \mathcal{L}^0_{\alpha_1\cdots\alpha_i}(H)$$

belongs to R^L, where $\mathcal{L}^s_\alpha$ is defined over metaexpressions as follows

$$
\begin{aligned}
\mathcal{L}_\alpha^s(x) &= (\alpha)_s(x) \\
\mathcal{L}_\alpha^s(\mathbf{f}(\vec{x}^{\,0}.\,H_0,\cdots,\vec{x}^{\,m}.\,H_m)) &= (\alpha)_s(\mathbf{f}(\vec{x}^{\,0}.\,\mathcal{L}_\alpha^{s0}(H_0),\,\cdots,\,\vec{x}^{\,m}.\,\mathcal{L}_\alpha^{sm}(H_m)) \\
\mathcal{L}_\alpha^s(X[{}^{H_0}/_{x_0},\,\cdots,\,{}^{H_n}/_{x_n}]) &= (\alpha)_s(X[{}^{\mathcal{L}_\alpha^{s0}(H_0)}/_{x_0},\,\cdots,\,{}^{\mathcal{L}_\alpha^{sn}(H_n)}/_{x_n}])
\end{aligned}
$$

Example 6 *Let us explain the above labeling in the case of λ-calculus. The β-reduction $@(\lambda(x.\,X),\,Y) \to X[{}^{Y}/_{x}]$ gives rise, in the labeled version, to the following rules:*

$$
@(\alpha_1(\cdots(\alpha_i(\,\lambda(x.\,X)\cdots),\,Y) \;\to\; \mathcal{L}_\ell^0(X[{}^{Y}/_{x}])
$$

where $\ell = \alpha_1\cdots\alpha_i$. Note that, by definition, $\mathcal{L}_\ell^0(X[{}^{Y}/_{x}]) = (\ell)_0(X[{}^{(\ell)_{00}(Y)}/_{x}])$ and Lévy's labeling can be easily recognized by replacing $(\ell)_0$ with $\overline{\ell}$ and $(\ell)_{00}$ with $\underline{\ell}$.

Observe that λ-calculus has exactly two label operations (besides concatenation) because β-reduction can create two types of connections: "towards the top", i.e. the connection linking the context and the outermost symbol of the functional part, and "towards the bottom", namely the connection between the outermost symbol of the argument part and the bound variables. These two connections must be marked in different ways (hence overlinings and underlinings) because they can be (or become) different redexes (i.e. redexes that are not copies). We leave to the reader to find examples.

Labeled expressions have simple graphical representations, that take advantage of the fact that (proper) labels are forms of arity 0. The idea is to depict the graph of the unlabeled term and mark edges with labels, such that the leftmost proper label of an edge is the outermost in the expression, etc. So the skeleton of two expressions that differ for the labels are equal. This means that we shall address redexes of labeled expressions with the same mechanisms of unlabeled ones, without caring at all about labels.

Remark 4 *We remark that labeled IS's are not IS's, because the lhs's of of labeled rewriting rules is not an interaction destructor-constructor. Nevertheless, labeled IS's are Left Normal CRS's, thus Theorems 1 and 2 hold for them (these are the unique properties we shall use). In the following we often will switch from an IS to its labeled variant, talking about derivations and* isomorphic *labeled derivations. Since this notion is intuitive, we allow ourselves a bit of informality (see Sections 3.5 and 3.6 of [Laneve 1993] for the formal stuff).*

Definition 3 *(***Initial labeling***) Let t be a labeled IS-expression. The predicate* `INIT`(t) *is true if and only if the labels of all subterms of t (i.e. the label of each edge of t in its graphical representation) are atomic and pairwise different.*

NOTATION Let t be an IS-expression: we shall denote with t_I the labeled λ-expression such that `INIT`(t_I) and t_I is equal to t up to the labeling. Let $t \xrightarrow{\sigma}\!\!\!\!\rightarrow t'$ and u be a redex in t'. We shall denote with $\partial_t^\sigma(u)$ the *degree* (i.e. the label) of u along the labeled derivation isomorphic to σ and starting at t_I. The fact that u is a redex in the ending term of a derivation σ (namely, u has history σ) is expressed by the notation σu.

Definition 4 *Let ρ and σ be two labeled derivations starting at an expression t. σu and ρv are in the same* family *if and only if $\partial_t^\sigma(u) = \partial_t^\rho(v)$.*

5.2 About the structure of labels

Labels have an "abstract structure": roughly this structure describes how the label has been built up, apart from the name of the atomic labels which appears inside the label. So, in the case of λ-calculus, the structure of a label gives the positions of the atomic labels and the overlinings and underlinings surrounding such positions.

The structure of a label can be defined by means of a weaker form of initial labeling. Consider the following set of atomic labels

$$L^\bullet \ = \ \{\bullet\} \cup \{\bullet_i \mid i \textsf{ is a natural number }\}$$

Labels $\mathbf{L}^\bullet$ are defined as in Definition 2. Given an expression t labeled by elements in $\mathbf{L}^\bullet$, we say that $\mathtt{INIT}^\bullet(t)$ is *true* when every edge joining two free ports is labeled by $\bullet$ and an edge e incoming a bound port is labeled by $\bullet_n$ where n is the number of edges incoming the bound port which are to the left of e.

Definition 5 *Let $t \xrightarrow{\sigma}\!\!\!\!\rightarrow t'$ and $t_\bullet \xrightarrow{\sigma}\!\!\!\!\rightarrow t'_\bullet$ be labeled isomorphic derivations and $\mathtt{INIT}^\bullet(t_\bullet)$. Let u be an edge in t' and $\partial^\sigma_t(u) = \ell$. The* structure *of ℓ, written $\mathtt{str}(\ell)$, is the label $\partial^\sigma_{t_\bullet}(u)$.*

Note that the definition is sound, in the sense that it only depends from the label, and not from the edge (i.e. same labels on different edges have the same structure). We may equivalently define the $\mathtt{str}(\ell)$ by an obvious substitution for atomic symbols from L to $L^\bullet$. Indeed, assuming $\mathtt{INIT}(t)$ and $\mathtt{INIT}^\bullet(t_\bullet)$, it is possible to prove (omitted because standard) that the structure of ℓ is equal to $\ell[\mathtt{s}]$, where $\mathtt{s}$ is the substitution replacing every label in t with the label of the corresponding edge in $t^\bullet$. In the following we shall use both these characterizations of structure, without caring about.

Theorem 3 *(**The Connection-Property**) Let $\mathtt{INIT}(t)$, and let ℓ be a label generated along some labeled reduction σ starting at t. Then ℓ is uniquely determined by its structure $\mathtt{str}(\ell)$ and an occurrence of a, $a \in L$, in $\mathtt{str}(\ell)$.*

The above property is crucial in order to fix the correspondence between labels and the extraction process. The underlying idea behind the Connection-Property is quite simple: given an occurrence of an atomic symbol inside $\mathtt{str}(\ell)$, the surrounding structure uniquely determinates all the other symbols "by connection". The formalization of this latter concept and the proof of the theorem can be found in Appendix A.

6 The extraction relation

The approach to families based on the extraction relation relies on the intuition that two redexes (with histories) σu and ρv are in the same family if and only if the "relevant" part of σu and ρv coincide. In order to define the relevant part, the extraction relation exploits few elementary operations pushing out of histories all the reductions which have not been relevant for the creation of the final redex. This results into a process, called the *extraction process* in [Lévy 1980].

The canonical form obtained at the end of the extraction process essentially expresses the causal dependencies of u along the derivation (we may deal with causal *chains* instead of *partial orders* since only *standard derivations* are considered).

Our approach will differ a little from that in [Lévy 1980] because Lévy's definition, even if smarter, would have the same inadequacy problems of the zig-zag relation when it is generalized to IS's (see Section 9).

6.1 The case of λ-calculus

As we said in Example 4, λ-calculus consists of a unique reduction rule: $@(\lambda(x.X), Y) \to X[{}^{Y}/_{x}]$. So redexes have the shape $@(\lambda(x.X), Y)$, where X and Y are called the functional and the argument part, respectively.

Let u be a β-redex in t. The *set of occurrence numbers* of the functional part X along $\mathbf{u}$, denoted $o_X(X[{}^{Y}/_{x}])$, is $\{\varepsilon\}$; the *set of occurrence numbers* of the argument part Y along $\mathbf{u}$, denoted $o_Y(X[{}^{Y}/_{x}])$, is $o_x(t_X)$, namely the occurrences of the free variable x in the expression t_X instantiating X in u.

Definition 6 *Let* $t \xrightarrow{u} t' \overset{v_1;\cdots;v_n}{\twoheadrightarrow}$. *The derivation* $\mathbf{v}_1;\cdots;\mathbf{v}_n$ *is* internal to the metavariable Z *($Z = X$ or $Z = Y$) of* $\mathbf{u}$ *if there exists an occurrence number* w *of* Z *such that, for every* $1 \le i \le n$*:*

$$\exists v_i'.\ v_i = u \cdot w \cdot v_i' \quad \textbf{and} \quad t_Z \overset{v_1';\cdots;v_i'}{\twoheadrightarrow} \tag{1}$$

where t_Z *is the subexpression of* t *replacing* Z *along* $\mathbf{u}$.

When ρ is internal to the metavariable Z of $\mathbf{u}$, we may lift it up before the firing of u, namely it is possible to perform the reduction ρ before the firing of $\mathbf{u}$. Here is the formalization of this intuition.

Definition 7 (The lift operation) *Let* ρ *be internal to the metavariable* Z *of* $\mathbf{u}$*,* w *be an occurrence number of* Z *and* w' *be the access path of* Z *in the lhs of the rewriting rule. Then the* lift *of* ρ *w.r.t.* u *(notation* $\rho \uparrow_{w}^{w'} u$*) is defined by replacing, inside every reduction of* ρ*, all the prefixes* $u \cdot w$ *with* $u \cdot w'$.

Remark 5 *The operation of lift resembles Lévy's* $\|$ *operation (see [Lévy 1980]) Anyway, there are some differences between the two. Let us recall the definition of* $\|$. *Let* $\mathbf{v};\rho$ *be internal to* u*; then*

$$\begin{array}{rcl} \varnothing \| u & = & \varnothing \\ (\mathbf{v};\rho)\|u & = & v'/\mathbf{u};(\rho/V)\|u \qquad \textbf{where } v \in v'/\mathbf{u} \textbf{ and } V = v'/(\mathbf{u};\mathbf{v}) \end{array}$$

Thus, the derivation $\rho \| u$ is the "completion" of ρ with all the redexes which are co-residuals of some redex in ρ w.r.t. the firing of u. As shown by the following picture, $\rho \uparrow u$ gives a derivation which is coinitial with $\mathbf{u}; \rho$ whilst $\rho \| u$ gives a reduction which is coinitial with ρ.

$$\begin{tikzcd}
t \arrow[r, "\mathbf{u}"] \arrow[d, "\rho \uparrow u"'] & t' \arrow[r, "\rho"] \arrow[d, "\rho \| u"] & {} \\
{} \arrow[r, "\mathbf{u}"] & {} &
\end{tikzcd}$$

It is possible to prove that, in the case of λ-calculus, $\rho \uparrow u; \mathbf{u}$ is permutation equivalent to $\mathbf{u}; \rho \| u$.

Unfortunately, Lévy's definition will not work properly for IS's, because of the possible nesting of two occurrences of a same metavariable. In general, $\rho \| u$ may erase part of the original derivation ρ (see Example 10).

Definition 8 *The* extraction relation $\rhd$ *is the union of the following two relations:*

1. $(\rho; \mathbf{u}; \sigma)v \rhd_1 (\rho; \sigma)v$ if $\mathbf{u}$ and $\sigma; \mathbf{v}$ are two *disjoint* derivations[1]
2. $(\rho; \mathbf{u}; \sigma)v \rhd_2^i (\rho; \sigma \uparrow_w^{w'} u)v'$ if $\sigma; \mathbf{v}$ is internal to the metavariable X_i of $\mathbf{u}$.

where, in 2, w and w' are, respectively, the occurrence number of X_i in the rhs and lhs of the β-rewriting schema, $v = u \cdot w \cdot v''$ and $v' = u \cdot w' \cdot v''$.

The extraction relation has the Church-Rosser property and strongly normalizes. Here is the important theorem.

Theorem 4 *(Lévy, [Lévy 1980]) Let ρ and σ be two standard reductions. Then $\rho u \simeq \sigma v$ if and only if $\rho' u \trianglerighteq \tau w \trianglelefteq \sigma' v$, for some τw, where $\rho'; \mathbf{u}; \rho''$ and $\sigma'; \mathbf{v}; \sigma''$ are the standard forms of $\rho; \mathbf{u}$ and $\sigma; \mathbf{v}$, respectively.*

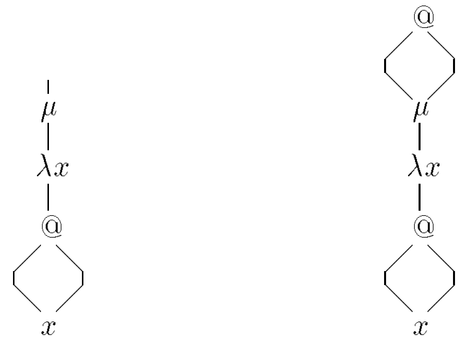
6.2 Counterexamples to Lévy's extraction process

If labeling has a smooth generalization to IS's, this is not the case with the extraction process. Let us begin by adapting Definition 8 to the general case. To this aim it is enough to generalize the notion of occurrence number of a metavariable. Let $\mathbf{u}$ be a reduction and let r_u be the schema contracting $\mathbf{u}$. Every metavariable X_i appearing in the lhs of r_u is matched by $\mathbf{u}$ with a subexpression t_i. Let X be a metavariable of $\mathbf{u}$ and H_u be the rhs of r_u.

An *occurrence number* of X is an element of the set $o_X(H_u)$ defined inductively as follows:

$$\begin{array}{rcl}
o_X(x) & = & \emptyset \\
o_X(\mathbf{f}(\vec{x}^{\,1}.\, H_1, \cdots, \vec{x}^{\,n}.\, H_n)) & = & \bigcup_{1 \le i \le n} i \cdot o_X(H_i) \\
o_X(Y[{}^{H_1}/_{x_1}, \cdots, {}^{H_n}/_{x_n}]) & = & S \cup (\bigcup_{1 \le i \le n} o_{x_i}(t_Y) \cdot o_X(H_i))
\end{array}$$

[1]Two coinitial derivations are *disjoint* if they concern disjoint subexpressions of the initial term.



