Minimal and Optimal Computations of Recursive Programs

GÉRARD BERRY

École des Mines, Valbonne, France

AND

JEAN-JACQUES LÉVY

Iria-Laboria, Rocquencourt, France

ABSTRACT. Vuillemin's results on optimal computations of recursive programs are generalized. New syntactic results are obtained by considering spaces of derivations instead of terms. The results apply to classes of interpretations more general than the sequential interpretations of Vuillemin.

KEY WORDS AND PHRASES: semantics, recursive programs schemes, rewriting systems, parameters, passing mechanisms

CR CATEGORIES: 5.24

1. *Introduction*

1.1 PURPOSE. Procedure call mechanisms have been studied in two main frameworks: the λ-calculus for the most general case where procedures can be passed as arguments and given as results, and the recursive program schemes for the simpler case where arguments and results are pure values. We restrict here our attention to the recursive programs, but we shall indicate which results also hold in the λ-calculus. A complete treatment of the pure λ-calculus is given in [15].

The behavior of an interpreter is modeled by some *operational semantics*: According to some parameter passing rules (Algol's call-by-value or call-by-name, for example), some function calls are evaluated by replacing the function call by the function body. It is well known that different computation rules can give different results, that there is a well-defined notion of "best result," and also that the computation costs may differ from one rule to another. The optimality problem we study in this paper, continuing the work of [25], is therefore very simply stated: How can we obtain the best result with the least possible cost? It can be split into two parts: How can we obtain the best result, and how can we obtain any result with the least possible cost?

The first problem is usually called the *correctness problem*, and is related to Scott's denotational semantics. This semantics considers a program as a system of functional equations whose solution is the least fixpoint of some continuous function on appropriately ordered function domains. Cadiou [5] shows that the best result is precisely the result defined by the denotational semantics and that it can be computed by a full substitution process. Vuillemin [25, 26] gives a sufficient safety criterion for a rule to be correct, i.e. to compute the best result. This criterion is extended by Downey and Sethi [9] into a necessary

Authors' addresses: G. Berry, École des Mines, Sophia-Antipolis, 06560 Valbonne, France; J.-J. Lévy, Iria-Laboria, 78150 Rocquencourt, France.

and sufficient security criterion of a more syntactic nature. The second problem is called the *optimality problem.* Vuillemin shows that under some sequentiality conditions every safe rule can be transformed into an optimal one by using a sharing mechanism in the implementation. This result is shown by different techniques in Montangero, Pacini, and Turini [18], and the underlying concept of "lazy evaluation" is used in [10].

Our purpose is to extend these results in several directions and simplify their proofs. Our point of view is purely syntactic: We study properties of the program scheme considered as a rewriting system and use them to reduce semantic problems into syntactic ones. We first notice that the results of Vuillemin [26] and Downey and Sethi [9] are based on a lattice property of terms. This property holds only under syntactic restrictions which are not very compelling on recursive programs but disallow any easy extension to more complicated rewriting systems such as the λ-calculus. We remove these conditions and reconstruct a lattice property in derivation spaces instead of term spaces. Similarly, the sharing techniques used in recursive programs are rather trivial and can also not be easily extended to other systems. By introducing a notion of *family* of subterms which is also definable in the λ-calculus [15] (and therefore may probably be axiomatized), we show why this sharing technique is really the suitable one and explain its behavior.

Infinite derivations can be directly studied with these techniques. In particular we define a notion of $\mathcal{F}$-optimality for infinite derivations based on the notion of family of subterms. This $\mathcal{F}$-optimality implies the usual optimality for finite derivations. We show that $\mathcal{F}$-optimal derivations exist in the *symbolic or Herbrand interpretation* which plays an important role in the study of program equivalence and symbolic computations [7]. We then give a sufficient *projectivity condition* for $\mathcal{F}$-optimal derivations to exist in an interpretation (this condition is also necessary when the interpretation is algebraic [7]). This condition strictly extends Vuillemin's sequentiality condition [26], since the Herbrand interpretation is projective but not sequential. Other interpretation classes defined in [2, 13] also contain projective interpretations.

The rest of this section presents an overview of the techniques and results and technical preliminaries. Section 2 is devoted to the study of derivation spaces. Families and sharing techniques are presented in Section 3. Minimality and optimality results are given in Section 4.

1.2 Overview and Examples

Program Schemes and Interpretations. The programs that we consider are recursive programs without assignments or side effects, like the Fibonacci program

$$\text{P1}: \mathit{fib}(x) = \textbf{if } x = 0 \textbf{ or } x = 1 \textbf{ then } 1 \textbf{ else } \mathit{fib}(x-1) + \mathit{fib}(x-2)$$

It is immediately necessary to distinguish between the *program scheme* and the *interpretation.* The program scheme in this example is a purely syntactic object of the form

$$\Sigma_1\colon \phi(x) = f(x, h(\phi(p(x)), \phi(p(p(x))))),$$

where the symbols f, h, and p are called basic function symbols and ϕ is called the unknown function symbol. The interpretation is defined by some data domain and by some interpretation of the basic function symbols as mappings on this domain. Following Scott [23], we require data domains to be complete partial orders (i.e. ordered sets having least element $\perp$, the "undefined," and such that any increasing chain has a limit) and we require functions to be monotonic and continuous. For example, the program P1 is obtained from the program scheme Σ_1 by the following interpretation I:

$$D_I = \begin{tikzcd}[column sep=small] 0 \arrow[drr, dash] & 1 \arrow[dr, dash] & 2 \arrow[d, dash] & \cdots\ n \arrow[dl, dash] \ \cdots \\ & & \perp & \end{tikzcd} \qquad \text{with } \perp \sqsubseteq n \text{ for all } n;$$

$$f_I(m, n) = \begin{cases} 1 & \text{if } m = 0 \text{ or } m = 1, \\ n & \text{if } m \neq \perp,\ m \neq 0,\ m \neq 1, \\ \perp & \text{otherwise;} \end{cases}$$

$$h_I(m,n) = \begin{cases} m+n & \text{if } m \neq \perp \text{ and } n \neq \perp, \\ \perp & \text{otherwise;} \end{cases}$$

$$p_I(m) = \begin{cases} m-1 & \text{if } m \neq \perp \text{ and } m \neq 0, \\ \perp & \text{otherwise.} \end{cases}$$

Let us call a *data mapping* or *valuation* any mapping v from the set V of variables into the interpretation domain D_I. Once I fixes the values of the function symbols and v fixes the values of the variables, the value of any term t not containing unknown function symbols is completely determined. We call it $(I, v)t$.

The Operational Semantics. The Herbrand Interpretation. We call a *program* a triple $P = (\Sigma, I, v)$. Given a program P, a *semantics* is a way of associating a value with every term (which may contain unknown function symbols). The semantics we consider are operational semantics defined in terms of computations (see [20, 25] for the definitions and equivalence proofs of algebraic and fixpoint semantics). We define the semantics from the set of all possible computations of a term. We sometimes informally refer to computation rules, which are recursive ways of associating computations to terms [1, 9, 25].

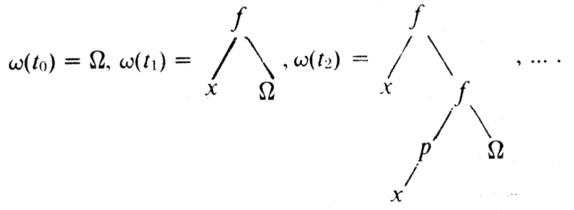
The basic syntactic operation is the *rewriting of an unknown function symbol*, which corresponds to procedure calls in usual languages. The rewritten symbol is replaced by the corresponding right-hand-side term of the program scheme, as in the following example:

$$\Sigma_2\colon \phi(x) = f(x, \phi(p(x))),$$
$$f(y, \phi(q(x))) \rightarrow f(y, f(q(x), \phi(p(q(x))))).$$

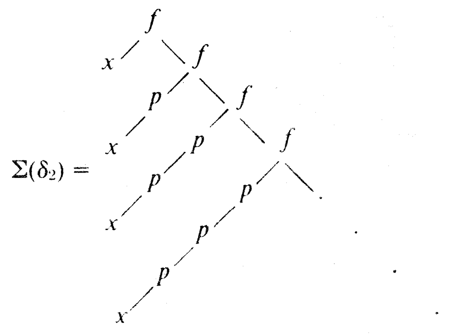
A *derivation* (respectively *infinite derivation*) is a finite sequence (respectively infinite sequence) of such rewritings. An example of an infinite derivation of $\phi(x)$ by Σ_2 is

$$\delta_2\colon t_0 = \phi(x) \rightarrow t_1 = f(x, \phi(p(x))) \rightarrow t_2 = f(x, f(p(x), \phi(p(p(x))))) \rightarrow \cdots.$$

Let us denote by $\omega(t)$ the term obtained by replacing all occurrences of unknown function symbols by the syntactic undefined symbol Ω. Then any derivation δ: $t_0 \rightarrow t_1 \rightarrow \cdots \rightarrow t_n \rightarrow \cdots$ determines a sequence $\omega(t_0), \omega(t_1), \ldots, \omega(t_n), \ldots$. For example, the sequence determined by δ_2 is



We use a tree representation to show that the $\omega(t_i)$'s form an increasing chain in the "initial segment" ordering $<$, defined by $t < t'$ holds iff t' can be obtained from t by replacing some Ω's by some other terms. The *symbolic value* $\Sigma(\delta)$ computed by δ is the limit of the increasing chain $\omega(t_i)$ and is represented by some infinite tree. We have, for example,



If I is an interpretation and ν is a data mapping, then every derivation δ: $t_0 \rightarrow t_1 \rightarrow \cdots \rightarrow t_n \rightarrow \cdots$ determines an increasing chain $(I, \nu)\omega(t_0)$, $(I, \nu)\omega(t_1)$, ... , $(I, \nu)\omega(t_n)$, ... in D_I. This chain has a limit $(I, \nu)\delta$ in D_I, called the *value computed by* δ *in* (I, ν). With I and δ_2 as in the above examples and with $\nu(x) = 2$, the chain determined by δ_2 is $\perp, \perp, 2, 2, \ldots$ and the value computed by δ_2 is 2.

Note that $\Sigma(\delta)$ is the value computed by δ in a particular interpretation (H, ν_H): The domain of H is the set of all finite and infinite trees written with basic function symbols, variables, and Ω. The functions f_H are trivially defined by

$$f_H(A_1, A_2, \ldots, A_k) = \begin{tikzcd}[column sep=small] & f \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\ A_1 & A_2 \cdots & A_k \end{tikzcd}$$

and ν_H is given by $\nu_H(x) = x$ for all variables x.

Note also that by the continuity hypothesis for functions, the value $(I, \nu)\delta$ is also the value $(I, \nu)(\Sigma(\delta))$ of the infinite tree computed by δ in H.

Using the Church-Rosser property (see Section 2), it can be shown that the set of values of all possible computations starting from a term t has a least upperbound $P(t)$ in D_I, defined as the *semantics of t for P.*

The Correctness Problem. Assume (I, ν) is given, and call *correct* a derivation δ such that $(I, \nu)\delta = P(t)$. The problem is to characterize in a syntactic way the correct derivations. Several results are well known: the "full derivation" [5, 20, 25], which simultaneously rewrites all occurrences of unknown function symbols (an operation defined in Section 2), or the "parallel outermost derivation" in which only all outermost occurrences are rewritten, are always correct. "Innermost derivations" like the usual call-by-value derivation are not correct in general, as Morris's classical example [19] shows:

$$\text{P3: } \phi(x, y) = \textbf{if } x = 0 \textbf{ then } 0 \textbf{ else } \phi(x - 1, \phi(x, y))$$

Here the innermost evaluation of $\phi(2, 0)$ is (with simplifications):

$$\delta_3\text{: } \phi(2, 0) \rightarrow \phi(1, \phi(2, 0)) \rightarrow \phi(1, \phi(1, \phi(2, 0))) \rightarrow \cdots,$$

which has value $\perp$ while $\text{P3}(\phi(2, 0)) = 0$. Downey and Sethi [9] and Vuillemin [26] show that, roughly speaking, correct derivations are of the type "outermost first."

The Optimality Problem. Call *optimal* a derivation which computes the best possible value with a minimum number of rewritings, assuming for simplicity that some finite derivation does compute this best value. Since an optimal derivation is correct, it must perform the rewritings in an "outermost first" way, at least if we want to produce it by a uniform "computation rule" implemented in some interpreter. However, outermost derivations tend to have the classical inefficiencies of call-by-name: If the term $\phi(\phi(1, 0), 0)$ is computed in an outermost way with P3, we get

$$\phi(\phi(1, 0), 0) \rightarrow \textbf{if } \phi(1, 0) = 0 \textbf{ then } 0 \textbf{ else } \phi(\phi(1, 0) - 1, \phi(\phi(1, 0), 0))$$

where $\phi(1, 0)$ has been duplicated and must be computed twice, so that the computation of the result 0 needs 6 steps. It takes only 4 steps if $\phi(1, 0)$ is evaluated first.

Therefore the optimality problem is twofold: We must avoid useless rewritings and avoid duplications. The usual solution is to share duplicated objects [18, 25], which can be described by representing terms as directed acyclic graphs (dag's) or pointer structures:

$$\begin{tikzcd}[column sep=tiny, row sep=small] \phi(\phi(1, 0), 0) \rightarrow \textbf{if} & {} \arrow[dr] & = 0 \textbf{ then } 0 \textbf{ else } \phi(& {} \arrow[dl] & - 1, \phi(& {} \arrow[dlll] & , 0)) \\ & & \phi(1, 0) & & & & \end{tikzcd}$$

The occurrences of $\phi(x - 1, 0)$ are now represented by a unique object, and outermost derivations become as efficient as innermost ones.

Notice that we count only function calls and not simplification steps when evaluating the cost of a derivation: We evaluate "for free" the terms $\omega(t_i)$. A possibly more realistic approach might consider computations of the form $fib(2) \to fib(1) + fib(0) \to 1 + 1 \to 2$, where the last step is a basic function evaluation of unit cost.

Residuals and Families of Occurrences. Two techniques can be used to model derivations with sharing: Abandon terms and consider dag derivations as in [18], or introduce a new notion of derivation in terms which simulates the derivation in dag's as in [25]. We choose the second technique, which is certainly more general. In particular it is applicable to the λ-calculus [15] where dag's will not suffice (the corresponding data structure is presently not known). Our constructions are based on Church's [6] notions of *residual.* Consider the following derivation, where the superscripts name occurrences:

$$\Sigma_5\colon \phi(x) = f(x, \phi(x)),$$

$$\begin{tikzcd}
\phi^1 \arrow[d, dash] & & & \phi^4 \arrow[d, dash] & \\
t_0 = \phi^2 \arrow[d, dash] & \mathbf{2}, & & t_1 = f \arrow[dl, dash] \arrow[dr, dash] ; & \\
\phi^3 \arrow[d, dash] & & \phi^5 \arrow[d, dash] & & \phi^6 \arrow[d, dash] \\
x & & x & & \phi^7 \arrow[d, dash] \\
& & & & x
\end{tikzcd}$$

here the occurrence 2 is rewritten. Then it is clear that ϕ^4 is a copy or *residual* of ϕ^1, that ϕ^5 and ϕ^7 are similarly residuals of ϕ^3, while ϕ^6 is not a residual of some occurrences in t_0 and is *created* by the rewriting of ϕ^2. Consider now Figure 1, which shows all possible derivations of $\phi(\psi(x))$ by the following program scheme:

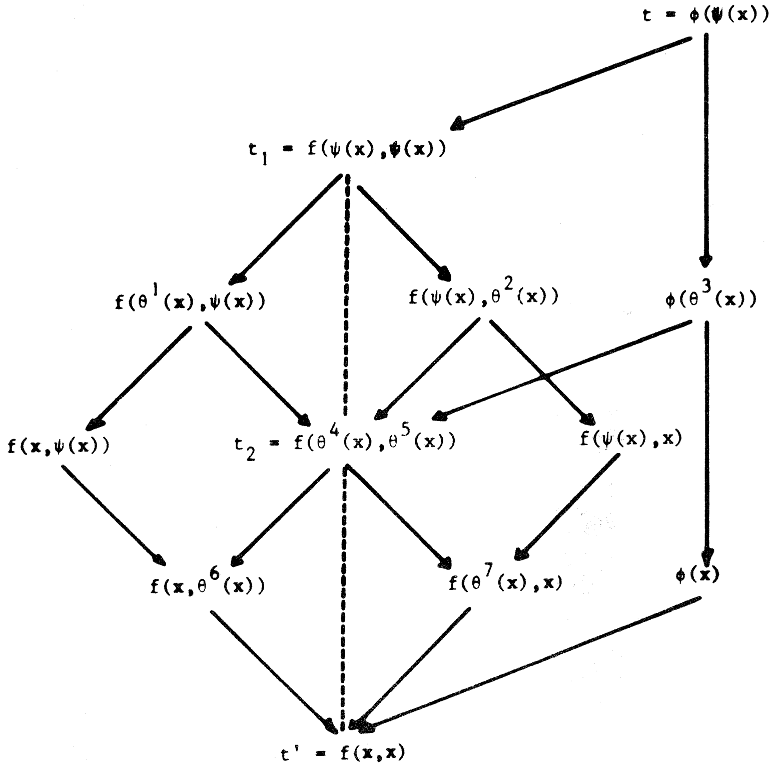


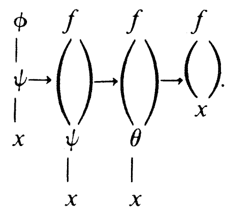
FIG. 1

$$\Sigma_6: \begin{cases} \phi(x) = f(x, x), \\ \psi(x) = \theta(x), \\ \theta(x) = x. \end{cases}$$

We first see that the diagram is a lattice, a central property in Vuillemin's work [25, 26]. We then see that the shortest derivation from $t = \phi(\psi(x))$ to $t' = f(x, x)$ is

$$\phi(\psi(x)) \rightarrow \phi(\theta(x)) \rightarrow \phi(x) \rightarrow f(x, x),$$

while the corresponding outside-in derivation in dag's would be



This derivation corresponds to the dotted line in Figure 1. That the two occurrences of ψ in t_1 should be shared is clear from Figure 1, since they are both residuals of the ψ in t. That θ^4 and θ^5 should be shared in t_2 is less clear, since they have no common ancestor along the dotted line. Sharing of θ^4 and θ^5 is justified by two observations:

(1) θ^4 and θ^5 are created "in the same way" by the two previously shared occurrences of ψ in t_1.

(2) They have indeed a common ancestor θ^3 along the derivation $t \rightarrow \phi(\theta^3(x)) \rightarrow t_2$.

By applying these two observations, we may hope to characterize the shared derivations in any diagram. In fact we can relate occurrences in different terms and define a notion of *family of occurrences* in two hopefully equivalent ways corresponding to the two observations above. Write $(c, t) \leq (c', t')$ if the occurrence c' in t' is a residual of the occurrence c in t by some derivation from t to t':

(1) If $(c, t) \leq (c', t')$, then the two occurrences are in the same family. Two occurrences created "in the same way" by two occurrences of the same family are of the same family. Here all ψ are residuals of the ψ in t and form a family, and all θ are created "in the same way" from the different ψ. (Vuillemin's labeling system [25, 26] is one way of formalizing the notion of family of occurrences according to this definition.)

(2) The families are defined by the symmetric and transitive closure of the relation $\leq$. Here all ψ are of the same family as before, and so are the θ for $\theta^3 \leq \theta^4 \leq \theta^7$, $\theta^3 \leq \theta^5 \leq \theta^6$, $\theta^1 \leq \theta^4$, and $\theta^2 \leq \theta^5$. (The term t in (c, t) is assumed here from the context: θ^4 abbreviates (θ^4, t_2).)

It is not true that all the θ are residuals of some unique occurrence. However, θ^3 can be considered as a canonical element in the family, since the derivation $\phi(\psi(x)) \rightarrow \phi(\theta(x))$ does nothing but create a member of the family as quickly as possible. The properties of this canonical element will be crucial below in studying *complete derivations*, which model the dag derivations and consist in deriving simultaneously all members of a family. In Figure 1, $t \rightarrow t_1 \rightarrow t_2 \rightarrow t'$ is a complete derivation.

The Lattice of Derivations. Figure 2 presents a more complicated situation. It shows the derivations of $I(K(I(x), y))$ by the program scheme

$$\Sigma_7: \begin{cases} I(x) = x, \\ K(x, y) = x. \end{cases}$$

(In the λ-calculus, take $I = \lambda x.x$ and $K = \lambda xy.x$). Figure 2 is not a lattice, and we have $I^1 \leq I^4 \leq I^7 \geq I^3 \geq I^2$, so that I^1 and I^2 should be of the same family and shared. This is clearly impossible. This situation is forbidden by Vuillemin [25] since he requires every

$$\begin{tikzcd}
 & & I^1(K(I^2(x),y)) \arrow[dll] \arrow[d] \arrow[drr] & & \\
K(I^3(x),y) \arrow[dr] \arrow[drrr] & & I^4(K(x,y)) \arrow[dl] \arrow[dr] & & I^5(I^6(x)) \arrow[dl, shift left] \arrow[dl, shift right] \\
 & K(x,y) \arrow[dr] & & I^7(x) \arrow[dl] & \\
 & & x & &
\end{tikzcd}$$

FIG. 2

right-hand-side term of the program to begin with a basic function symbol and to contain every symbol of the left-hand side. It is possible to preprocess the schemes and initial terms to satisfy these conditions, but the computation costs are drastically changed. Moreover, it is less clear how to do such preprocessing in the λ-calculus.

Four derivations reach $I^7(x)$. In two of them I^7 is a residual of I^1, and in the other two I^7 is a residual of I^2. Since I^1 and I^2 are different objects in the initial term, we can consider that the four derivations reach two different objects which are equal as terms by a syntactical coincidence. This is shown by using labels in Figure 3: This revised diagram is a lattice and the family definition is consistent. Several formalisms can be used to define the objects we manipulate. For technical convenience (and in particular for easy extension to the infinite case) we choose to use the notions of derivation classes and residuals: Two derivations reaching the same term with different residuals should be considered as different objects.

In fact considering only residuals of occurrences is not sufficient, since the phenomenon in Figure 2 can be hidden by using creations: Just add to Σ_7 the equation $\phi(x, y) = I(K(I(x), y))$ and start with $\phi(x, y) \to I(K(I(x), y))$. Then the occurrence of ϕ has no residual in any nonempty derivation. We therefore generalize Church's definition in Section 2 and define the residual d_2/d_1 of a *derivation* d_2 by a derivation d_1. We introduce the equivalence $d_1 \sim d_2$ iff $d/d_1 \sim d/d_2$ for any d. We give several equivalent definitions of $\sim$, and show in particular that $d_1 \sim d_2$ holds iff d_1 and d_2 perform the same operations in different orders. The equivalence is then used to define an ordering $d_1 \leq d_2$ if d_2 performs all operations of d_1. Derivation classes now form a lattice. We eventually relate $\sim$ to the classical standardization theorem [8] and to Vuillemin's labeling system [25].