(a) (b)

Figure 5: Graphical representations of μ-expressions

where $S = \{\varepsilon\}$ if $X = Y$ or $S = \emptyset$, otherwise; the expression t_Y replaces Y in the instance of the rewriting rule and the concatenation of access paths is empty if one of the arguments is empty.

The main problem with this adaptation of Definition 8 is that, in general, it is possible, *at the same time*, to create forms, involve them into redexes and duplicate such redexes. Then the canonical derivation should be the same for every copy. But this is not the case. Let us see an example.

Example 7 *Take the expression $t = \mu(\lambda(x.\,@(x,\, x)))$. By assuming a graphical representation with a sharing mechanism, we could represent t as in Figure 5(a). Then, contracting t with the rule of Example 5, we yield the graph of Figure 5(b), whose corresponding syntactical term is*

$$@(\overbrace{\mu(\lambda(x.\,@(x,\, x)))}^{v'},\ \underbrace{\mu(\lambda(x.\,@(x,\, x)))}_{v''})$$

Notice that the two redexes v' and v'' are shared *into a* unique *redex in Figure 5(b), that is v' and v'' should belong to the same family. Indeed, this can be easily realized by computing their labels. The syntactical duplication of v' and v'' is due to the substitution process only. More precisely, the rewriting rule of the pair μ-λ creates a redex μ-λ along its firing, but this redex appears in the expression which must be substituted for the bound variable. Thus there will be as many copies of the* same *redex μ-λ as the occurrences of x in the body of the λ. We leave to the reader the check that the extraction relation of Definition 8 fails in identifying $\mathbf{u}v'$ and $\mathbf{u}v''$.*

To cope with Example 7 we have to define an operation which gives a representative among a set of derivations that are canonical w.r.t. Lévy's $\trianglerighteq$. The operation we have in mind should choose the derivation creating the leftmost copy as the canonical one. For instance, in Example 7, the canonical representative should be the derivation $\mathbf{u}v'$.

Let $X[{}^{H}/_{x}]$ be a substitution in the rhs of a rule. For a time, we hoped to cope with the problem of duplication by "shifting" a derivation internal to an instance of the expression replacing H towards its isomorphic derivation in the leftmost outermost instance of H.

Whilst this device works well to cope with Example 7, it is not general enough, as it is proved by the following counterexample.

Example 8 *Consider the λ-calculus extended with the following rewriting rules:*

$$\mathbf{d}(\mathbf{c}(Z,\, x.\, X), y.\, Y) \to \mathbf{g}(Z,\, z.\, X[^{Y[^z/_y]}/_x])$$

$$\mathbf{g}(\mathbf{f}(Z),\, x.\, X) \to X[^Z/_x]$$

and take the term

$$M \;=\; \mathbf{d}(\mathbf{c}(\mathbf{f}(\lambda(x.\, x)),\, x.\, @(x,\, x)),\, y.\, @(y,\, y))$$

After two reductions, which we call $\mathbf{u}_1; \mathbf{u}_2$, we get the term

$$M' \;=\; @(@(\lambda(x.\, x),\, \lambda(x.\, x)),\, @(\lambda(x.\, x),\, \lambda(x.\, x)))$$

Let v' and v'' be, respectively, the leftmost and rightmost redexes in M'. The reader can convince himself that v' and v'' are copies by reading M' as

$$M' \;=\; @(x,\, x)[^{@(z,\, z)}/_x][^{\lambda(x.\, x)}/_z]$$

Now, if our shift-operation worked only if a tailing reduction was internal to an argument of a substitution then the shift of v'' towards v' did not succeed because the context created by $\mathbf{u}_1$ is needed in order to realize that v' and v'' are copies. In particular, the applications of the contractions $\mathbf{v}'$ and $\mathbf{v}''$ come from a duplication performed by $\mathbf{u}_1$ whilst the abstractions result from a duplication performed by $\mathbf{u}_2$. That is the creation of the two redexes has been performed in several steps. Therefore, the problem will be to discover subexpressions of M' which have been duplicated in the past and large enough to contain the tailing derivations $\mathbf{v}'$ and $\mathbf{v}''$.

As a matter of facts the situation is further entangled by the subtle interplay between substitutions and creation of forms in the right hand sides of the schemas contracting redexes. In particular, several reductions along the history can participate to the creation of a redex. For instance, one can create the constructor, another the destructor and a third one makes them interact. Therefore, recognizing the presence of copies of a redex u requires, in general, a backward scanning of the history, looking for those reductions that have played some role in the creation of u. This search will be performed by the algorithm `shift-indexes`.

6.3 The shift operation

Let $\mathbf{u}$ be a reduction of an IS and r_u be the schema contracting it. Let H be a sub(meta)-expression of H_u, the rhs of r_u. Mark the subexpression H with a bullet $\bullet$ (in order to avoid pattern matchings with other occurrences of the same metaexpression) and let H' be H marked. An occurrence number of H is an element of the set $o_{H'}(H_u)$ defined inductively as follows:

$$
\begin{aligned}
o_{H'}(x) &= \varnothing \\
o_{H'}(H^\bullet) &= \{\varepsilon\} \\
o_{H'}(\mathbf{f}(\vec{x}^{\,1}.\, H_1, \cdots, \vec{x}^{\,n}.\, H_n)) &= \bigcup_{1 \le i \le n} i \cdot o_{H'}(H_i) \\
o_{H'}(X[^{H_1}/_{x_1}, \cdots, ^{H_n}/_{x_n}]) &= \bigcup_{1 \le i \le n} o_{x_i}(t_X) \cdot o_{H'}(H_i)
\end{aligned}
$$

where t_X replaces X in the rewriting rule $\mathbf{u}$.

Definition 9 *Let $t \xrightarrow{u} t' \overset{v_1;\cdots;v_n}{\twoheadrightarrow}$ and H be a subexpression of the rhs of the rule contracting u. The derivation $\mathbf{v}_1; \cdots ; \mathbf{v}_n$ is* internal *to H if there exists an occurrence number w of H such that, for every r, $v_r \succeq u \cdot w$.*

The core of the operation of shift is an algorithm which computes the access path where the shift has to be performed. Given a standard derivation $(\mathbf{v}_1; \cdots ; \mathbf{v}_n)u$, take a tailing $(\mathbf{v}_{k+1}; \cdots ; \mathbf{v}_n)u$. The (nondeterministic) procedure `shift-indexes` in the table below gives a triple (w_0, w_1, w_2) along which $(\mathbf{v}_{k+1}; \cdots ; \mathbf{v}_n)u$ must be "shifted" or gives a `negative answer` which means that $(\mathbf{v}_{k+1}; \cdots ; \mathbf{v}_n)u$ cannot be simplified w.r.t. $\mathbf{v}_k$.

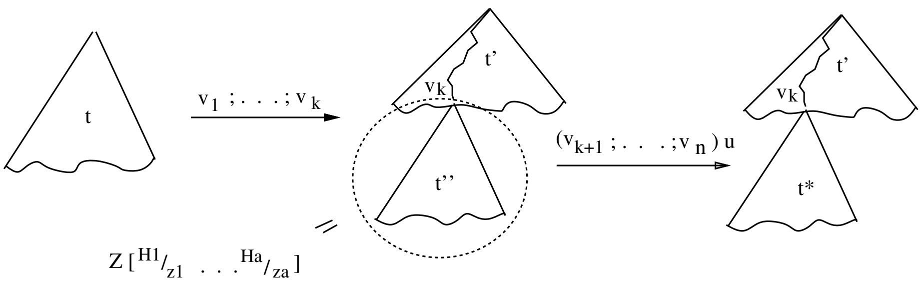
Along the algorithm, we have used three functions:

- `lmo` takes a set of access paths and gives the leftmost outermost;
- `find-metavar` takes a metaexpression H, two natural numbers k and j and an access path u. It is assumed the existence of a derivation $\mathbf{v}_1; \cdots ; \mathbf{v}_n$, $j \le k$ and H is supposed to be a subexpression of the rhs H^j of the rule firing the j-th reduction. `find-metavar` checks whether H has a sub-expression of the shape $Z[^{H_1}/_{z_1}, \cdots, ^{H_a}/_{z_a}]$ such that
  1. there exists $w \in o_H(H^j)$ such that $(\mathbf{v}_{k+1}; \cdots ; \mathbf{v}_n)u$ is internal to $u \cdot w$;
  2. $(\mathbf{v}_{k+1}; \cdots ; \mathbf{v}_n)u$ is not internal to one of the metavariables in the rule firing $\mathbf{v}_j$.

  In such a case `find-metavar` gives the pair $\langle Z, w\rangle$. Otherwise it gives the pair $\langle$`undefined`, `undefined`$\rangle$.
- `check-var` is a function which takes an expression t, a set $\{z_1, \cdots, z_a\}$ of free variables in t, a natural number k and an access path u. It verifies if there exists some occurrence number w in $o_{z_p}(t)$ (for some $1 \le p \le a$) such that and $(\mathbf{v}_{k+1}; \cdots ; \mathbf{v}_n)u$ is internal to $u \cdot w$. In this case `check-var` gives the pair $\langle p, w\rangle$. Otherwise it gives $\langle$`undefined`, `undefined`$\rangle$.

So, assume to have a derivation $(\mathbf{v}_1; \cdots ; \mathbf{v}_n)u$ illustrated below:

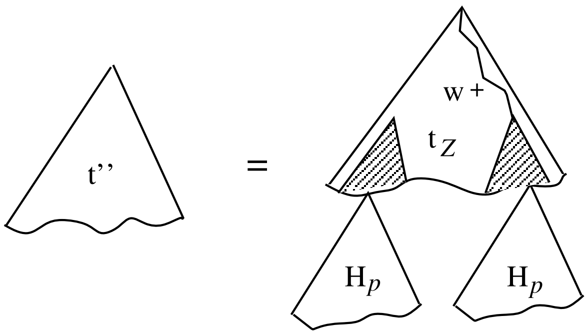
algorithm shift-indexes	
`input`	a standard derivation $(\mathbf{v}_1; \cdots; \mathbf{v}_n)u$ and a shift number k $(1 \le k \le n)$.
`output`	a triple $(w_0,\, w_1,\, w_2)$ or a `negative answer`.
`initialization step`: $j := k$; $w := \varepsilon$. The variable j shall range over reductions $(1 \le j \le k)$ and w is the basis of the shift.	
`step 1`	Take the schema $\mathbf{h}(\mathbf{g}(\vec{y}^{\,1}.\, Y_1, \cdots, \vec{y}^{\,r}.\, Y_r), \cdots, \vec{y}^{\,s}.\, Y_s) \to H^k$ contracting $\mathbf{v}_j$. Let $\langle Z,\, w^+\rangle$ = `find-metavar(`$H^k, k, k, v_k \cdot w$`)`. If w^+ is `undefined` then stop with a `negative answer` otherwise continue with $w := w^+$.
`step 2`	Let $\mathbf{f}$ be the form among $\mathbf{h}$ and $\mathbf{g}$ binding Z along $\mathbf{v}_j$ and let t_Z be the expression replacing Z along $\mathbf{v}_j$. Moreover, let $\{z_1, \cdots, z_a\}$ be the variables bound by $\mathbf{f}$ on t_Z. Let $\langle p,\, w'\rangle$ = `check-var(`$t_Z, \{z_1, \cdots, z_a\}, k, v_k \cdot w$`)`.
`step 3`	There are several cases: $p \neq$ `undefined` and $w' \neq$ `lmo(`$o_{z_p}(t_Z)$`)`: Then stop with the triple $(v_k \cdot w,$ `lmo`$(o_{z_p}(t_Z)),\, w')$. $p \neq$ `undefined` and $w' =$ `lmo`$(o_{z_p}(t_Z))$: Then stop with a `negative answer`. $\mathbf{f}$ is a residual of a form in the initial expression: Then stop with a `negative answer`. `otherwise`: Let $\mathbf{v}_q$ be the reduction creating $\mathbf{f}$ $(q \le k-1)$. Put $j := q$ and let $\mathbf{h}(\mathbf{g}(\vec{y}^{\,1}.\, Y_1, \cdots, \vec{y}^{\,r}.\, Y_r), \cdots, \vec{y}^{\,s}.\, Y_s) \to H^q$ be the schema of v_q. Let H^q_i be the schema of the i-th argument of $\mathbf{f}$ in H^q and $\langle Z,\, w^\bullet\rangle$ = `find-metavar(`$H^q_i, k, j, v_k \cdot w$`)`. If $w^\bullet$ is `undefined` then stop with a `negative answer` else $w := w \cdot w^\bullet$ and go-to `step 2`.



Moreover, for simplicity, assume that the rhs of the schema contracting $\mathbf{v}_k$ has the shape $Z[^{H_1}/_{z_1}, \cdots, ^{H_a}/_{z_a}]$ and that $(\mathbf{v}_{k+1}; \cdots; \mathbf{v}_n)u$ is internal to $Z[^{H_1}/_{z_1}, \cdots, ^{H_a}/_{z_a}]$. In the above picture, t'' is the expression replacing $Z[^{H_1}/_{z_1}, \cdots, ^{H_a}/_{z_a}]$ and it is changed into t^* by the tailing derivation. So, we have successfully terminated the function `find-metavar` before the `step 2` of the algorithm (well, we are also assuming that $(\mathbf{v}_{k+1}; \cdots; \mathbf{v}_n)u$ is not internal to some metavariable of $\mathbf{v}_k$).

Now there are several cases (we are just at the beginning of `step 3`). For instance, it is possible that the tailing derivation is internal to one of the H_p (that is the output of the function `check-var` is defined). In this case, if the instance of the H_p is not the leftmost, we finish successfully providing as output the triple with the leftmost access path of the H_p as second component.

Otherwise, even if $(\mathbf{v}_{k+1}; \cdots; \mathbf{v}_n)u$ is not internal to some H_p, it can be internal to parts of Z that have been duplicated along the previous derivations (we are in the `otherwise` case of `step 3`). The situation is graphically described below:



Notice that the expression t_Z replacing the metavariable Z has two grey parts, that we assume to be duplicated along the derivation $\mathbf{v}_1; \cdots; \mathbf{v}_k$. Now, it may happen that the tailing derivation $(\mathbf{v}_{k+1}; \cdots; \mathbf{v}_n)u$ is actually internal to one of these copies or not. In the latter case we must `stop` with a `negative answer`. In the first case, the question is: how to recognize the presence of such copies?

To this aim take the unique form between the constructor and the destructor of the redex v_k that has Z as argument. Let $\mathbf{f}$ be such form. There are two subcases: the first is that $\mathbf{f}$ is residual of a form in the initial expression. Notice that, in this case, Z could not have subexpressions that have been copied since, by the hypothesis of standardization, reductions in $\mathbf{v}_1; \cdots; \mathbf{v}_{k-1}$ have to be disjoint to the left of $\mathbf{f}$ or external w.r.t. $\mathbf{f}$. In both cases the arguments of $\mathbf{f}$ cannot be affected. The second subcase is that $\mathbf{f}$ has been created along $\mathbf{v}_q$ ($q \leq k-1$). In order to recognize duplications in Z, the idea is to analyze the shape of the rhs of the schema contracting v_q. In particular, the metaexpression H_i^q corresponding to the argument of $\mathbf{f}$ represented by Z.

Now, we pose a question that we have already found: does a subexpression of H_i^q exist whose shape is $Z'[^{H_1'}/_{z_1'}, \cdots, ^{H_b'}/_{z_b'}]$ such that $(\mathbf{v}_{k+1}; \cdots; \mathbf{v}_n)u$ is internal to it? That is we must call `find-metavar` again and reiterate `step 2`. Notice that the new iteration is performed with a different value of the offset w taking into account the access path of H_i^q.

We remark that the algorithm `shift-indexes` is effective because every elementary step used by the algorithm is decidable. Perhaps, the unique step that deserves clearings is the possibility of recognizing in an effective way whether a form is created by a redex or less. A way out consists of keeping trace of forms created by redexes by marking them (with a bullet, for instance). Then $\mathbf{f}$ is created by $\mathbf{v}_j$ in $\mathbf{v}_1; \cdots; \mathbf{v}_n$ if, starting from the initial expression of $\mathbf{v}_j$ and marking forms created by $\mathbf{v}_j$, $\mathbf{f}$ is marked after $\mathbf{v}_j; \cdots; \mathbf{v}_n$. An easy induction over the length of the derivation shows that, for every form in the ending expression of $\mathbf{v}_1; \cdots; \mathbf{v}_n$, there exists *at most* one reduction creating it along $\mathbf{v}_1; \cdots; \mathbf{v}_n$.

Definition 10 *Take a standard derivation $\sigma; \mathbf{u}; \rho$ and assume that ρ is internal to a metaexpression $X[^{H_1}/_{x_1}, \cdots, ^{H_n}/_{x_n}]$ in the rhs of the schema contracting u. Let (w_0, w_1, w_2) be the output of the algorithm* `shift-indexes`. *Then the* shift *of ρ w.r.t. $\mathbf{u}$ (notation $\rho \upharpoonleft_{w_1}^{w_2} w_0$) is the derivation ρ where prefixes $w_0 \cdot w_1$ of redexes are replaced by $w_0 \cdot w_2$. Often we will omit indexes w_1 and w_2.*

Example 9 *(***The operator** μ**, continued***) Let us clarify the above algorithm by means of Examples 7 and 8.*

As far as Example 7 is concerned, let $\mathbf{u}$ be the first reduction starting at the initial expression $\mu(\lambda(x.\, xx))$ and let v' and v'' be the two redexes created in the ending expression of $\mathbf{u}$. Let us show that v'' in $\mathbf{u}v''$ can be shifted w.r.t. $\mathbf{u}$. Indeed, the rhs of the rule contracting the pair μ-λ is $X[^{\mu(\lambda(x.X))}/_x]$ and v'' is internal to the shared part of the substitution (moreover, v'' is not the leftmost occurrence of such part). Thus the algorithm `shift-indexes` *stops at the first step providing the triple $(\varepsilon,\, v'',\, v')$.*

In Example 8 the computation of `shift-indexes` *is a bit longer. Actually, take the derivation $(\mathbf{u}_1; \mathbf{u}_2)v''$. Let us prove that* `shift-indexes` *shifts v'' w.r.t. $\mathbf{u}_2$. Indeed, v'' is internal to the schema of the rhs of $\mathbf{u}_2$ (which is $X[^Z/_x]$). Anyway, v'' is not internal to Z. Therefore we must discover whether X has copies or not. Observe that X is the metavariable of $\mathbf{g}$, that has been created by $\mathbf{u}_1$. Therefore* `shift-indexes` *performs a second step. Now, the schema of the second argument of the form $\mathbf{g}$ created by $\mathbf{u}_1$ is $X[^{Y[^z/_y]}/_x]$ and we can verify that v'' is internal to the shared part of such substitution. Simple calculations yield that the output of* `shift-indexes` *is $(\varepsilon,\, v'',\, v')$.*

6.4 The extraction relation

The generalization of the extraction relation consists of adding the operation performing the shift to the two elementary operations $\rhd_1$ and $\rhd_2$ in Definition 8.

Definition 11 *The* extraction relation $\rhd$ *(warning: from now on* $\rhd$ *shall have a different meaning) be the union of the following three relations:*

1. $\rho; (\mathbf{u} \sqcup \sigma v) \ \rhd_1 (\rho; \sigma)v$ if $\mathbf{u}$ and $\sigma; \mathbf{v}$ are two *disjoint* reductions
2. $(\rho; \mathbf{u}; \sigma)v \ \rhd_2^i (\rho; \sigma \uparrow_w^{w'} u)v'$ if $\sigma; \mathbf{v}$ is internal to the metavariable X_i of $\mathbf{u}$.
3. $(\rho; \mathbf{u}; \sigma)v \ \rhd_3 (\rho; \mathbf{u}; \sigma \Lsh_{w_1}^{w_2} w_0)v'$ if $\sigma; \mathbf{v}$ can be shifted w.r.t. $\mathbf{u}$ and (w_0, w_1, w_2) is the output of `shift-indexes`.

where, in 2, w *and* w' *are the occurrence numbers of* X_i *in the rhs and lhs of the rewriting schema firing* $\mathbf{u}$*, respectively. Moreover* $v' = u \cdot w' \cdot v''$*, provided that* $v = u \cdot w \cdot v''$*. In 3,* $v = w_0 \cdot w_1 \cdot v^+$ *and* $v' = w_0 \cdot w_2 \cdot v^+$*.*

We denote with $\unrhd$ *the reflexive and transitive closure of* $\rhd$*.*

Observe that, due to Example 9, the above relation $\unrhd$ completely solves the problems we had in Examples 7 and 8 in recognizing redexes that are copies. However, till now, we have not guaranteed the well-definition of the operations of lift and shift. In particular the latter one is involved, due to the algorithm for computing indexes of the shift. We will devote the next section to prove properties of $\unrhd$ as well definition, Church-Rosser and strong normalization.

7 Properties of the extraction process

Proposition 1 *The operation of lift is well defined.*

Proof: Under the hypotheses of Definition 7, let $\rho = \mathbf{v}_1; \cdots; \mathbf{v}_n$ where $v_i = u \cdot w^+ \cdot v_i'$. We have to show the existence of the reduction $\rho' = \mathbf{v}_1'; \cdots; \mathbf{v}_n'$ starting from t_i, the expression matched by $\mathbf{u}$ with the metavariable X_i. An easy induction on the structure of the rhs of the rule firing u gives that, at each step of ρ, let us say j, the expression whose access path is $w = u \cdot w^+$ has the form $t_i^{(j)}[{}^{t_1'}/_{x_1}, \cdots, {}^{t_n'}/_{x_n}]$, where t_i' are always fixed along ρ. This implies the existence of ρ' and the well-definition of the operation of lift. ▯

In the proof of Proposition 2 we will exploit a notational device, in order to emphasize the presence of parts of an expression that have been copied along a derivation. For instance, take $t \xrightarrow{u} t'$ and assume that the rhs of the schema contracting u has a subexpression $X[{}^{H}/_{x}]$. Therefore, in t' there will be copies of the expression replacing H' according to the occurrences of x in the expression replacing X. In order to show up such situation we will write the expression instantiating $X[^{H}/_{x}]$ as

$$(X[^{H}/_{x}])_u$$

where the index u means that expressions replacing metavariables are those used along $\mathbf{u}$. In the above notation, if t_X, the expression replacing X, has two copies of x and t_H is the expression replacing H, we will find two copies of t_H in t' (the occurrences can be determinated through standard machineries). Notice that, we have already used a similar notation in Example 8, to remark the presence of copies in the expression M'.

Proposition 2 *The operation of shift is well-defined. That is,* `shift-indexes` *always terminates and if its output is* $(v \cdot w,\ w',\ w'')$ *then* $(\sigma_1; \mathbf{v}; \sigma_2)u$ *can be simplified into* $\sigma_1; \mathbf{v}; (\sigma_2 u) \Lsh^{w''}_{w'} v \cdot w$.

Proof: Termination is due to the fact that each time the body of `shift-indexes` is performed, the index of the reduction under examination strictly decreases.

Let $t \xrightarrow{\sigma_1}\!\!\!\!\rightarrow t' \xrightarrow{v} t_v \xrightarrow{\sigma_2 u}\!\!\!\!\rightarrow$, $\sigma_1 = \mathbf{v}_1; \cdots; \mathbf{v}_k$ and $\sigma_2 u$ be internal to a metaexpression $Z_1[^{H_1^1}/_{z_1^1}, \cdots, ^{H_q^1}/_{z_q^1}]$ of the rhs of the rule contracting v. We verify that the subexpressions denoted by $v \cdot w \cdot w'$ and $v \cdot w \cdot w''$ are equal. By induction over the number n of steps performed by the algorithm `shift-indexes`, we show that the schema of a subexpression of Z_1 addressed by $v \cdot w$ and such that σ_2 is internal to it is

$$(\cdots(Z_n[^{H_1^n}/_{z_1^n}, \cdots, ^{H_p^n}/_{z_p^n}])_{j_n} \cdots [^{H_1^1}/_{z_1^1}, \cdots, ^{H_q^1}/_{z_q^1}])_{j_1} \tag{2}$$

where $(-)_j$ means that the replacement of expressions for metavariables must be done in the initial expression of the reduction $\mathbf{v}_j$. The basic case is easy. Let us see the inductive step.

By inductive hypothesis we assume that the subexpression of the ending term of $\mathbf{v}$ and recognized by $v \cdot w$ is (2) and that σ_2 is internal to it. Let $\mathbf{v}_{j_n}$ be the reduction under examination at step n. According to the algorithm `shift-indexes` there are two cases. The first is when there exists a k such that $\sigma_2 u$ is internal to H_k^n at occurrence w' (w' is not the leftmost occurrence of H_k^n). The possibility to shift σ_2 towards the leftmost occurrence of H_p^n is an obvious consequence of the hypothesis. The algorithm terminates here, thus the proposition follows by induction.

The second case is when the derivation $\sigma_2 u$ contracts redexes in Z_n, too. Here we have to check if Z_n has still subparts which are copies. That is, let $\mathbf{f}$ be the unique form between the constructor and the destructor whose argument is represented by Z_n in the schema contracting v_{j_n}. The non trivial case is when there exists a reduction $\mathbf{v}_i$ ($i < j_n$) creating $\mathbf{f}$. Observe that reductions in $\mathbf{v}_{i+1}; \cdots; \mathbf{v}_{j_n - 1}$ do not fire redexes internal to some argument of $\mathbf{f}$. Indeed, if $\mathbf{f}$ is a constructor then, by the hypothesis of standardization, it is not possible to fire a redex internal at $\mathbf{f}$ because $\mathbf{f}$ is fired by $\mathbf{v}_{j_n}$ only. If $\mathbf{f}$ is a destructor then the situation is the same because metavariables mentioned in the rewriting schema do not refer to the first argument of the destructor. Therefore the expression replacing Z_n remains unchanged along $\mathbf{v}_{i+1}; \cdots; \mathbf{v}_{j_n-1}$. Hence we can recognize the presence of copies in Z_n by looking at the schema of the rule creating $\mathbf{f}$, i.e. $\mathbf{v}_i$.

Now, assume that the schema of the rule contracting $\mathbf{v}_i$ has a metaexpression of the form $Z_{n+1}[^{H_1}/_{z_1}, \cdots, ^{H_r}/_{z_r}]$ in the argument of $\mathbf{f}$ determinated by Z_n (let it be the h-th). Let w^+ be an occurrence number of the above subexpression (relative to the beginning of the h-th argument of $\mathbf{f}$) such that $\sigma_2 u$ is internal to $v \cdot w \cdot w^+$. Observe that, by definition of substitution, subexpressions recognized by different instances of H_k (for every k) are equal. This yields the proposition. ∎

The next step consists of showing that the extraction process always terminates with a unique normal form. To this aim we need to prove the property of Church-Rosser of $\trianglerighteq$ and its strong normalization.

Lemma 1 *The relation $\trianglerighteq$ has the Church-Rosser property, i.e. if $\rho \vartriangleright \sigma$ and $\rho \vartriangleright \tau$ then there exists a ν such that $\sigma \vartriangleright \nu$ and $\tau \vartriangleright \nu$ (ρ is assumed standard).*

Proof: By cases (m, n), according to the rule applied for $\sigma \vartriangleright_m \nu$ and $\tau \vartriangleright_n \nu$. The proof is standard: the details can be found in [Laneve 1993] (pp. 124–126). ∎

Proposition 3 *$\trianglerighteq$ is strong normalizing.*

Proof: Let $\rho = \mathbf{v}_1; \cdots; \mathbf{v}_n$ be a standard derivation. Notice that, according to Lemma 1, a reduction $\rho \vartriangleright_i \sigma$ do not delete or duplicate a different reduction $\rho \vartriangleright_j \tau$. Furthermore, $\vartriangleright_1$ and $\vartriangleright_2$ always shorten the derivation ρ. Therefore there are at most n reductions of type "1" or "2".

Now take a reduction $\mathbf{v}_i$ and let κ_i be the number of subexpressions of the form $X[^{H_1}/_{x_1}, \cdots, ^{H_m}/_{x_m}]$ in the rhs of the schema contracting $\mathbf{v}_i$. Then there are at most $\sum_{i=1}^{n} \kappa_i$ redexes of type "3" in ρ. Hence the strong normalization. ∎

Proposition 4 *The simplifications $\trianglerighteq_i$ are effective.*

Proof: The simplifications $\vartriangleright_1$ and $\vartriangleright_2$ are trivially effective. The effectivity of $\vartriangleright_3$ relies upon that of the algorithm `shift-indexes`. Verify that every elementary step of such algorithm is effective. ∎

7.1 The canonical derivation

The properties proved in this section guarantee that every σu can be simplified w.r.t. $\trianglerighteq$ till a normal form. Therefore, for any σu, it is possible to find a canonical representative. Unfortunately it is possible that $\rho \equiv \sigma$ and ρu does not reduce to the same representative of σu. This because two independent causes of u can appear along ρ and σ in different order and the extraction process does not succeed to change the order. This problem disappears if we restrict to standard derivations.