The families, their canonical elements, and the complete derivations can be defined as before by considering equivalence classes of derivations instead of terms. Their study is the purpose of Section 3.

Minimality and Optimality Results. Minimality results are first shown in the Herbrand interpretation. For any finite approximation a of the infinite tree $\Sigma(t)$ computed by Σ on t, we construct a *least* derivation (with respect to $\leq$) d_a computing a. Ordering infinite derivations as well as finite ones, we also construct a least infinite derivation δ_A computing any infinite approximation A of $\Sigma(t)$. We show that the associated complete derivations $\bar{d}_a$ and $\bar{\delta}_A$ are outermost and are also least computations of a or A in the sublattice of complete derivations.

$$\begin{tikzcd}
 & I^{\alpha}(K(I^{\beta}(x),y)) \arrow[dl] \arrow[d] \arrow[dr] & \\
K(I^{\beta}(x),y) \arrow[d] \arrow[drr] & I^{\alpha}(K(x,y)) \arrow[dl] \arrow[d] & I^{\alpha}(I^{\beta}(x)) \arrow[dl] \arrow[d] \\
K(x,y) \arrow[dr] & I^{\alpha}(x) \arrow[d] & I^{\beta}(x) \arrow[dl] \\
 & x &
\end{tikzcd}$$

FIG. 3

Our next goal is to obtain optimality results from the minimality results. We first show that we can restrict our attention to those implementations in which physical computations simulate complete derivations (like dag's or delay rules [18, 25]): They are indeed the best possible implementations under a fairly general constraint. In these implementations the number of operations performed in a physical computation is proportional to the length $|\bar{d}|$ of the corresponding complete derivation $\bar{d}$. Hence we say that $\bar{d}$ is *optimal for computing a* if it computes a and satisfies $|\bar{d}|\leq|\bar{d}'|$ for any $\bar{d}'$ which computes a (and starts from the same term). We also notice that $|\bar{d}|$ is nothing but the number of elements of the set $\mathscr{F}(\bar{d})$ of families derived in $\bar{d}$. Hence $\mathscr{F}(\bar{d}) \subset \mathscr{F}(\bar{d}')$ implies $|\bar{d}|\leq|\bar{d}'|$. We say that $\bar{d}$ *is $\mathscr{F}$-optimal for computing a* if it computes a and satisfies $\mathscr{F}(\bar{d}) \subset \mathscr{F}(\bar{d}')$ for any $\bar{d}'$ which computes a. We show that minimality and $\mathscr{F}$-optimality coincide for outermost complete derivations, so that $\bar{d}_a$ defined above is $\mathscr{F}$-optimal and hence optimal for computing a. An advantage of the $\mathscr{F}$-optimality notion is that it also makes sense for infinite derivations. We show that $\bar{\delta}_A$ is $\mathscr{F}$-optimal for computing A. We finally show that, in the Herbrand interpretation, some correct and optimal derivations are generated by a simple computation rule.

The situation is more complex in arbitrary interpretations. An interpretation may contain "parallel functions" which require any computation rule to evaluate their arguments in parallel. In this case least and $\mathscr{F}$-optimal derivations may not exist, and optimal derivations have no simple characterizations. Several conditions were introduced in order to forbid parallel functions: sequentiality conditions [13, 25] or a stability condition [2]. We introduce here a *projectivity condition*: An interpretation I is projective if for any infinite tree A and any approximation α of $(I, \nu)A$, there exists a least $A_\alpha < A$ such that $\alpha \subset (I, \nu)A_\alpha$. Then computing α in (I, ν) is nothing but computing A_α in H, and all results obtained for the Herbrand interpretation apply: If P is a program and if $\alpha \subset P(t)$, then α has least and $\mathscr{F}$-optimal derivations. We show that sequential [13, 25] or stable [2] interpretations are projective.

1.3 PRELIMINARIES.

Let us give the formal definitions of the objects we manipulate. As in [20], we define terms as members of free algebras.

F-Algebras. Let $F = \{f_1, f_2, \ldots, f_m\}$ be a set of *function symbols*; each symbol f_i is given with its *arity* $\rho(f_i) \geq 0$. We denote by F^k the set of k-ary symbols in F. The nullary symbols are called *constants*. An *F-algebra* is an object $I = \langle D_I, f_1^I, \ldots, f_m^I\rangle$ where D_I is a set and each f_i^I is a mapping from $D_I^{\rho(f_i)}$ into D_I. A *morphism* of F-algebras I and I' is a mapping $\theta: D_I \to D_{I'}$ which preserves the operations f_i:

$$\forall k, \forall f_i \in F^k, \forall \mathbf{d} \in D_I^k, \quad \theta(f_i^I(\mathbf{d})) = f_i^I(\theta(\mathbf{d})).$$

The Free F-Algebra. Consider a denumerable set $V = \{x_1, x_2, \dots, x_k, \dots\}$ of *variable symbols* and let $V_k = \{x_1, x_2, \dots, x_k\}$. A *free F-algebra* $M(F, V)$ *generated by* V is an F-algebra containing V and such that for any F-algebra I, every mapping $\nu: V \rightarrow D_I$ extends in a unique way into a morphism $(I, \nu): M(F, V) \rightarrow I$. All free F-algebras generated by V are clearly isomorphic. The elements of $M(F, V)$ can be represented by well-formed formulas or trees: $f(x, g(x, y))$ with $\rho(f) = \rho(g) = 2$, or

```
     f
    / \
   x   g
      / \
     x   y
```

The interpretation of the function symbols is obvious, and $f^{M(F,V)}$ will be simply denoted by f:

—in well-formed formulas: $f: t_1, t_2, \dots, t_{\rho(f)} \rightarrow f(t_1, t_2, \dots, t_{\rho(f)})$.

—in trees:

```
                                   f
f: t1, t2, ..., tρ(f)  →        // \
                              t1  t2 ... tρ(f)
```

Intuitively, $(I, \nu)t$ is the value of t when the function symbols f are interpreted by the functions f^I and the variables x by the values $\nu(x)$.

A *substitution* is a mapping $\sigma: V \rightarrow M(F, V)$, which extends to a morphism $\sigma: M(F, V) \rightarrow M(F, V)$. We write $\sigma: \mathbf{t}/\mathbf{x}$, and for $t \in M(F, V)$, $\sigma t = t[\mathbf{t}/\mathbf{x}]$. The term σt is obtained by replacing all occurrences of x_i in t by the corresponding t_i.

Occurrences of Symbols in Terms. Let $\cdot$ denote concatenation and ϵ be the empty word. For $t \in M(F, \nu)$ and $s \in F \cup V$, *the set* $\mathscr{C}(s, t)$ *of occurrences of* s *in* t is defined by

(i) $t = x_i \in V$, $\mathscr{C}(x_i, x_i) = \{\epsilon\}$, $\mathscr{C}(s, x_i) = {}_{a}$ for $s \neq x_i$.

(ii) $t = f(\mathbf{t})$, $f \in F^k$, $\mathscr{C}(s, t) = \mathscr{C} \cup \bigcup_{j=1}^{k}((f, j) \cdot \mathscr{C}(s, t_j))$ with $\mathscr{C} = \{\epsilon\}$ if $s = f$ and $\mathscr{C} = \varnothing$ otherwise.