Definition 12 *Every derivation σu which is standard and in normal form w.r.t. $\trianglerighteq$ will be called* canonical.

Canonical derivations are the basic notion for the definition of family based on the extraction relation.

Definition 13 *A redex u with history ρ (notation ρu) is in the same* extraction relation *of a redex v with history σ, written $\rho u \simeq \sigma u$, when there exists a τw such that $\rho_1 u \trianglerighteq \tau w \trianglelefteq \sigma_1 v$, where $\rho_1; \rho_2; \rho_3$ and $\sigma_1; \sigma_2; \sigma_3$ are respectively the standard derivations of ρ and σ and ρ_2 is a derivation internal to u and ρ_3 is disjoint to the right of u (similarly for σ_2 and σ_3 w.r.t. v).*

Notice that, at the moment we have two different relations on redexes with histories: the family relation of Definition 4 and the extraction relation of Definition 13. In the next section we shall prove their coincidence.

8 The equivalence of extraction and labeling

This section is devoted to prove that two redexes have the same degree if and only if their canonical derivations are the same. Lévy proved this result in the case of λ-calculus ([Lévy 1980], pp. 68–113) by using a complex notion of labeled subcontext. In [Asperti and Laneve 1992] (pp. 30–37) we have simplified Lévy's proof by exploiting the *path*-characterization of λ-calculus labels. The proof strategy is the same here, but proofs are more difficult because of two phenomena that are absent in the λ-calculus. Let $\mathtt{INIT}(t)$ and $t \xrightarrow{u} t'$ and $a = \partial_t\ (u)$:

1. in the case of λ-calculus, labels in t' having a as sublabel identify uniquely the edge. This is false in general. For instance, take the expression $t = \mu(\lambda(x.\,@(x,\,x)))$ of Example 7. The reader can check that, after a labeling, the two redexes (i.e. different edges) created by the first reduction $\mathbf{u}$ have the same label (which, in turn, contains a).

2. it is possible that a form internal w.r.t. u has edges incoming into the bound port that are equal (this is again not possible in the λ-calculus). As an example take λ-calculus plus a new destructor $\mathbf{d}$ which interacts with λ through the rule:

$$\mathbf{d}(\lambda(x.\,X)) \quad \rightarrow \quad \lambda(y.\,X[^{\mathbf{d}(y)}/_x])$$

   and consider the expression $\mathbf{d}(\lambda(x.\,@(x,\,x)))$. After firing the redex in an isomorphic labeled expression, the abstraction in the ending expression will have two edges incoming into the bound port that are labeled in the same way.

The easy part of the equivalence extraction-labeling is that two redexes with histories that are in the same family have the same degree. This because the extraction process does not change the degree of the ending redex.

Theorem 5 *Let* $t \stackrel{\rho}{\twoheadrightarrow} t^\rho \xrightarrow{u}$ *and* $t \stackrel{\sigma}{\twoheadrightarrow} t^\sigma \xrightarrow{v}$. *Then* $\rho u \simeq \sigma v$ *implies* $\partial_t^\rho(u) = \partial_t^\sigma(v)$.

Proof: If $\rho u \simeq \sigma v$ then $\rho_1 u \trianglerighteq \tau w \trianglelefteq \sigma_1 v$, where $\rho_1; \rho_2; \rho_3$ and $\sigma_1; \sigma_2; \sigma_3$ are the standardizations of ρ and σ, respectively, and ρ_2 is internal to u and ρ_3 is disjoint to the right w.r.t. u (similarly for σ_2 and σ_3 w.r.t. v). By Theorem 2, standardizing ρ and σ does not change the ending terms t^ρ and t^σ (even the ending terms of the isomorphic labeled derivation starting at t_I are unchanged). Thus, after the process of standardization, $\partial_t^\rho(u)$ and $\partial_t^\sigma(v)$ are unchanged. Similarly after the erasing of reductions that are internal or disjoint to the right.

Let $t_I \stackrel{\rho_1}{\twoheadrightarrow} t_I^A \xrightarrow{u}$ and $t_I \stackrel{\sigma_1}{\twoheadrightarrow} t_I^B \xrightarrow{v}$. Then $\partial_t^{\rho_1}(u) = \partial_t^\rho(u)$ and $\partial_t^{\sigma_1}(v) = \partial_t^\sigma(v)$. The theorem is verified if every elementary step of the extraction process does not change the degree of the rightmost redex. By cases over the type of the elementary step.

$\rho; (\mathbf{u} \sqcup \sigma v) \blacktriangleright_1 (\rho; \sigma)v$ We are removing a reduction $\mathbf{u}$ that is disjoint (to the left) w.r.t. σv. Notice that σv by no means is affected by $\mathbf{u}$ because they work on disjoint subexpressions. This remark implies the statement of the theorem when the isomorphic labeled reduction is taken.

$(\rho; \mathbf{u}; \sigma)v \blacktriangleright_2 (\rho; \sigma(\uparrow_z^{z'} u)v'$ (where $u \cdot z$ is the occurrence number of the metavariable X to which σv is internal, z' is the access path of X in the lhs of the schema contracting u and $v' = u \cdot z'$) By definition of the (labeled) rewriting rule, $\mathbf{u}$ can not affect expressions replacing metavariables because they are taken as an atomic unit. Now, by Proposition 1, we can lift the derivation σv w.r.t. $\mathbf{u}$ and, of course, this is also possible over the labeled derivation. Since redexes along σv are internal to X, their label can not change as a consequence of lifting.

$(\rho; \mathbf{u}; \sigma)v \blacktriangleright_3 (\rho; \mathbf{u}; \sigma \Lsh_{w_1}^{w_2} w_0)v'$ (where $v = w_0 \cdot w_1 \cdot v^+$ and $v' = w_0 \cdot w_2 \cdot v^+$) This is an immediate consequence of a slight rephrasing of Proposition 2 to labeled derivations.

$\square$

The converse, which holds when the initial expression owns `INIT` relies upon two main lemmas, which will be proved in Appendix B and C, respectively.

Lemma 2 (The Contracted Label Lemma) *Let* $t \xrightarrow{w} t' \stackrel{\rho}{\twoheadrightarrow} t^\rho \xrightarrow{u}$ *and* $t \xrightarrow{w} t' \stackrel{\sigma}{\twoheadrightarrow} t^\sigma \xrightarrow{v}$ *be canonical. If* $\mathtt{str}(\partial_{t'}^\rho(u)) = \mathtt{str}(\partial_{t'}^\sigma(v))$ *then*

$$\partial_t^{w;\rho}(u) = \partial_t^{w;\sigma}(v) \quad \Rightarrow \quad \partial_{t'}^\rho(u) = \partial_{t'}^\sigma(v).$$

The proof of this lemma exploit a property of degrees stating that the label fired by the first reduction in the canonical derivation occurs in the degree. Moreover, under the hypothesis of the lemma, we know that the labels $\partial_{t'}^\rho(u)$ and $\partial_{t'}^\sigma(v)$ can be obtained from $\partial_t^{w;\rho}(u)$ and $\partial_t^{w;\sigma}(v)$ through a substitution $\mathtt{s}$ mapping an atomic label in t'_I into that of the corresponding edge in $t^\star$, where $t_I \xrightarrow{u} t^\star$. Let $a = \partial_t\,(w)$. Hence $a \in (\partial_{t'}^\rho(u))[\mathtt{s}]$ and $a \in (\partial_{t'}^\sigma(v))[\mathtt{s}]$. The crucial point is to show that there exists a label b, with $a \in b[\mathtt{s}]$, occurring in the same position both in $\partial_{t'}^\rho(u)$ and $\partial_{t'}^\sigma(v)$. Then, since $\mathtt{str}(\partial_{t'}^\rho(u)) = \mathtt{str}(\partial_{t'}^\sigma(v))$, the Connection Property implies $\partial_{t'}^\rho(u) = \partial_{t'}^\sigma(v)$.

Lemma 3 (The Contracted Structure Lemma) *Let* $t \xrightarrow{w} t' \xrightarrow{\rho}\!\!\!\!\rightarrow t^{\rho} \xrightarrow{u}$ *and* $t \xrightarrow{w} t' \xrightarrow{\sigma}\!\!\!\!\rightarrow t^{\sigma} \xrightarrow{v}$ *be canonical. Then*

$$\mathtt{str}(\partial^{\rho}_{t'}(u)) \neq \mathtt{str}(\partial^{\sigma}_{t'}(v)) \quad \Rightarrow \quad \partial^{w;\rho}_{t}(u) \neq \partial^{w;\sigma}_{t}(v).$$

We remark that this case is easily proved for the λ-calculus. Let us see why. As said before, $\partial^{\rho}_{t'}(u)$ and $\partial^{\sigma}_{t'}(v)$ can be obtained from $\partial^{w;\rho}_{t}(u)$ and $\partial^{w;\sigma}_{t}(v)$ through a substitution $\mathtt{s}$ mapping an atomic label in t'_I into that of the corresponding edge in $t^{\star}$, where $t_I \xrightarrow{u} t^{\star}$. If the structures of $\partial^{\rho}_{t'}(u)$ and $\partial^{\sigma}_{t'}(v)$ differ for underlinings or overlinings (or, in general, for operations "$(-)_s$") then the lemma is easily true (also in the general case). When $\mathtt{str}(\partial^{\rho}_{t'}(u))$ and $\mathtt{str}(\partial^{\sigma}_{t'}(v))$ differ for "bound bullets", the situation is similar, because β-reduction can not equate the labels of two bound edges occurring in $\partial^{\rho}_{t'}(u)$ or $\partial^{\sigma}_{t'}(v)$.

The last property does not hold in general. This is the case when, for instance, the rhs of the rule firing $\mathbf{w}$ has a subexpression of the shape $\mathbf{f}(z.\,X[^{\mathrm{g}(z)}/_{x}])$ and the bound variable is z. Observe that, contracting w will create as many occurrences of z as the occurrences of x in the subexpression replacing X. Moreover any such occurrence has, by definition, the same label in t'. On the other hand, since t'_I holds `INIT`, those occurrences will be labeled in a different way.

In this case, it happens that at least one of $(\mathbf{w};\rho)u$ and $(\mathbf{w};\sigma)v$ is not canonical. Actually, if $\partial^{\rho}_{t'}(u) = \partial^{\sigma}_{t'}(v)$, by the Connection Property, both ρu and σv must be eventually internal to the substitution $X[^{\mathrm{g}(z)}/_{x}]$ (in particular to $\mathbf{g}(z)$). Thus, since one of the atomic labels in $\partial^{\rho}_{t'}(u)$ and $\partial^{\sigma}_{t'}(v)$ mark an occurrence of the bound variable to the left of the other, it results that ρu or σv can be furtherly simplified w.r.t. $\mathbf{w}$. Which contradicts the hypothesis.

Theorem 6 *Let* $t \xrightarrow{\rho}\!\!\!\!\rightarrow t' \xrightarrow{u}$ *and* $t \xrightarrow{\sigma}\!\!\!\!\rightarrow t'' \xrightarrow{v}$*. Then*

$$\partial^{\rho}_{t}(u) = \partial^{\sigma}_{t}(v) \quad \Rightarrow \quad \rho u \simeq \sigma v\ .$$

Proof: Theorem 5 shows that we can safely reduce to consider the case when ρu and σv are canonical. Thus we have to prove that:

$$\rho u \ \neq \ \sigma v \quad \Rightarrow \quad \partial^{\rho}_{t}(u) \ \neq \ \partial^{\sigma}_{t}(v)$$

By induction over the minimum of the lengths of ρ and σ. Assume that it is always ρ.

($\rho = \underline{\emptyset}$) Here u has to be a redex in the initial term t (thus its label is atomic in t_I). There are two subcases: $\sigma = \underline{\emptyset}$ and $\sigma = \mathbf{v}';\sigma'$. The first subcase is immediate because $u \neq v$ implies that their degree is different. In the second case $\partial^{\sigma}_{t}(v)$ can not be atomic. If, by contradiction it was, then $\partial^{\sigma}_{t}(v)$ individuates a redex w in t_I such that $v \in w/\sigma$. Hence it is possible to show that σv is not in normal form w.r.t. $\trianglerighteq$.

($\rho = \mathbf{w};\rho'$) There are two subcases: $\sigma = \mathbf{w}';\sigma'$ $(w \neq w')$ and $\sigma = \mathbf{w};\sigma'$. In the first case, one between w and w' will be external to the other or they are disjoint. In every case the leftmost outermost in $\mathtt{at}(\partial^{w;\rho}_{t}(u))$ and $\mathtt{at}(\partial^{w';\sigma}_{t}(v))$ are different, thus, by Proposition 6, $\partial^{w;\rho}_{t}(u) \neq \partial^{w';\sigma}_{t}(v)$.

For the latter case, observe that, the hypothesis $\rho u \neq \sigma v$ implies that $\rho' u \neq \sigma' v$. Now let $t^\star$ be such that $t \xrightarrow{w} t'$ and $t_I \xrightarrow{w} t^\star$ are isomorphic reductions. Notice that we can not apply the induction because $\mathtt{INIT}(t^\star)$ is false. However, by induction, $\partial^{\rho}_{t'}(u) \neq \partial^{\sigma}_{t'}(v)$. Thus, in order to prove the theorem, we have to show that $\partial^{\rho}_{t'}(u) \neq \partial^{\sigma}_{t'}(v) \;\Rightarrow\; \partial^{w;\rho}_{t}(u) \neq \partial^{w;\sigma}_{t}(v)$. There are two subcases, according to $\mathtt{str}(\partial^{\rho}_{t'}(u)) = \mathtt{str}(\partial^{\sigma}_{t'}(v))$ or not. In the first case, the equivalent implication

$$\partial^{w;\rho}_{t}(u) = \partial^{w;\sigma}_{t}(v) \;\;\Rightarrow\;\; \partial^{\rho}_{t'}(u) = \partial^{\sigma}_{t'}(v)$$

follows by Lemma 2. When $\mathtt{str}(\partial^{\rho}_{t'}(u)) \neq \mathtt{str}(\partial^{\sigma}_{t'}(v))$ then use Lemma 3 to yield the statement of the theorem.

$\square$

9 Other approaches to the family relation

(**The zig-zag relation**) The most abstract approach to the notion of family is the so called *zig-zag* [Lévy 1978]. In this case, duplication of redexes is formalized as residuals modulo permutations. In particular, a redex u with history σ is a *copy* of a redex v with history ρ iff $\rho v \leq \sigma u$, i.e., there exists τ such that $\sigma \equiv \rho;\tau$ and $u \in v/\tau$. The family relation is then the symmetric and transitive closure of the copy-relation (pictorially, this gives rise to the "zig-zag").