Example. $t = f(x_1, f(x_2, x_1))$. Then $\mathscr{C}(f, t) = \{\epsilon, (f, 2)\}$ and $\mathscr{C}(x_1, t) = \{(f, 1), (f, 2)(f, 2)\}$. The number j in each pair (f, j) represents the chosen argument of a function symbol f (see [7]); for convenience we also keep the function symbol, which is in fact redundant. Thus occurrences denote the path to a node in a tree (see Rosen [22]).

The *size* $\|t\|$ of a term t is the number of distinct occurrences in t.

LEMMA 1.3.1. *Let* $t'' = t[t'/x_i]$ *and* $c'' \in \mathscr{C}(s, t'')$. *Then either* $c'' = c \cdot c'$ *with* $c \in \mathscr{C}(x_i, t)$ *and* $c' \in \mathscr{C}(s, t')$, *or* $c'' \in \mathscr{C}(s, t)$.

PROOF. By induction on the length of c''. □

Recursive Program Schemes. Derivation and Residuals. Consider two disjoint sets $F = \{f_1, f_2, \dots, f_m\}$, of *basic function symbols*, and $\Phi = \{\phi_1, \phi_2, \dots, \phi_N\}$, of *unknown function symbols*, given with the arities $\rho(f_i) \geq 0$ and $\rho(\phi_i) \geq 0$. We write $M = M(F \cup \Phi, V)$ and $\mathscr{C}(\Phi, t) = \bigcup_{i=1}^{N} \mathscr{C}(\phi_i, t)$. All terms implicitely belong to M, and since we shall only be interested in occurrences in $\mathscr{C}(\Phi, t)$, we abbreviate $c \in \mathscr{C}(\phi, t)$ into c *in* t. Given c, c' in t, then c *contains* c', if c is a prefix of c', and c and c' are *disjoint*, if neither c contains c' nor c' contains c. An occurrence c in t is *outermost in* t if it is not contained in another c' in t.

A recursive program scheme Σ is a system of equations

$$\Sigma: \begin{cases} \phi_i(x_1, x_2, \dots, x_{\rho(\phi_i)}) = \tau_i, \\ i = 1, 2, \dots, N \text{ and } \tau_i \in M(F \cup \Phi, V_{\rho(\phi_i)}). \end{cases}$$

Since we consider a fixed program scheme Σ, we will omit explicit mention of Σ below. Let $t \in M$ and $c \in \mathscr{C}(\phi_i, t)$, and let y be an additional variable. There exists a unique decomposition $t = t_c[\phi_i(\mathbf{t})/y]$ such that $t_c \in M(F \cup \Phi, V \cup \{y\})$ and $\mathscr{C}(y, t_c) = \{c\}$. Then t is derived *immediately into t' by rewriting c*, written $t \xrightarrow{c} t'$, iff $t' = t_c[\tau_i[\mathbf{t}/\mathbf{x}]/y]$. Hence t' is obtained by replacing the occurrence c of the subterm $\phi_i(\mathbf{t})$ by the subterm $\tau_i[\mathbf{t}/\mathbf{x}]$, i.e. by replacing the function call by the corresponding body.

A *derivation* $d{:}t \xrightarrow{*} t'$ is a sequence $d{:}t \xrightarrow{c_1} t_1 \xrightarrow{c_2} t_2 \cdots \xrightarrow{c_k} t_k = t'$. The concatenation of $d{:}t \xrightarrow{*} t'$ and $d'{:}t' \xrightarrow{*} t''$ is written $d;d'$ and ϵ denotes the empty derivation. When t is clear from the context, a derivation is simply denoted by $d = c_1;c_2;\cdots;c_k$. Therefore c will denote either an occurrence or its derivation, unambiguously in each context.

Let $t \xrightarrow{c} t'$ as above. Given $c' \in \mathscr{C}(\phi_j, t')$, three cases are possible by Lemma 1.3.1:

(i) $c' = c\cdot c_1$ with $c_1 \in \mathscr{C}(\phi_j, \tau_i)$. Then *$c'$ is created by the derivation.*

(ii) $c' \in \mathscr{C}(\phi_j, t_c)$. Then *$c'$ in t' is a residual of c' in t.*

(iii) $c' = c\cdot c_1\cdot c_2$ with $c_1 \in \mathscr{C}(x_k, \tau_i)$ and $c_2 \in \mathscr{C}(\phi_j, t_k)$. Then *$c'$ in t' is a residual of $c\cdot(\phi_i, k)\cdot c_2$ in t.* (See Church [6] for the corresponding definitions in the λ-calculus.)

Example. Σ: $\phi(x) = f(x, \phi(x))$, $t = f(\phi(\phi(x)), \phi(x)) \xrightarrow{(f,1)} f(f(\phi(x), \phi(\phi(x))), \phi(x))$ with $t_{(f,1)} = f(y, \phi(x))$. Here $(f, 1)(f, 2)$ is created, $(f, 2)$ is a residual of $(f, 2)$, and $(f, 1)(f, 1)$ and $(f, 1)(f, 2)(\phi, 1)$ are residuals of $(f, 1)(\phi, 1)$.

Given $C \subset \mathscr{C}(\phi, t)$, let C/c denote the set of residuals of the elements of C by immediate derivation of c, and let c'/c abbreviate $\{c'\}/c$.

The notion of residuals is extended to derivations for $C \subset \mathscr{C}(\phi, t)$ by $C/\varnothing = C$ and $C/(c;d) = (C/c)/d$ and thus allows us to keep track of occurrences along derivations. Note that C/d is a set of occurrences in the final term of d.

2. *Equivalence of Derivations Modulo Permutations*

2.1 Preequivalent Derivations. We consider derivations between two expressions and write $d \equiv d'$ iff d and d' are two derivations of the form $t \xrightarrow{*} t'$. It is straightforward that $\equiv$ is an equivalence relation and forms a congruence with respect to concatenation of derivations. Furthermore, if $d \leq d'$ means that there is some d_1 such that $d;d_1 \equiv d'$, one can check easily that there are d and d' such that $d \leq d' \leq d$ and $d \not\equiv d'$.

We can have $d \equiv d'$ by some syntactical coincidences. For instance, in the above program scheme Σ_7, Figure 2, one has $d \equiv d'$ if

$$d{:}I(I(x)) \xrightarrow{\epsilon} I(x) \quad\text{and}\quad d'{:}I(I(x)) \xrightarrow{(I,1)} I(x)$$

or

$$d{:}K(K(x, y), y) \xrightarrow{\epsilon} K(x, y) \quad\text{and}\quad d'{:}K(K(x, y), y) \xrightarrow{(K,1)} K(x, y).$$

Note that in the above examples, derivations equivalent with respect to $\equiv$ do not yield the same set of residuals. Namely, one has $\epsilon/d = \varnothing$ and $\epsilon/d' = \{\epsilon\}$. In order to avoid these problems, we consider a first refinement of $\equiv$.

Notation 2.1.1. Let $\mathscr{D}_0(t)$ be the set of derivations starting from t.

Definition 2.1.1. Two derivations d and d' of $\mathscr{D}_0(t)$ are *preequivalent*, written $d \simeq d'$, iff $d \equiv d'$ and $c/d = c/d'$ for any c in t.

This relation is obviously an equivalence relation which is a refinement of $\equiv$. Moreover, preequivalence is a congruence with respect to concatenation. Furthermore, if $d \lesssim d'$ means that there is some d_1 such that $d;d_1 \simeq d'$, one can see by the following example that one can have d and d' such that $d \lesssim d' \lesssim d$ and $d \not\simeq d'$. Take

$$\Sigma_7{:}\begin{cases}\phi(x, y) = \phi(y, x),\\ \psi(x) = x;\end{cases}$$

$$d{:}\phi(\psi(x), \psi(x)) \xrightarrow{\epsilon} \phi(\psi(x), \psi(x));$$

$$d' = d;d.$$

Then $d;d \simeq d'$ and $d';d \simeq d$, but $d \neq d'$. The relation $\simeq$ has also been considered by Hindley [11] for the λ-calculus. A first interesting case of preequivalent derivations appears when we try to attach some meaning to the simultaneous derivations of a given set C of occurrences in a term t.

Definition 2.1.2. Given a set C of occurrences in a term t, the set $\mathscr{R}(t, C)$ of *derivations relative to* C is defined inductively by

$$o \in \mathscr{R}(t, C) \quad \text{(where } o \text{ is the empty derivation)},$$

$$d;c \in \mathscr{R}(t, C) \quad \text{if } c \in C/d \text{ and } d \in \mathscr{R}(t, C).$$

If $d \in \mathscr{R}(t, C)$ and $C/d = \emptyset$, then d is a *complete derivation relative to C*.

Intuitively, a derivation in $\mathscr{R}(t, C)$ does nothing but derivations of residuals of C. It is not clear whether all complete derivations relative to a given C end on the same term, and furthermore occurrences in t have the same set of residuals in the final term.

First, there is an easy case, i.e. when C is a set of disjoint occurrences, it is straightforward to check that no derivation relative to C can be infinite and that all complete derivations relative to C are preequivalent. Moreover, notice that the set c_1/c_2 of residuals of an occurrence c_1 by immediate derivation of c_2 is a set of disjoint occurrences. These two remarks allow us to consider the following permutation lemma.

LEMMA 2.1.1 (permutation lemma). *Let* $t \xrightarrow{c_1} t_1$ *and* $t \xrightarrow{c_2} t_2$, *and let* d_1 *and* d_2 *be two complete derivations relative to* c_1/c_2 *in* t_2 *and* c_2/c_1 *in* t_1. *Then* $c_1;d_2 \simeq c_2;d_1$.

PROOF. Let $d_1' = c_1;d_2$ *and* $d_2' = c_2;d_1$. Let c be in t. One has $d_1' \equiv d_2'$ and $c/d_1' = c/d_2'$ by inspecting the possible relative positions of c, c_1, and c_2. $\square$

THEOREM 2.1.1. *If C is a set of occurrences in a term t, then the lengths of all derivations relative to C have an upper bound. And if* d_1 *and* d_2 *are two complete derivations relative to C, then* $d_1 \simeq d_2$.

PROOF. If $c \in C$, let $\pi(c, C)$ be the nesting level of c in C (i.e. the number of prefixes of c in C). Let $k = k(C)$ be the maximal nesting level in C and $n_i(C)$ be the number of occurrences in C at level i. Consider the k-tuple $n(C) = \langle n_k(C), n_{k-1}(C), \ldots, n_1(C)\rangle$. Then one checks that $n(C/c) < n(C)$ for any c in C when $<$ is the lexicographic ordering on N^k. Thus all derivations in $\mathscr{R}(t, C)$ are bounded in length. Now suppose d_1 and d_2 are two complete derivations. By induction on $n(C)$, *we get* $d_1 \simeq d_2$. The only difficult case is when $d_1 = c_1;d_1'$ and $d_2 = c_2;d_2'$. Let d_1'' and d_2'' be complete derivations relative to c_1/c_2 and c_2/c_1. By the previous lemma, we have $C/(c_1;d_2'') = C/(c_2;d_1'') = C'$. Consider d_3 complete relative to C'. Then $d_1'';d_3$ and $d_2'';d_3$ are two complete derivations relative to C/c_2 and C/c_1. By induction, one has $d_1' \simeq d_2'';d_3$ and $d_2' \simeq d_1'';d_3$. Thus, for any c in t, one gets $c/(c_1;d_1') = c/(c_1;d_2'';d_3) = c/(c_2;d_1'';d_3) = c/(c_2;d_2')$ by use of the permutation lemma. Hence $d_1 \simeq d_2$. $\square$

This theorem corresponds to the property E of Curry and Feys [8] shown for the λ-calculus, and shows that the simultaneous derivation of any set of occurrences in a term makes sense. The order in which a set of occurrences is derived has no importance. Derivations are bounded and one always gets the same expression. Furthermore, residuals remain consistent.

Thus, if C is a set of occurrences in t, we write $t \xrightarrow{C} t'$, or t is *derived immediately into* t' *by rewriting* of C, if any complete derivation relative to C is of the form $t \xrightarrow{*} t'$. Corresponding derivations are written $D: t \xrightarrow{*} t'$ and are sequences

$$D: t \xrightarrow{C_1} t_1 \xrightarrow{C_2} t_2 \cdots \xrightarrow{C_n} t_n = t'$$

when $C_1, C_2, \ldots, C_n$ are sets of occurrences. Concatenation and residuals are written in the same way as before. (Residuals can be consistently defined by the previous preequivalence property.) Preequivalence is defined as in Definition 2.1.1. Let $\mathscr{D}(t)$ be the set of derivations of set of occurrences starting from t. Note that $\mathscr{D}_0(t)$ can be considered as a subset of $\mathscr{D}(t)$ by identifying rewritings of occurrences to rewritings of singletons. By convention, we use small d's for elements of $\mathscr{D}_0(t)$ and capital D's for elements of $\mathscr{D}(t)$.

Strictly speaking, equality of derivations in $\mathscr{D}(t)$ is equality step by step of the rewritten sets. However, it will be more convenient to identify steps of the form $t \xrightarrow{\varnothing} t$ with empty derivations. For instance,

$$t \xrightarrow{\varnothing} t \xrightarrow{C} t_1 = t \xrightarrow{C} t_1 \xrightarrow{\varnothing} t_1 = t \xrightarrow{C} t_1.$$

As previously, we write $D = C_1; C_2; \cdots; C_n$ when the initial term t is assumed by the context. Using the same notations for sets of occurrences and their derivations is justified by the fact that $t \xrightarrow{C_1} t_1$ and $t \xrightarrow{C_2} t_2$ are equal as derivations iff C_1 and C_2 are equal as sets.

With this convention, the empty derivation can be noted $\varnothing$. Furthermore, the length $|D|$ of a derivation D is the number of nonempty sets rewritten in D.

We can now extend the permutation lemma to sets of occurrences, and this corresponds to the lemma of "parallel moves" in [8].

LEMMA 2.1.2. *If C_1 and C_2 are sets of occurrences in a term t, then* $C_1;(C_2/C_1) \simeq C_2;(C_1/C_2)$.

PROOF. Obvious by considering complete derivations relative to $C_1 \cup C_2$. □

2.2 RESIDUALS OF DERIVATIONS AND EQUIVALENT DERIVATIONS. The previous lemma was used by Curry and Feys [8] for proving the Church-Rosser theorem, which states that, for any D_1 and D_2 in $\mathscr{D}(t)$, there are two derivations D_1' and D_2' such that $D_1;D_1' \equiv D_2;D_2'$. The situation can be illustrated by Figure 4, where each elementary square is an application of Lemma 2.1.2.

In fact, Lemma 2.1.2 shows that D_1' and D_2' can be chosen such that $D_1;D_1' \simeq D_2;D_2'$. But first, we use this lemma to define residuals of derivations and then an equivalence on derivations stronger than preequivalence.