The example below shows the inadequacy of the zig-zag relation.

Example 10 *(***The operator** μ*). Take λ-calculus extended with a recursion operator μ as defined in Example 5 (for simplicity, we shall write the application in the usual infix form). Take the following term:*

$$\underbrace{\mu(\lambda(x.\,(\overbrace{\lambda(y.\,\mathbf{I})\,x}^{u})\,z))}_{v}$$

where the two redexes are explicitly named by the under/over-braces.

$$\begin{tikzcd}
 & \mu(\lambda(x.\,(\lambda(y.\,\mathbf{I})\,x)\,z)) \arrow[dl, "v"'] \arrow[dr, "u"] & \\
\overbrace{(\lambda(y.\,\mathbf{I}))(\mu(\lambda(x.\,(\underbrace{\lambda(y.\,\mathbf{I})\,x}_{u'})\,z)))}^{u''}\;z \arrow[dr, "u'"] & & \mu(\lambda(x.\,\underbrace{\mathbf{I}\,z}_{w_1})) \\
 & (\lambda(y.\,\mathbf{I}))(\mu(\lambda(x.\,\underbrace{\mathbf{I}\,z}_{w_2})))\;z &
\end{tikzcd}$$

Now, by using Lévy's notion of zig-zag, there is no way to relate the two redexes w_1 and w_2 (which are intuitively *in the same family). Indeed if we attempted to complete the derivation $\mathbf{v}; \mathbf{u'}$ by firing the redex u'', the redex w_2 is canceled. The interesting fact is that this phenomenon does not happen with the operator $\mathbf{Y}$ and the rule $\mathbf{Y}f \longrightarrow f(\mathbf{Y}f)$. This because now the above reduction v is split in simpler moves and the new intermediate states allows us to make the right 'zig-zag'.*

Following this idea, you may derive plenty of counterexamples to the zig-zag notion of family in Interaction Systems. It is enough to add to the λ-calculus new "combinators" performing several steps of β-reduction at a time.

The above counterexample relies on the fact that the mechanism of completion, at the basis of Lévy's zig-zag, does not work in presence of reduction yielding copies of redexes which are one inside the other. Observe that the same mechanism underlies Lévy's operation of "$\|$", that provides an elementary step of his extraction process (see Remark 5). This is another reason for the failure of Lévy's extraction process in general.

(**Paths**) We have defined a new approach to families in the λ-calculus by means of suitable paths in the initial term of the derivation [Asperti and Laneve 1993b]. According to this characterization, redexes in a same family are created by "contraction" (via β-reduction) of a unique common path in the initial term. This fact provides new evidence about the "common nature" of redexes in a same family, and about the possibility of sharing their reduction. In particular, two redexes are in a same family if and only if they have been originated by contracting a unique common path. Moreover, sharing the redexes means to share the contraction of their common paths.

Unfortunately there is no mean to generalize this approach, since it is based on the property that β-reductions creates just two types of connections. Hence it is possible to equip each type with one direction of traveling (and this gives paths). But, in general, IS-rules create much more than two connections, thus making inadequate the concept of direction.

10 Conclusions

We have introduced a class of higher order rewriting systems, called Interaction Systems and undertaken the theory of optimality. Up to our knowledge, this work is the first significative progress about the theory of optimal reductions (we mean the generalization of the notion of family) in higher order term rewriting systems since Lévy's PhD-thesis in 1978 [Lévy 1978]. The purpose of our research was to provide optimal implementations in terms of Interaction Nets [Lafont 1990] for a class of (functional) languages subsuming λ-calculus. To this aim, this paper provides the theoretical background for any optimal implementation. The paradigmatic description of optimal evaluators *á la* Lamping can be found in [Asperti and Laneve 1993a].

Much work must be done about the theory of optimal reductions in functional languages.

As far as the family relation is concerned, it should be possible to define it for orthogonal Combinatory Reduction Systems. In this case the approach based on labels seems problematic since labels should keep much more structure (they should be labeled trees). In particular, we would eventually face the same problems of orthogonal term rewriting systems in [Maranget 1991]. On the other hand, the extraction process seems much more robust: so much to support the following

CONJECTURE: The extraction process still works in the case of orthogonal Combinatory Reduction Systems, providing the correct family relation.

There is another problem concerning optimal reductions, beside sharing. A derivation, in order to be optimal, must always evaluate (families of) redexes that are *needed*, i.e. that any other derivation should eventually perform. This is the case of the leftmost-outermost strategy in λ-calculus or Interaction Systems or even Left Normal CRS's [Klop 1980]. But the existence of a call-by-need strategy is not guaranteed in the class of CRS's, since it is possible to define functions as the *parallel-or*. In this view, the class of *sequential* CRS's is a reasonable candidate for studying optimal reductions.

Acknowledgements

We are very greatful to Jean Jacques Lévy and Georges Gonthier for the interesting discussions and the relevant suggestions.

References

[Aczel 1978] P. Aczel (1978) A general Church-Rosser theorem. Draft, Manchester.

[Asperti and Laneve 1992] A. Asperti and C. Laneve (1992) Interaction Systems I: the theory of optimal reductions. Technical Report 1748, INRIA-Rocquencourt.

[Asperti and Laneve 1993a] A. Asperti and C. Laneve (1993a) Optimal Reductions in Interaction Systems. In *TapSoft '93*, volume 668 of *Lecture Notes in Computer Science*, pages 485 – 500. Springer-Verlag.

[Asperti and Laneve 1993b] A. Asperti and C. Laneve (1993b) Paths, Computations and Labels in the λ-calculus. In *RTA '93*, volume 690 of *Lecture Notes in Computer Science*, pages 152 – 167. Springer-Verlag.

[Asperti and Laneve 1993c] A. Asperti and C. Laneve (1993c) Interaction Systems II: the practice of optimal reductions. Technical Report UBLCS-93-12, Laboratory for Computer Science, Universitá di Bologna, May 1993. Also Research Report n. 2001, INRIA-Sophia Antipolis. Also available via anonymous `ftp` on `ftp.cs.unibo.it` in the directory `/pub/TR/UBLCS` under the name `93-12.ps.Z`.

[Asperti and Laneve 1994] A. Asperti and C. Laneve (1994) Interaction Systems. in Proc. of Higher Order Algebras, Logic and Term Rewriting, 23–24 September 1993, Amsterdam. To appear in *Lecture Notes in Computer Science*.

[Danos 1990] V. Danos (1990) *La Logique Linéare appliquée à l'étude de divers processus de normalisation (principalement du λ-calcul)*. PhD thesis, Université Paris VII.

[Field 1990] J. Field (1990) On laziness and optimality in lambda interpreters: tools for specification and analysis. In *Proceedings* 17^{th} *ACM Symposium on Principles of Programmining Languages*, pages 1 – 15.

[Girard 1986] J. Y. Girard (1986) Linear Logic. *Theoretical Computer Science*, 50.

[Girard *et al.* 1989] J. Y. Girard, P. Taylor, and Y. Lafont (1989) *Proofs and Types*. Cambridge University Press.

[Gonthier *et al.* 1992a] G. Gonthier, M. Abadi, and J.J. Lévy (1992a) The geometry of optimal lambda reduction. In *Proceedings* 19^{th} *ACM Symposium on Principles of Programmining Languages*, pages 15 – 26.

[Gonthier *et al.* 1992b] G. Gonthier, M. Abadi, and J.J. Lévy (1992b) Linear logic without boxes. In *Proceedings* 7^{th} *Annual Symposium on Logic in Computer Science*.

[Kathail 1990] V. Kathail (1990) *Optimal Interpreters for lambda-calculus based functional languages*. PhD thesis, MIT.

[Klop 1980] J. W. Klop (1980) *Combinatory Reduction System*. PhD thesis, Mathematisch Centrum, Amsterdam.

[Lafont 1990] Y. Lafont (1990) Interaction Nets. In *Proceedings* 17^{th} *ACM Symposium on Principles of Programmining Languages*, pages 95 – 108.

[Lamping 1990] J. Lamping (1990) An algorithm for optimal lambda calculus reductions. In *Proceedings* 17^{th} *ACM Symposium on Principles of Programmining Languages*, pages 16 – 30.

[Laneve 1993] C. Laneve (1993) *Optimality and Concurrency in Interaction Systems*. PhD thesis, Dip. Informatica, Università di Pisa. Technical Report TD – 8/93.

[Lévy 1978] J.J. Lévy (1978) *Réductions correctes et optimales dans le lambda calcul*. PhD thesis, Université Paris VII.

[Lévy 1980] J.J. Lévy (1980) Optimal reductions in the lambda-calculus. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry, Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 159 – 191. Academic Press.

[Malacaria and Regnier 1991] P. Malacaria and L. Regnier (1991) Some results on the interpretation of λ-calculus in operator algebras. In *Proceedings* 6^{th} *Annual Symposium on Logic in Computer Science,* Amsterdam, pages 63 – 72.

[Maranget 1991] L. Maranget (1991) Optimal Derivations in Weak lambda-calculi and in Orthogonal Terms Rewriting Systems. In *Proceedings* 17^{th} *ACM Symposium on Principles of Programmining Languages*, pages 255 – 269.

[Regnier 1991] L. Regnier (1992) *Lambda Calcul et Réseaux.* PhD thesis, Université Paris VII.

A The Connection property

In order to prove the Connection Property we must firstly formalize the notion of connection of a label. Connections will be ports of specified instances of forms. In particular, every proper label α will have two connections, respectively denoted: α^- and α^+. Now, two cases are possible.

The first case is when α is atomic. Then it uniquely defines an edge in the original term t (we assume $\mathtt{INIT}(t)$), and the two connections α^- and α^+ are the two connections of this edge.

The other case, is when the proper label α has been created by the firing of some rule of the labeled system. In this case, we have the following definition

Definition 14 *(***Connections of a label***) Let r be a rewriting rule of an IS, whose (labeled) lhs is*

$$\mathbf{d}(\alpha_1(\cdots(\alpha_k(\mathbf{c}(\vec{x}^{\,1}.X_1,\cdots,\vec{x}^{\,n}.X_n)\cdots),\cdots,\vec{x}^{\,m}.X_m)$$

be its lhs. Let $\ell=\alpha_1\cdots\alpha_k$, the connections *$\beta^+$ and β^- of the proper label $\beta=(\ell)_s$ w.r.t. an instance of the rule r^L are defined by cases according to the position of β in the rhs of r^L:*

β^- : *There are three subcases:*

1. *in the situation $\beta(\mathbf{f}(\ldots))$, β^- is* connected to *the positive port of the output port of the form* $\mathbf{f}$*;*

2. *in the situation $\beta(x)$, where x is a bound variable, β^- is* connected to *the suitable bound port of the binder of x (note that according to the restrictions on IS-rules, this binder is eventually in the rhs of the instance of r^L);*

3. *in the situation $\beta(X_i[\ldots])$, where X_i is the i-th metavariable of* $\mathbf{c}$ *(*$\mathbf{d}$*), β^- is* connected through *the unique negative port of the i-th input partition of* $\mathbf{c}$ *(resp.* $\mathbf{d}$*). That is, the connection "passes through" the form* $\mathbf{c}$ *or* $\mathbf{d}$ *containing X (forms* $\mathbf{c}$ *and* $\mathbf{d}$ *are erased along the reduction). This is the reason of the apparent type mismatching: at the port expressing the connection, β^- will actually meet the positive port of the label contiguous to this port. Such a label β will be called a (negative)* border *label, since it (negatively) traverses the* border *between the rhs of the rule and a metavariables.*

β^+ : *Again, we shall distinguish several subcases:*

1. $(\ell)_0^+$ *is* connected through *the output port of the destructor* $\mathbf{d}$. *As in the third case above, the connection of the label traverses the destructor and will meet the label connected to the output port of* $\mathbf{d}$. *This label opens the subterm relative to the rhs of* r^L; *it is a (positive)* border *label.*

2. *In the situation* $\mathbf{f}(\ldots, \vec{x}^{\,i}.\,\beta(H_i), \ldots)$, β^+ *is* connected to *the negative port of the i-th input partition of the form* $\mathbf{f}$.

3. *In the situation* $X_i[\ldots, {}^{\beta(H)}/_{x}, \ldots]$, *where* X_i *is the i-th metavariable of* $\mathbf{c}$ *(resp.* $\mathbf{d}$*),* β^+ *is* connected through *the positive bound port of the i-th input partition of* $\mathbf{c}$ *(resp.* $\mathbf{d}$*) individuated by* x. *In this case, the "control flow" comes back from the metavariable* X_i *to the rhs of the rule. Again,* β *is a (positive)* border *label.*

Note that the previous definition is completely "local", i.e. it only concerns ports mentioned by the specified rule r^L, and no ports inside the metavariables. Note also that a proper label may be at the same time a positive and negative border line; for instance this is always the case for not atomic proper labels in the λ-calculus.

Remark 6 *Notice that, assuming* `INIT`, *when a positive (negative) border label* β *appears inside a label generated along a derivation over some edge, we shall always find another label at its left (right). For this reason, instead of talking of positive (negative) border labels, we shall say sometimes, with some abuse of terminology, that a label is positively (negatively)* connected to another label. *Conversely, we will say that a proper label is positively (negatively)* connected to a form *when it is not a positive (negative) border label.*

Definition 15 *(***Consistent joins***) Let* α *and* β *be two proper labels. The join* $\alpha\beta$ *is* consistent *if* $\alpha^- = \beta^+$.

Proposition 5 *Rewriting rules make consistent joins.*

Proof: Obvious consequence of the definition of connection and the format of rewriting rules. ∎

Now we need a few properties concerning forms, and their "contiguous" labels. The idea is that proper labels contiguous to ports of a given form are never removed by reductions. The case of *bound* ports, requires some care, since the number of links at these ports can change during the derivation, according to the number of bound variables.

Definition 16 *Let* $\mathbf{f}$ *be a form in a term reached by a derivation* σ *from a term* t *such that* `INIT`(t) *holds.*

(Contiguous labels) *A label* α *is* contiguous *to* $\mathbf{f}$, *along an edge exiting from a given port, if it is the* proper *label closest to* $\mathbf{f}$ *along that edge.*

(Interfaces) *The* interface *of* $\mathbf{f}$ *is the set of proper labels contiguous to* not bound *ports of* $\mathbf{f}$. *We assume that the notion of interface also specifies, for every label, the port to which it is connected.*

A simple example in the λ-calculus shows that the number of incoming edges into bound ports changes during the derivation (and the labels can be copied). Take the expression $\lambda(x.\,@(\lambda(y.\,@(y,\,y)),\,x)$. The bound port of the leftmost λ has one incoming edge. After the contraction, in the same port there are two incoming edges.

The definition of residuals given in Section 4 may be obviously extended to arbitrary forms. An alternative, simpler way is to mark a form $\mathbf{f}$ and to take as residuals of $\mathbf{f}$, along a derivation σ, all the forms which are marked in the ending expression of σ. We have then some simple results.

Lemma 4 *Interfaces are preserved by reductions. That is, if a form* $\mathbf{f}'$ *is a residual of* $\mathbf{f}$ *along a reduction* σ*, their interfaces are the same. Moreover, for every edge leading to a bound port of* $\mathbf{f}'$*, if any, there exists an edge leading to the corresponding bound port of* $\mathbf{f}$ *having the same contiguous label.*

Proof: The proof is by induction over the length of σ. The basic case is obvious. When $\sigma = \sigma'; \mathbf{u}$, assume that the property holds in the ending expression of σ'. By cases over the relative positions of u and $\mathbf{f}$.

- **(u internal to the i-th argument of f)** According to the labeled rewriting rule contracting u, the interface of $\mathbf{f}$ cannot be changed by σ. Notice that, if $\mathbf{f}$ binds at the i-th argument, it is possible that the edges incoming into a bound port x can be deleted or copied. Assume that an edge incoming into the bound port x of $\mathbf{f}'$ follows from copying. Then its contiguous label must be the same of the original edge copied in $\mathbf{f}$ because it must appear inside some metavariable of the rewriting schema and it is copied without breaking joins. Thus the lemma.

- **(u external to f)** Then $\mathbf{f}$ must be inside one metavariable X of the rule contracting u. Thus, if $\mathbf{f}'$ is a residual of $\mathbf{f}$ along $\mathbf{u}$ then $\mathbf{f}'$ has still the same interface of $\mathbf{f}$ because X can be duplicated as a basic unit. Similarly, no change concerns edges incoming into bound ports of $\mathbf{f}$.

- **(σ is disjoint w.r.t. f)** Obvious.

$\square$

Lemma 5 *Let* $\mathbf{f}$ *be a form in a term reached by a derivation* σ *from a term* t *such that* $\mathtt{INIT}(t)$ *holds. Then*

1. *all the labels in the interface of* $\mathbf{f}$ *are different from each other.*

2. *if a label contiguous to* $\mathbf{f}$ *is atomic then all the others are atomic, too.*

Proof: (1) by the previous lemma, it suffices to prove the statement at the time **f** has been created. If **f** is a residual of a form in t, this is true by INIT(t). Otherwise, it is true by the definition of labeling in the rhs of the rewriting rules.

(2) According to the labeled rewriting rules, forms created along σ have not atomic contiguous labels. Therefore **f** is a residual of a form in t. The lemma is an obvious consequence of Lemma 4. ∎

Observe that, by Lemma 5.(1), we have a bijective correspondence between ports of **f** which are not bound and contiguous labels in the interface. Such correspondence fails for bound ports, however the situation is similar. In particular, note that if $\mathbf{f}'$ is a residual of **f**, we may have several contiguous labels to a same bound port which are identical, due to duplication of bound variables. By Lemma 4 these contiguous labels are a subset of the initial ones. The converse is the interesting result: every label contiguous to a port of same form (the port can be bound, too), uniquely identifies both the form and the bound port (provided that INIT holds, of course).

Theorem 7 *Let* INIT(t), *and let α be a proper label contiguous to some form in an expression reached from t. Then α uniquely defines the form to which it is connected, the port where the connection takes place and the interface of the form.*

Proof: By induction over the structure of the proper label α.

($\alpha = a$, $a \in L$) Let **f** be the form connected to a^+ (resp. a^-) which must be determinated. Firstly observe that **f** must be residual of a form $\mathbf{f}'$ in t. Indeed, according to the labeled rewriting rules, forms created along derivations starting at t have contiguous labels that are not atomic. Therefore, the form $\mathbf{f}'$ can be determinated by looking at the form positively (resp. negatively) connected at the unique edge labeled a in t. If a is not connected to a bound port, the theorem is an obvious consequence of Lemma 4. If a^- is connected to a bound port, again by Lemma 4, there exists an edge incoming into the bound port of $\mathbf{f}'$ having a as label. By INIT(t) there exists a unique node satisfying such property. The theorem is a consequence of Lemma 4.

($\alpha = (\ell)_s$) Note that the only possibility to create a label $(\ell)_s$ is to fire a redex whose degree is ℓ. Let $\ell = \alpha_1 \cdots \alpha_n$, where α_i's are proper. Thus the two forms connected to the edge labeled ℓ must be a destructor and a constructor **d** - **c**. Furthermore, these two forms, the ports involved by the connection and their interfaces only depend from α_1 and α_n. By induction there exists a unique pair of forms contiguous to α_1^+ and α_n^-. Therefore, it is possible to determinate the pair **d** - **c**. Let r be the unique rewriting rule firing the pair. In order to discover the pair of forms contiguous to α, we have just to look at the rhs of the labeled rule r. By definition there exists a unique occurrence of $(\ell)_s$ inside the rhs. Such occurrence must be positively (resp. negatively) connected to some form (otherwise the hypothesis should be invalidated). According to the definition of connection, the unique possibility is that this form is created by r. Thus, we are able to determinate uniquely the port where α is connected and the interface of the form by inspection of the rule r. By definition of connection, the unique case in which two forms may be connected by an edge labeled with $(-)_s$ is when they are both created by the rule. So the two forms connected by $(\ell)_s$ and the ports of this connection only depend from the rule r, which is uniquely individuated by ℓ.

$\square$

Corollary 1 *Let* INIT(t)*, and let ℓ be a label of an edge generated along a derivation starting at t. Then ℓ uniquely defines the two forms connected by the edge marked by it, the two ports where the connection takes place and the interfaces of the forms.*

Proof: Obvious consequence of Theorem 7. $\square$

Let us come to the proof of the main theorem.

Proof: (Of Theorem 3) We shall prove that, as soon as we know the structure of a label ℓ, any proper label α inside ℓ uniquely defines its "neighbours". As soon as we discover its neighbours, we reiterate the process, individuating in this way the entire label.

The proof is by cases according to the connections of α^- and α^+. Let us list them.

connection of α^-: (1) to the principal port of a constructor; **(2)** to the output port of of a destructor; **(3)** to a bound port of a destructor; **(4)** to a bound port of a constructor; **(5)** to the positive port of a proper label (i.e. α is a negative border label, see Remark 6).

connection of α^+: (A) to the principal port of a destructor; **(B)** to an auxiliary port of a destructor; **(C)** to an auxiliary port of a constructor; **(D)** to the negative port of a proper label (i.e. α is a positive border label, see Remark 6).

(negative connection) (1) Let **c** be the constructor to which α is connected. By definition of connection, the structure at its right (if any) must eventually be a bracket "$)_s$", for some s. That is α is the innermost label marking a redex whose constructor is **c**. More precisely, we have a label as $\alpha_1 \cdots \alpha_n \alpha$ which marks a redex. As soon as $\alpha_n, \cdots, \alpha_1$ are discovered, we can apply Theorem 7 in order to derive the connections of $(\alpha_1 \cdots \alpha_n \alpha)_s$ and, thus, reiterate the procedure.

(2) Let **d** be the destructor connected to α^-. Then, if a structure is present to the right of α, it must be a label $(\alpha_1 \cdots \alpha_n)_0$ where α_1 is the label contiguous to the principal port of **d**. By Theorem 7, α_1 can be determinated uniquely.

(3) Let $(\alpha_1 \cdots \alpha_n)_s$ be the structure to the right of α, where, again, α_1 is the label contiguous to the principal port of the destructor connected to α^-. Theorem 7 ensures that α_1 can be determinated uniquely. The situation is dual in case **(4)**, where we have a similar structure to the right but, now α_n is the label contiguous to the principal port of the constructor connected to α^-. Also α_n can be determinated uniquely.

(5) The label α cannot be atomic. Thus let $\alpha = (\ell)_s$. By definition of connection, the contiguous label to the right of α must be one (and it is determinated uniquely by the index s) of those contiguous to the negative, auxiliary ports of the constructor or destructor involved in the redex marked by ℓ, and, by Theorem 7, such labels can be uniquely determined. In particular, α is eventually a label of some metavariable X of the reduction marked by ℓ. Suppose X is the i-th metavariable of the constructor $\mathbf{c}$ (resp. of the destructor $\mathbf{d}$); then, at the right of α, we shall find the label contiguous to the unique negative port of the i-th input partition of $\mathbf{c}$ (resp. $\mathbf{d}$).

(positive connection) The case **(A)** is dual of **(1)**. In particular, here, the form connected to α^+ is a destructor $\mathbf{d}$. By definition of connection, the structure at its right (if any) must eventually be an open bracket. That is, α is the outermost label marking a redex whose destructor is $\mathbf{d}$. Let $(\alpha\alpha_1 \cdots \alpha_n)_s$ be such label. As soon as we determinate $\alpha_1, \cdots, \alpha_n$, we can apply Theorem 7 in order to derive the connections of $(\alpha_1 \cdots \alpha_n\alpha)_s$ and, thus, reiterate the procedure.

(B) According to the definition of connection, to the left of α we must find a structure like $(\alpha_1, \cdots, \alpha_n)_s$, where α_1 is the proper label contiguous to the principal port of the destructor which is connected to α^+. Theorem 7 guarantees that α_1 can be determinated uniquely. Case **(C)** is dual of the latter one. Here the proper label we are able to discover is α_n.

The interesting case is **(D)**. Note that, till now, we never use the information borrowed by the indexes of bullets in the structure. Furthermore α must be structured. Let $\alpha = (\ell)_s$. The unique possibility to yield a label of this format is to fire a redex marked by ℓ. Thus, by Theorem 7, ℓ defines a pair $\mathbf{d}$-$\mathbf{c}$ (and, hence, the rewriting rule contracting such redex). Since α^+ is connected to one contiguous label of a bound port x, then we must find the label relative to a *particular instance* of the variable x. Two cases are possible.

- At the left of α we may find the structure $\bullet_n$ (this is always the case, in the λ-calculus). Then this bullet represents a label contiguous to the form binding x (i.e. $\mathbf{c}$ or $\mathbf{d}$) in the suitable port individuated by x. In the case of λ-calculus we do not even have the problem of individuating the variable, since abstraction is always over a single variable. By definition of structure, $\bullet_n$ represents an atomic label. Thus the form binding x must be a residual of a form $\mathbf{f}$ in the initial expression t (forms created by σ, have structured contiguous labels, by definitions of labeled rewriting rules). Hence, by Lemma 4, the labels contiguous to the bound port x must be among those contiguous to the same bound port of $\mathbf{f}$. Now, by definition of structure, the index n uniquely determinates such atomic label.

- At the left of α we may find a structure $(\ell')_{s'}$. This means that the bound variable has been created along the reduction, and it has been eventually created at the same time of its binder. Note that the binder is one of the two forms $\mathbf{c}$ and $\mathbf{d}$ involved in the reduction labeled α. Suppose the binder is $\mathbf{c}$ (the case of $\mathbf{d}$ is analogous). Recall that $\alpha = (\ell)_s$ and suppose that $\ell = \beta_1 \dots \beta_k$, where every β_i is a proper label. Then β_k belongs to the interface of $\mathbf{c}$ (in the case of the destructor we shall take β_1). Since $\mathbf{c}$ has been created at the same time of x, β_k must be of the kind $(\ell')_{s''}$ (note the redundancy of information, here!).

$\square$

B The Contracted Label Lemma

The Contracted Label Lemma states that the labels "at the second step" are equal provided their structure are equal and the equality of the labels "at the first step". To this aim, a basic property is that the label of the first reduction occurs inside the degree yielded by a canonical derivation. This is guaranteed by the following proposition.

Proposition 6 *Let $t \xrightarrow{\sigma}\!\!\!\!\rightarrow t^\sigma \xrightarrow{v}$ be canonical and u be the leftmost outermost redex in t such that $\partial_t\,(u) \in \mathtt{at}(\partial_t^\sigma(v))$. Then u is the first redex fired by $\sigma; \mathbf{v}$.*

Proof: By induction over the length of σ. The basic case is immediate. Let us discuss the inductive step, i.e. $\sigma = t \xrightarrow{u} t' \xrightarrow{\sigma'}\!\!\!\!\rightarrow$.

As an easy consequence of the inductive hypothesis, there exists a redex u^+ in t such that $\partial_t\,(u^+) \in \mathtt{at}(\partial_t^\sigma(v))$. Therefore we must prove that the first redex fired by $\sigma; \mathbf{v}$ is exactly the leftmost outermost inside $\mathtt{at}(\partial_t^\sigma(v))$. Let $t_I \xrightarrow{u} t^\star$ be the isomorphic labeled reduction of $t \xrightarrow{u} t'$ and let $\mathtt{s}$ be the substitution mapping a label in t'_I into the label of the corresponding edge in $t^\star$. Let w be the first redex fired along σ'. Then, by induction $\partial_{t'}(w) \in \mathtt{at}(\partial_{t'}^{\sigma'}(v))$. The non trivial case is when $(\partial_{t'}(w))[\mathtt{s}]$ is atomic. That is $\partial_{t'}(w)$ must be the label of a redex which is internal to the subexpression singled out by u (because σv is canonical).

There are two subcases: **(A)** the reduction $\sigma'; \mathbf{v}$ is equal to $\mathbf{w}; \sigma_1; \mathbf{u}'; \sigma_2$ where $\mathbf{w}; \sigma_1$ is internal to the rhs of the reduction $\mathbf{u}$ and u' is a redex outside the rhs of $\mathbf{u}$ or **(B)** $\sigma'; \mathbf{v}$ is internal to the rhs of the reduction $\mathbf{u}$. Let $a = \partial_t\,(u)$.

(A) Let $t \xrightarrow{u;w;\sigma_1}\!\!\!\!\rightarrow t''$ then, necessarily, $a \in \mathtt{at}(\partial_t^{u;w;\sigma_1}(u'))$, this because the outermost form of the subexpression at u must be involved in the reduction u' (otherwise σv could not be canonical). Then the proposition follows easily by induction on $(\mathbf{u}'; \sigma_2)v$.