Definition 2.2.1. If $D_1, D_2 \in \mathscr{D}(t)$, then *the derivation D_2/D_1 residual of D_2 by D_1* is defined inductively by

$$\varnothing/D_1 = \varnothing,$$
$$(D_2;\ C_2)/D_1 = (D_2/D_1);(C_2/(D_1/D_2)).$$

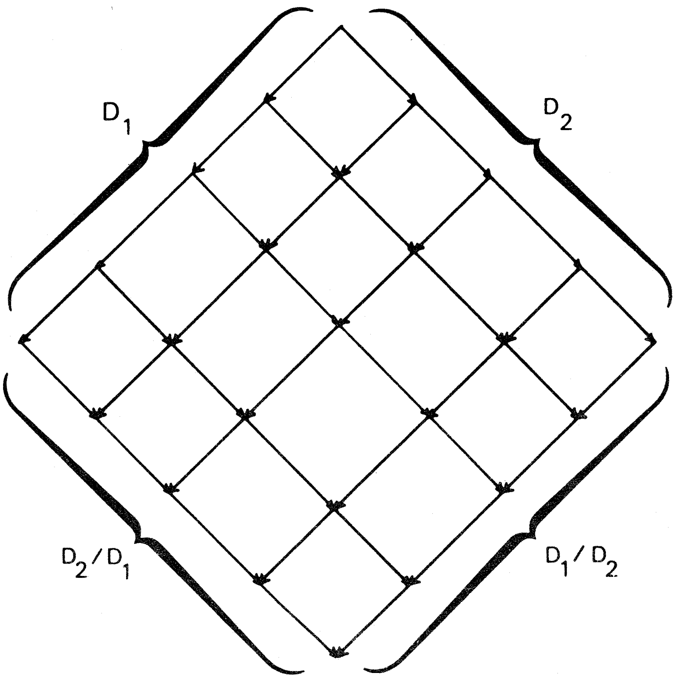


FIG. 4

Intuitively, the derivation D_2/D_1 is what remains to be done of D_2 after performing D_1 in order to close the Church-Rosser diagram, Figure 4. The fact that this definition makes sense and the following properties are easily proved by induction on $|D_1| + |D_2|$.

LEMMA 2.2.1. *If $D_1, D_2 \in \mathscr{D}(t)$, then*

(1) $D/\varnothing = D$;

(2) $(D_2;D_2')/D_1 = (D_2/D_1);(D_2'/(D_1/D_2))$;

(3) $D_2/(D_1;D_1') = (D_2/D_1)/D_1'$;

(4) $D_1;(D_2/D_1) \simeq D_2;(D_1/D_2)$.

Note that clause (4) implies the Church-Rosser property. We can now define the equivalence $\sim$ as the consistency of residuals of derivations.

Definition 2.2.2. Two derivations D_1 and D_2 in $\mathscr{D}(t)$ are *equivalent,* written $D_1 \sim D_2$, iff $D_1 \equiv D_2$ and $D/D_1 = D/D_2$ for any D in $\mathscr{D}(t)$.

Remember that equality of derivations means equality of each nonempty set of occurrences rewritten by each derivation. Furthermore, we need not worry about empty steps because, if $D/D_1 = D/D_2$ for any D, then the equality is true for any initial prefix of D and thus for any steps of D/D_1 and D/D_2.

It is obvious that $\sim$ is an equivalence relation and forms a congruence for concatenation. Moreover, if $D_1 \sim D_2$, then $D_1 \simeq D_2$ by considering any D which derives only one occurrence. Thus, the equivalence $\sim$ is a refinement of $\simeq$. The converse is not true. Take

$$\Sigma_8\colon \phi(x) = \phi(x),$$

$$d = \phi(x) \xrightarrow{\epsilon} \phi(x),$$

$$d' = d;d.$$

Then $d \simeq d'$, but $d'/d = d$ and $d'/d' = \varnothing$.

The relation $\sim$ is well behaved with respect to residuals.

LEMMA 2.2.2. $D_1 \sim D_2$ *iff* $\forall D \cdot D_1/D \simeq D_2/D$ *iff* $\forall D \cdot D_1/D \sim D_2/D$.

PROOF. Suppose D_1, D_2, and D are in $\mathscr{D}(t)$. Then, by Lemma 2.2.1, one has $D_1/D \equiv D_2/D$ iff $D_1;(D/D_1) \equiv D_2;(D/D_2)$. Suppose $D_1 \sim D_2$. Then $D_1 \equiv D_2$ and $D/D_1 = D/D_2$. Hence if t' is the final term of D and $D' \in \mathscr{D}(t')$, then $(D;D')/D_1 = (D;D')/D_2$ implies $D'/(D_1/D) = D'/(D_2/D)$ by Lemma 2.2.1. Therefore $D_1/D \sim D_2/D$ and $D_1/D \simeq D_2/D$ as $\sim$ is a refinement of $\simeq$. Suppose now that $D_1/D \simeq D_2/D$ for any D in $\mathscr{D}(t)$. Then $D_1 \equiv D_2$ by taking $D = \varnothing$. Now, by induction on $|D|$ and using Definition 2.2.1, one easily gets $D/D_1 = D/D_2$. □

We now strengthen clause (4) of Lemma 2.2.1.

LEMMA 2.2.3. *If $D_1, D_2 \in \mathscr{D}(t)$, then $D_1;(D_2/D_1) \sim D_2;(D_1/D_2)$.*

PROOF. Consider any D in $\mathscr{D}(t)$. We want to prove that $D/(D_1;(D_2/D_1)) = D/(D_2;(D_1/D_2))$. By induction on $|D|$, it is enough to prove for any set C of occurrences in t that $C/(D_1;(D_2/D_1)) = C/(D_2;(D_1/D_2))$ (which is already known by Lemma 2.2.1) and that $(D_1;(D_2/D_1))/C = D_1';(D_2'/D_1')$ and $(D_2;(D_1/D_2))/C = D_2';(D_1'/D_2')$ for some D_1' and D_2'. In fact we will take $D_1' = D_1/C$ and $D_2' = D_2/C$. By induction on $|D_1|+|D_2|$ and with Lemma 2.2.1, we only consider the case where $D_1 = C_1$ and $D_2 = C_2$ and C_1 and C_2 are two sets of occurrences in t. Let $C_1' = C_1/C$ and $C_2' = C_2/C$. Then, as $C_2/(C_1;(C/C_1)) = C_2/(C;C_1') = C_2'/C_1'$ by Lemma 2.1.2, one has $(C_1;(C_2/C_1))/C = C_1';(C_2'/C_1')$. Similarly $(C_2;(C_1/C_2))/C = C_2';(C_1'/C_2')$. □

The equivalence $\sim$ was given in terms of consistency of residuals of derivations. But two other characterizations of it can be proved.

PROPOSITION 2.2.1. $D_1 \sim D_2$ *iff* $D_1/D_2 = D_2/D_1 = \varnothing$.

PROOF. Suppose D_1 and D_2 are in $\mathscr{D}(t)$ and $D_1 \sim D_2$. Then by definition, we have $D/D_1 = D/D_2$ for any D of $\mathscr{D}(t)$. Thus $D_2/D_1 = D_2/D_2 = \varnothing$. Similarly $D_1/D_2 = D_1/D_1 = \varnothing$. Conversely, suppose $D_1/D_2 = D_2/D_1 = \varnothing$. By the previous lemma, one has $D//(D_1;(D_2/D_1)) = D/(D_2;(D_1/D_2))$. Hence, by an application of Lemma 2.2.1, we get $D/D_1 = D/D_2$. □

PROPOSITION 2.2.2. *Let $\approx$ be the least relation satisfying*

(1) *$C_1;(C_2/C_1) \approx C_2;(C_1/C_2)$ if C_1 and C_2 are sets of occurrences in a term t.*
(2) *$D_1;D_2;D_3 \approx D_1;D_2';D_3$ if $D_2 \approx D_2'$,*
(3) *$D_1 \approx D_2$ if $D_1 \approx D_3$ and $D_3 \approx D_2$ for some D_3.*

Then $D_1 \sim D_2$ iff $D_1 \approx D_2$.