(B) Let H_u be the rhs of the schema contracting u. By induction over the structure of the sub(meta)expression H of H_u such that $\sigma' v$ is internal to H. We prove that, whatever the structure is, if the reduction $\sigma' v$ is internal to the rhs H then $\partial_t^\sigma(v)$ must satisfy $a \in \mathtt{at}(\partial_t^\sigma(v))$.
(**case** $H = x$ **or** $H = \mathbf{c}$) (the arity of $\mathbf{c}$ is empty) They are vacuous because there exists no reduction $\mathbf{w}$ inside H.
(**case** $H = \mathbf{f}(\vec{x}^1.\,H_1, \cdots, \vec{x}^n.\,H_n)$) If $\mathbf{f}$ is a constructor then $\sigma' v$ must be inside one H_i because $\sigma' v$ is canonical. Thus apply inductive hypothesis. If $\mathbf{f}$ is a destructor then it is possible that a (strict) prefix of $\sigma' v$ is internal to the first argument of $\mathbf{f}$ till a constructor $\mathbf{g}$ is produced such that $\mathbf{f}$-$\mathbf{g}$ can be fired. Notice that, in this case such redex, according to our labeled rules, has a degree containing a. Thus we yield the lemma through a reasoning similar to subcase **(A)**.

(**case** $H = X[^{H_1}/_{x_1}, \cdots, ^{H_n}/_{x_n}]$) If $\mathbf{w}; \sigma'; \mathbf{v}$ is internal to some instance of H_i the theorem follows by induction. The case in which w is internal to some H_i and the derivation $\sigma'; \mathbf{v}$ eventually goes outside can be proved similarly to **(A)**. Thus the interesting case is when w is a redex of X. In general the reduction $\sigma'; \mathbf{v}$ will be of the form $\sigma_1; \sigma_2; \mathbf{u}'; \sigma_3$ where σ_1 reduces redexes in X and σ_2 is a derivation internal to some instance of H_i and u' is the first redex not internal to that instance. We prove that the degree of u' contains a (and thus we can run again the reasoning in **(A)** for proving the theorem). Of course, $\sigma_2; \mathbf{u}'; \sigma_3$ must be a proper derivation otherwise $\sigma' v$ should be internal to X, contrarily to the fact that σv is canonical.

Now the critical observation is that all the instances of H_i (for every i) can be duplicated or deleted along σ_1, still remaining disjoint. This because after $\mathbf{u}$ such instances are trivially disjoint and a standard induction over the length of σ_1 shows that this property is preserved by firing redexes in X.

Thus if u' was a redex in another instance of H_i or in an instance of H_j, $j \neq i$, then a contradiction is easily raised. Actually $\mathbf{u}'$ must be disjoint "at the right" of σ_2 and the tailing $\mathbf{u}'; \sigma_3$ can not fire redexes involving subexpressions "disjoint to the left" (because σv is standard). Therefore, we can eventually simplify σv according to $\trianglerighteq$, contrarily to the hypothesis.

Consequently, the unique case is that u' is external to the instance of H_i. Here the outermost form of H_i must be involved in the reduction, otherwise the redex was ready to be fired before σ_2. Therefore the lemma, because the label of the outermost symbol of H_i contains a. $\square$

As already remarked, in general, a structured label "at the second step" (thus containing the degree of the first redex) no more identifies uniquely the edge. The phenomenon of duplication of structured labels (at the second step) plays an important role in the following. The main purpose of Proposition 7 below is to fix the cause of such duplications: the presence of substitutions in the rhs of schemas. To this aim, we need the notion of metacontext.

Given an IS, the set of *metacontexts*, ranged over $\mathcal{C}[\,]$, will be taken as the least set closed under the rules (1), (2) and (3) below:

$$(1)\ \ [\,] \qquad (2)\ \ \mathbf{f}(\vec{x}^{\,1}.\, H_1, \cdots, \vec{x}^{\,i}.\, \mathcal{C}[\,], \cdots, \vec{x}^{\,n}.\, H_n) \qquad (3)\ \ X[^{H_1}/_{x_1}, \cdots, ^{\mathcal{C}[\,]}/_{x_i}, \cdots ^{H_n}/_{x_n}]$$

that is, metacontexts are metaexpressions (also built up by using substitution) having an hole. We say that a (meta) subexpression H' of H *appears at layer* i if $H = \mathcal{C}[H']$ and $\mathcal{C}[\,]$ is a metacontext generated by exactly i instances of rule (3).

Proposition 7 *Let $t \xrightarrow{u} t'$ and $\partial_t(u) = a$. If there are two different edges in t' having $(a)_s$ as sublabel then the rhs of the labeled schema contracting u is a metaexpression of the form $\mathcal{C}[X[^{H_1}/_{x_1}, \cdots, ^{H_n}/_{x_n}]]$ where $(a)_s$ occurs in one of the H_i. Furthermore, if $(a)_s$ is the outermost label of the H_i and the two edges have the same label connected to $(a)_s^+$ then $X[^{H_1}/_{x_1}, \cdots, ^{H_n}/_{x_n}]$ appears at layer greater than 1.*

The last preliminary statement in view of Lemma 2 expresses the relationship between atomic labels inside degrees.

Proposition 8 *Let* $t \xrightarrow{\sigma}\!\!\!\!\rightarrow t' \xrightarrow{v}$ *be canonical and* a *be the leftmost outermost label (warning: label, not redex) in* $\mathtt{at}(\partial^\sigma_t(v))$ *and* u *be the access path of the form connected to the positive port of* a *in* t*. Then*

(1) *every redex* v' *fired by* $\sigma; \mathbf{v}$ *is of the form* $u \cdot v''$*, for some* v''*;*

(2) *if* $b \in \mathtt{at}(\partial^\sigma_t(v))$ *then there exists a connection of labels between* a *and* b*.*

Proof: (1) is an easy induction on length of σ. (2) is an immediate consequence of the property that labels in degrees are consistently joined. □

Now we have all the preliminaries needed by Lemma 2. For the sake of readability, we recall its statement.

Lemma 2 Let $t \xrightarrow{w} t' \xrightarrow{\rho}\!\!\!\!\rightarrow t^\rho \xrightarrow{u}$ and $t \xrightarrow{w} t' \xrightarrow{\sigma}\!\!\!\!\rightarrow t^\sigma \xrightarrow{v}$ be canonical. If $\mathtt{str}(\partial^\rho_{t'}(u)) = \mathtt{str}(\partial^\sigma_{t'}(v))$ then

$$\partial^{w;\rho}_t(u) \;=\; \partial^{w;\sigma}_t(v) \quad \Rightarrow \quad \partial^\rho_{t'}(u) \;=\; \partial^\sigma_{t'}(v)\,.$$

Proof: We prove the existence of a bullet (or bound-bullet) in $\mathtt{str}(\partial^\rho_{t'}(u))$ which is replaced by the same atomic label both in $\partial^\rho_{t'}(u)$ and $\partial^\sigma_{t'}(v)$. Then Theorem 3 makes the remaining work.

Let H_w be the rhs of the rule contracting w. Let $t_I \xrightarrow{w} t^+$ be the labeled derivation isomorphic to $t \xrightarrow{w} t'$ and $\mathtt{s}$ be the substitution replacing labels in t'_I with the corresponding labels in t^+. Then $(\partial^\rho_{t'}(u))[\mathtt{s}] = (\partial^\sigma_{t'}(v))[\mathtt{s}]$. Moreover, by Proposition 6, there exists an occurrence of $a = \partial_t\,(w)$ in $(\partial^\rho_{t'}(u))[\mathtt{s}]$. This occurrence is of the form $(a)_k$.

The property we prove is the existence in $(\partial^\rho_{t'}(u))[\mathtt{s}]$ and $(\partial^\sigma_{t'}(v))[\mathtt{s}]$ of a sublabel ℓ (in the same position) containing $(a)_k$ (for some k) such that $\mathtt{s}$ contracts ℓ in the same atomic label both in $\partial^\rho_{t'}(u)$ and $\partial^\sigma_{t'}(v)$. So, pick an occurrence of $(a)_k$ in $(\partial^\rho_{t'}(u))[\mathtt{s}]$: the property will be shown by induction over the layer at which $(a)_k$ appears.

The basic case (layer 0) is obvious. Notice that the property also holds when $(a)_k$ appears "on the border" of level 1, i.e. as in $X[\cdots, {}^{(a)_k H_g}/_{x_g}, \cdots]$. This because the positive connection of $(a)_k$ is the same both in $(\partial^\rho_{t'}(u))[\mathtt{s}]$ and $(\partial^\sigma_{t'}(v))[\mathtt{s}]$. Then, by Proposition 7, ℓ identify uniquely an edge in t'.

Let us discuss the inductive step (layer $n+1$). Assume the Lemma holds when $(a)_k$ appears at layer n. Now, suppose that H_w has the form $\mathcal{C}[X[{}^{H_1}/_{x_1}, \cdots, {}^{H_r}/_{x_r}]]$ where $\mathcal{C}$ has been built up with exactly n instances of rule (3) and $(a)_k$ appears in H_i. The interesting subcase is when there are several edges in t^+ having $(a)_k$ as sublabel. Let $a^{(1)}, \cdots, a^{(j)}$ be the atomic labels of such edges in t'_I. Suppose furthermore that the same $\bullet$ in $\mathtt{str}(\partial^\rho_{t'}(u))$ and $\mathtt{str}(\partial^\sigma_{t'}(v))$ is replaced by $a^{(p)}$ and by $a^{(q)}$ with $p \neq q$. There are *two* subcases.
(*subcase A*) The label $(a)_k$ marks H_i. This case follows by induction because the positive connection of $(a)_k$ is equal both in $(\partial^\rho_{t'}(u))[\mathtt{s}]$ and $(\partial^\sigma_{t'}(v))[\mathtt{s}]$.

(*subcase B*) The label $(a)_s$ appears inside the H_i. Thus take $A_u = \mathtt{at}(\partial^{\rho}_{t'}(u))$ and $A_v = \mathtt{at}(\partial^{\sigma}_{t'}(v))$. The case when A_u or A_v individuates (by Proposition 8) a subexpression which is internal to H_i is not possible because at least one between $(\mathbf{w};\rho)u$ and $(\mathbf{w};\sigma)v$ should be $\trianglerighteq$-simplifiable. When both A_u and A_v are not internal, let b be an atomic label which is outside the subexpression replacing H_i in $t^{\dagger}_I$ and belonging to A_u. We reason over A_u, the case of A_v can be treated symmetrically. By Proposition 8, there exists a connection between $a^{(p)}$ and b. There are two possibilities for $a^{(p)}$ to be connected to b. The first one is to cross the outermost label of H_i, let us say c. Observe that, according to the labeled rewriting rules, $(c)[\mathtt{s}]$ must have a sublabel of the form $(a)_{k'}$, for some k'. Thus, take the occurrence of $(a)_{k'}$ in $\partial^{w;\rho}_t(u)$: we have reduced to the subcase A.

The second possibility is that $a^{(p)}$ is connected to b through a bound port. Namely, there exists a form $\mathbf{f}$ outside the term replacing H_i whose bound variable occurs inside the term replacing H_i and $a^{(p)}$ is connected to that variable. Let z be the occurrence of the edge of such variable in t' and let ℓ_z be its label in t'_I. Notice that if $(\ell_z)[\mathtt{s}]$ is atomic then, by the format of our rules, $\mathbf{f}$ must be a form outside w. Thus, by Proposition 9, it is not possible to fire $\mathbf{f}$ during $\mathbf{w};\rho;\mathbf{u}$. Hence this case is vacuous. If $(\ell_z)[\mathtt{s}]$ is not atomic, then it must contain $(a)_{k'}$, for some k'. So $(a)_{k'}$ occurs in $(\partial^{\rho}_{t'}(u))[\mathtt{s}]$. Such occurrence is, by the assumptions ($\mathbf{f}$ is outside H_i) at a layer less then $n+1$. $\square$

C The Contracted Structure Lemma

Lemma 3 needs some preliminaries. Firstly we have to recognize whether a form is fired along a canonical derivation or less. The following two propositions fulfil this aim. Proofs are omitted because standard. Below we say that an argument of a form $\mathbf{f}$ is bound if $\mathbf{f}$ binds some variable over it.

Proposition 9 *Let $t \xrightarrow{w;\sigma}$ be standard and let $\mathbf{f}$ be a form at $v \prec w$ in t.*

1. *If w is internal to a bound argument of $\mathbf{f}$ then $\mathbf{f}$ can not be fired along σ.*

2. *If w is internal to an argument of $\mathbf{f}$ greater than the first then $\mathbf{f}$ is never fired along σ.*

Proposition 10 *Let $\mathbf{f}$ be a form in t: $\mathbf{f}$ or a residual of its is fired along a canonical derivation $t \xrightarrow{\sigma u}\!\!\!\!\rightarrow$ if and only if there exists a label contiguous to $\mathbf{f}$ that belongs to $\mathtt{at}(\partial^{\sigma}_t(u))$.*

As said, the duplication of bound labels plays an important role in making matters more difficult in the general case. The following proposition states necessary and sufficient conditions so that a label marks two different bound edges.

Proposition 11 *Let $t \xrightarrow{u} t'$ and $\mathtt{INIT}(t)$. The expression t' has a form $\mathbf{f}$ at $v \succeq u$ having at least two bound edges marked in the same way if and only if the following three provisos hold:*

1. *the rhs of the schema contracting u has a subexpression $\mathbf{f}(\vec{x}^{\,1}.\,H_1,\cdots,\vec{x}^{\,n}.\,H_n)$ and the $\mathbf{f}$ at v is a copy of the above form;*

2. *the two bound edges come into the same bound port y (of the j-th input partition of $\mathbf{f}$);*

3. *H_j has a sub(meta)expression of type $X[^{H'_1}/_{x_1},\cdots,^{H'_m}/_{x_m}]$ where at least one H'_i is not a variable, y occurs free at least once into one of the expressions replacing (not variables) H'_i and t_X, the expression replacing X, has at least two free occurrences of x_i. The two bound edges are those inside the instance of $X[^{H'_1}/_{x_1},\cdots,^{H'_i}/_{x_i},\cdots,^{H'_m}/_{x_m}]$ and correspond to different instances of the same H'_i.*

Along Lemma 3, we will use the notion of contracted label. Here is the formal definition.

Definition 17 *Let $t \xrightarrow{w} t' \xrightarrow{\sigma}\!\!\!\!\rightarrow t^\sigma \xrightarrow{v}$. Then $\partial^\sigma_{t'}(v)$ will be called the* contracted label *of $\partial^{w;\sigma}_t(v)$ along $\mathbf{w}$.*

We recall that, if $\mathtt{s}$ is the substitution mapping a label in t'_I into a label in t^+, with $t_I \xrightarrow{w} t^+$ isomorphic to $t \xrightarrow{w} t'$, then $(\partial^\sigma_{t'}(v))[\mathtt{s}] = \partial^{w;\sigma}_t(v)$. This means that $\partial^\sigma_{t'}(v)$ has a simpler structure than $\partial^\sigma_t(v)$. The reader can gain intuition by thinking that (non atomic) sublabels of $\partial^{w;\sigma}_t(v)$ are "contracted" into atomic labels.

For the sake of readability, we recall the statement of Lemma 3.

Lemma 3 (The Contracted Structure Lemma) Let $t \xrightarrow{w} t' \xrightarrow{\rho}\!\!\!\!\rightarrow t^\rho \xrightarrow{u}$ and $t \xrightarrow{w} t' \xrightarrow{\sigma}\!\!\!\!\rightarrow t^\sigma \xrightarrow{v}$ be canonical. Then

$$\mathtt{str}(\partial^\rho_{t'}(u)) \neq \mathtt{str}(\partial^\sigma_{t'}(v)) \;\;\Rightarrow\;\; \partial^{w;\rho}_t(u) \neq \partial^{w;\sigma}_t(v).$$

Proof: Let $t_I \xrightarrow{w} t^+$ be the labeled reduction isomorphic to $t \xrightarrow{w} t'$ and $\mathtt{s}$ be the substitution mapping a label in t'_I into that marking the corresponding edge in t^+. Then $\partial^{w;\rho}_{t'}(u) = (\partial^\rho_{t'}(u))[\mathtt{s}]$ and $\partial^{w;\sigma}_t(v) = (\partial^\sigma_{t'}(v))[\mathtt{s}]$.

The lemma has an easy subcase: when the difference between $\mathtt{str}(\partial^\rho_{t'}(u))$ and $\mathtt{str}(\partial^\sigma_{t'}(v))$ is not due to bound bullets only. Indeed, it is immediate to check that $\mathtt{s}$ preserves the structure of $(-)_k$ (thus also the differences).

So assume that $\mathtt{str}(\partial^\rho_{t'}(u))$ and $\mathtt{str}(\partial^\sigma_{t'}(v))$ differ for bound bullets only and take a position in $\mathtt{str}(\partial^\rho_{t'}(u))$ with a bullet indexed differently of the corresponding position in $\mathtt{str}(\partial^\sigma_{t'}(v))$. Let b and c be the labels of the bound edges in t'_I replacing the two bullets in $\partial^\rho_{t'}(u)$ and $\partial^\sigma_{t'}(v)$.

By contradiction, assume that $\partial^{w;\rho}_t(u) = \partial^{w;\sigma}_t(v)$. Then the forms connected negatively to b and c are created by $\mathbf{w}$. Indeed, by Proposition 9 they cannot be outside w; on the other hand they can not be inside w because, in this case, $\mathbf{w}$ should copy the forms without modifying their bound edges.

Let H_w be the rhs of the schema contracting w and $a = \partial_t\,(w)$. Let $\mathbf{f}_1,\cdots,\mathbf{f}_p$ be the set of forms created by $\mathbf{w}$ and individuated by different bound bullets in $\mathtt{str}(\partial^\rho_{t'}(u))$ and $\mathtt{str}(\partial^\sigma_{t'}(v))$.

1. Unicity of the form $\mathbf{f}$. Let $\mathbf{f}$ be the leftmost outermost form in H_w among $\mathbf{f}_1, \cdots, \mathbf{f}_p$. Therefore $H_w = \mathcal{C}[\mathbf{f}(\cdots)]$, for some context $\mathcal{C}$. Firstly we prove that the instance of $\mathbf{f}$ to which b and c are connected must be the same. Indeed, in order to be duplicated, $\mathbf{f}$ must be in a context as $X[\cdots {}^{\mathcal{C}'[\mathbf{f}(\cdots)]}/_x \cdots]$. Now take $A_u = \mathtt{at}(\partial^\rho_{t'}(u))$ and $A_v = \mathtt{at}(\partial^\sigma_{t'}(v))$. If b and c are connected to different instances of $\mathbf{f}$, then A_u and A_v may be in different instances of $\mathcal{C}'[\mathbf{f}(\cdots)]$ or not. The first case immediately gives that one between $(\mathbf{w};\rho)u$ and $(\mathbf{w};\sigma)v$ can be simplified w.r.t. $\rhd_3$. In the second case A_u and A_v may exit "from the top" of $\mathcal{C}'[\mathbf{f}(\cdots)]$ or "from the bottom" through bound edges. In the first sub-case, by the Connection Property, we eventually find two corresponding labels in $\partial^\rho_{t'}(u)$ and $\partial^\sigma_{t'}(v)$ that $\mathtt{s}$ substitutes with two different labels ("the borders" of the instances of $\mathcal{C}'[\mathbf{f}(\cdots)]$). Thus the contradiction is immediate. In the other sub-case, the unique possibility is that we exit from two bound edges connected to a form created by H_w. This means that in $\mathtt{str}(\partial^\rho_{t'}(u))$ and $\mathtt{str}(\partial^\sigma_{t'}(v))$ we find two corresponding bullets $\bullet_e$ and $\bullet_{e'}$, respectively. The case $e \neq e'$ contradicts the assumption that $\mathbf{f}$ was the leftmost outermost form in H_w individuated by different bound bullets in the two structures. If $e = e'$ ($\bullet_e$ and $\bullet_{e'}$ are replaced by different bound edges of different instances of a form created by H_w) then start from them and reiterate the reasoning. Eventually a contradiction will be raised.

So the case we must discuss is that both ρu and σv fire the same form $\mathbf{f}$ and there exists one occurrence of $\bullet_B$ in $\mathtt{str}(\partial^\rho_{t'}(u))$ that is a bound edge of $\mathbf{f}$ and such that the corresponding bullet in $\mathtt{str}(\partial^\sigma_{t'}(v))$ is indexed by C, with $B < C$ (remember that b and c replace $\bullet_B$ and $\bullet_C$ in $\partial^\rho_{t'}(u)$ and $\partial^\sigma_{t'}(v)$, respectively). For simplicity assume that the label b marks the leftmost edge coming into the bound port of $\mathbf{f}$ and whose label in t^+ is $b[\mathtt{s}]$. The general case is similar.

2. Looking for simplifying $(\mathbf{w};\sigma)v$. We prove that $(\mathbf{w};\sigma)v$ is not canonical (it can be shifted). By Proposition 11, the duplication of b and c follows because one argument of $\mathbf{f}$ in H_w has the shape $H' = \mathcal{C}'[X[\cdots, {}^{H_i}/_{x_i}, \cdots]]$. The situation is illustrated in Figure 6 below. Notice again that, by traveling towards the positive connections of b and c it is not possible

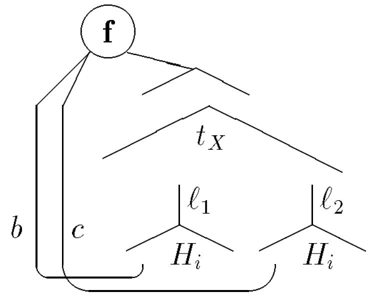


Figure 6: A sub-expression at the "second step" of σ

to exit H_i "from the top" because the labels ℓ_1 and ℓ_2 are different in t^+. The problematic case is the negative connection of b and c, because it is the same.

Let $\mathbf{w};\sigma = \mathbf{w};\mathbf{v}_1;\cdots;\mathbf{v}_m$ and σ_p be the tailing of $(\mathbf{w};\sigma)v$ "at the p-th step", i.e. $\sigma_p = t^p \xrightarrow{v_p} t^{p+1} \xrightarrow{v_{p+1}} \cdots \xrightarrow{v_m} t^{m+1} = t^\sigma$ and $t^p_I \xrightarrow{v_p} t^{p+1}_p \xrightarrow{v_{p+1}} \cdots \xrightarrow{v_m} t^{m+1}_p$. Moreover, let $\mathtt{s}_p$ be the substitution mapping atomic labels in t_I^p into t_0^p, the labeled term yielded by $\mathbf{w};\sigma$ after $p+1$ steps). Notice that $(\partial^{\sigma^p}_{t^p}(v))[\mathbf{s}_p] = \partial^{w;\sigma}_{t}(v)$.

Along σv, $\partial^{\sigma}_{t'}(v)$ is *via via* contracted till an edge (representing the redex v). The same happens at c. Let s be the index of the least contraction in $\mathbf{v}_1; \cdots; \mathbf{v}_m$ such that every $\mathbf{v}_i$, $1 \leq i < s$, always contracts the residual of c [2] into a bound edge (let us call it c^i) and $\mathbf{v}_s$ opens c^s in t^{s+1}. It is not difficult to prove that, for every $1 \leq j < s$:

> if T is the expression replacing H' along $\mathbf{w}$, then, the residuals of c during $\mathbf{v}_1; \cdots; \mathbf{v}_{s-1}$ always live into expressions that are residuals of T (i.e. expressions that are T except for the edges interfacing T with the external world: the positive connection of T and the free variables in it). Therefore, the relative positions of residuals of b and c never changes.

Now, let us see what happens when the bound edges b^s and c^s are opened (they are opened at the same time because they income into the same port). In particular, the rhs of $\mathbf{v}_s$ has a subexpression $Z[\cdots, {}^{H'_q}/_{z_q}, \cdots]$ such that b^{s+1} and c^{s+1} mark two copies of z_q. Figure 7 illustrates the scenario after the reduction $\mathbf{v}_s$. Furthermore, by the above discussion,

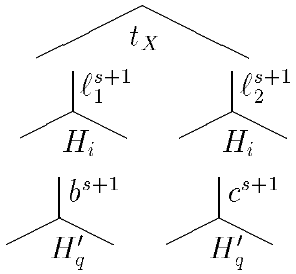


Figure 7: The situation after the reduction $\mathbf{v}_s$

we can single out a subexpression T in (that replacing) Z such that T is the "residual" of the subexpression replacing H' along $\mathbf{w}$ and b^{s+1} and c^{s+1} are internal to $T[\cdots, {}^{H'_q}/_{z_q}, \cdots]$.

We can be even more precise. As said before, by Proposition 11, we eventually find, inside H', a subexpression $\Upsilon = X[\cdots, {}^{H_i}/_{x_i}, \ldots]$ such that edges marked b and c are due to multiple occurrences of x_i in t_X, the expression replacing X along $\mathbf{w}$. Thus, by exploiting the same notation of Proposition 2, the subexpression of $T[\cdots, {}^{H'_q}/_{z_q}, \cdots]$ where b^{s+1} and c^{s+1} occur can be written as

$$\Upsilon_s \;=\; (\cdots \Upsilon[\cdots, {}^{y_1}/_{x_i}, \cdots])_{i_1} \cdots)[\cdots, {}^{z_q}/_{y_k}, \cdots])_{i_{k+1}}[\cdots, {}^{H'_q}/_{z_q}, \cdots])_s$$

Observe that reductions $\mathbf{v}_j$, $1 \leq j < s$ can just rename the bound variable represented by the edges labeled b^j and c^j. In this case, at the same time, $\mathbf{v}_j$ must create a new form binding again b^{j+1} and c^{j+1}.

The next step consists of proving

[2] Here we use the term *residual* with a bit of informality. It is quite clear that it means "the bullet resulting from the contraction of a structure where a given bullet participates". In the following of the proof, we also use the name *residual* for expressions, with the purpose of characterizing the change of the address of an expression after a reduction.

(1) the existence of an index s', $s < s' \leq m$, such that, for every $s < j < s'$, $\mathbf{v}_j$ fires a redex external to Υ_s and $\mathbf{v}_{s'}$ fires a redex having at least one form internal to Υ_s;

(2) that at each step $\mathbf{v}_j$, $s < j < s'$, the subexpressions recognized by the edges ℓ_1^j and ℓ_2^j are copies.

(1) is true because, by Proposition 10, the forms contiguous to c^{s+1} must be fired along $(\mathbf{v}_{s+1}; \cdots; \mathbf{v}_m)v$ and such forms are internal to Υ_s

(2) is an easy check by induction: the only interesting case is when the constructor or destructor involved in $\mathbf{v}_j$ binds free variables internal to ℓ_2^j. Then, by induction, it must also bind a variable internal to ℓ_1^j with the same name. Hence any possible substitution makes copies both in the subexpression of ℓ_2^j and in the subexpression of ℓ_1^j.

The important remark is that Υ_s changes along $\mathbf{v}_{s+1}; \cdots; \mathbf{v}_{s'}$. In particular, at the generic step $s < j < s'$:

$$\Upsilon_j \ = \ (\Upsilon_{j-1}[\cdots,{}^{H_r''}/_{z_r}, \cdots])_j$$

where $[\cdots,{}^{H_r''}/_{z_r}, \cdots]$ is the substitution involving some free variable in Υ_{j-1}.

3. Performing the shift. Now we state that $(\mathbf{v}_{s'}; \cdots; \mathbf{v}_m)v$ can be shifted w.r.t. $\mathbf{v}_{s'-1}$. The first step is to prove that the redex $v_{s'}$ is internal to the subexpression recognized by $\ell_2^{s'-1}$. Let us see why $\mathbf{v}_{s'}$ cannot fire a form $\mathbf{h}$ inside (the residual of) t_X. Indeed the occurrence of $c^{s'-1}$ in $\partial^{\sigma^{s'-1}}_{t^{s'-1}}(v)$ should be connected to the form $\mathbf{h}$. There are two ways to reach "by connection" the form $\mathbf{h}$ from $c^{s'-1}$:

(a) through bound edges internal to the edge $\ell_2^{s'-1}$ or

(b) by traversing the edge labeled $\ell_2^{s'-1}$.

The case (a) requires that $\mathbf{h}$ should bind some variable under $\ell_2^{s'-1}$. This is not possible, by definition of substitution. (b) implies that the edge $\ell_2^{s'-1}$ is reached "positively" from $c^{s'-1}$. In this case, it is possible to show that also the ancestor c should reach "positively" the ancestor ℓ_2. And this was not possible.

The last step consists of proving that the derivation $(\mathbf{v}_{s'}; \cdots; \mathbf{v}_m)v$ is internal to the expression recognized by $\ell_2^{s'-1}$. Actually there is no way for escaping, since forms binding variables under $\ell_2^{s'-1}$ cannot be fired any more, by Proposition 9.

We leave to the reader the check that, running the procedure `shift-indexes`, we are able to built an expression $\Upsilon_{s'}$. ∎