PROOF. First, if $D_1 \approx D_2$, then it is easy to show $D_1 \sim D_2$ by Lemma 2.2.3, since the equivalence $\sim$ is a congruence for concatenation and is transitive. Conversely, let $D_1 \sim D_2$. Then $D_1/D_2 = D_2/D_1 = \varnothing$ by Proposition 2.2.1. Moreover, for any D_1 and D_2, one has obviously $D_1;(D_2/D_1) \approx D_2;(D_1/D_2)$. Now, as $D_1 \sim D_2$, we get $D_1 \approx D_2$ by the construction of Figure 4. □

Thus, the equivalence $\sim$ corresponds to the least congruence for the concatenation generated by the parallel moves of Lemma 2.1.2. In fact, it is possible to generate this congruence only by the permutation lemma, Lemma 2.1.1. We can now name the relation $\sim$ as the *equivalence* of *derivation modulo permutations.*

PROPOSITION 2.2.3. *The equivalence $\sim$ is left cancellative. That is, if $D;D_1 \sim D;D_2$, then $D_1 \sim D_2$.*

PROOF. Obvious from Proposition 2.2.1. □

2.3 THE LATTICE OF DERIVATION CLASSES. We now define a preorder on derivations.

Definition 2.3.1. $D_1 \leq D_2$ iff $\exists D\cdot(D_1;D) \sim D_2$.

PROPOSITION 2.3.1. *The relation $\leq$ is a preorder, i.e. it is reflexive and transitive, with $\sim$ for associated equivalence, i.e. $D_1 \leq D_2 \leq D_1$ iff $D_1 \sim D_2$. Moreover $D_1 \leq D_2$ holds iff $D_1/D_2 = \varnothing$.*

PROOF. $D \leq D$ since $D \sim D$. Assume $D_1 \leq D_2 \leq D_3$. Then there are D_1' and D_2' such that $D_1;D_1' \sim D_2$ and $D_2;D_2' \sim D_3$. Hence $D_1;D_1';D_2' \sim D_3$ and $D_1 \leq D_3$. Now suppose $D_1 \leq D_2$. Then $D_1;D \sim D_2$ for some D. Thus Lemma 2.2.1 and Proposition 2.2.1 imply $(D_1;D)/D_2 = (D_1/D_2);D' = {}_{\text{a}}$. Therefore $D_1/D_2 = \varnothing$.

Conversely, if $D_1/D_2 = {}_{\text{a}}$, one has $D_1;(D_2/D_1) \sim D_2$ by Lemma 2.2.3, i.e. $D_1 \leq D_2$. Now if $D_1 \leq D_2 \leq D_1$, we have $D_1/D_2 = D_2/D_1 = \varnothing$ and thus $D_1 \sim D_2$ by Proposition 2.2.1. □

Notation 2.3.1. Let $[D]$ be the equivalence class of D and let $[\mathscr{D}(t)]$ be the set $\mathscr{D}(t)/\sim$.

THEOREM 2.3.1. *The set $[\mathscr{D}(t)]$ with the quotient ordering $\leq$ is an upper semilattice. Two elements $[D_1]$ and $[D_2]$ have the l.u.b.* (*least upper bound*)

$$[D_1;(D_2/D_1)] = [D_2;(D_1/D_2)].$$

PROOF. One has $D_1 \leq D_1;(D_2/D_1)$ and $D_2 \leq D_2;(D_1/D_2)$. Furthermore, let $D_1 \leq D$ and $D_2 \leq D$. By Proposition 2.3.1, we have $D_1/D = D_2/D = {}_{\text{a}}$. Lemma 2.2.1 implies $(D_1;(D_2/D_1))/D = (D_1/D);D' = D'$ where $D' = (D_2/D_1)/(D/D_1)$. But $D' = D_2/(D_1;(D/D_1)) = D_2/(D;(D_1/D)) = (D_2/D)/(D_1/D)$ by Lemmas 2.2.1 and 2.2.3 and Definition 2.2.2. Hence $D' = \varnothing$ and $D_1;(D_2/D_1) \leq D$ by Proposition 2.3.1. □

It is also shown in [3] that $[D_1]$ and $[D_2]$ have a g.l.b. (greatest lower bound). Note that Theorem 2.3.1 can be expressed as a Church-Rosser property way by "for any D_1 and D_2 in $\mathscr{D}(t)$, there is a minimum $[D]$ such that $D_1 \leq D$ and $D_2 \leq D$."

2.4 STANDARD DERIVATIONS. Standard derivations, which work in an outside-in way and (by convention) from left to right for disjoint occurrences, are introduced in [8]. The standardization theorem in [8] allows any reduction to be standardized. Although there can be several standard derivations between two given terms, each equivalence class contains a unique standard derivation. This will be our improved standardization theorem.

Definition 2.4.1. For $t = s(\mathbf{t})$ and $d_i{:}t_i \rightarrow t_i'$, let $s(\mathbf{d}) = ((s, 1)\cdot d_1)$; $((s, 2)\cdot d_2)$; $\cdots$; $((s, k)\cdot d_k)$ where $k = \rho(s)$. The *derivation* d is *standard* if either $d = \epsilon;d'$ and d' is standard, or $d = s(\mathbf{d})$ and d_i is standard for all i.

Notation 2.4.1. The set C of occurrences is *internal* if $\epsilon \notin C$. The derivation D is internal if it derives only internal sets. Let $\epsilon^n = \epsilon;\ \epsilon;\ \cdots;\ \epsilon$.

LEMMA 2.4.1 *For any C, there are n and C' internal such that $n \geq 0$ and $c \sim \epsilon^n;C'$.*

PROOF. By Theorem 2.1.1, since we can consider any outside-in complete derivation of C. □

THEOREM 2.4.1. *There is a unique standard derivation in each equivalence class of* $[\mathscr{D}(t)]$.

PROOF. *Existence* (Mitschke [17]). Consider any derivation $D{:}t \xrightarrow{*} t'$. With Lemma 2.4.1, we can write $D = C_1;\ C_2;\ \cdots;\ C_n$ where either $C_i = \{\epsilon\}$ or C_i is internal for all i. Let $l(D)$ be maximal such that $D = D';\epsilon^{l(D)};D''$ with D'' internal. Let $n(D)$ be the number of internal C_i preceding some ϵ-step. Formally $n(D)$ is the number of i such that C_i is internal and $\exists j,\ 1 \le i < j \le n$, such that $C_j = \{\epsilon\}$. Consider the triple $\langle \|t'\|, n(D), l(D)\rangle$. We have two cases. First, $D = \epsilon^{l(D)};D_2$ with D_2 internal. By induction on $\|t'\|$, we get $D_2 \sim D_2'$ where D_2' is standard and $D \sim \epsilon^{l(D)};D_2'$. Second, $D = D_1;C;\epsilon^{l(D)};D_2$ with C and D_2 internal and $l(D) > 0$. But C cannot create ϵ and $C;\epsilon$ is a complete derivation of $C \cup \{\epsilon\}$. Hence $C;\epsilon \sim \epsilon^p;C'$ with C' internal by Lemma 2.4.1 and $p > 0$ since $\epsilon \in C \cup \{\epsilon\}$. Therefore $D \sim D'$ with $D' = D_1;\epsilon^p;C';\ \epsilon^{l(D)-1};D_2$. Then $n(D') < n(D)$ if $C' = \varnothing$ or $l(D) = 1$, and $n(D') = n(D),\ l(D') < l(D)$ if $C' \neq \varnothing$ and $l(D) > 1$. By induction $D' \sim D''$ where D'' is standard.

Uniqueness. Let $d,d'{:}t \rightarrow t'$ be standard such that $d \sim d'$. We prove $d = d'$ by induction on $\langle |d| + |d'|, \|t'\|\rangle$. The induction works easily when $d = s(\mathbf{d}),\ d' = s(\mathbf{d}')$ or $d = \epsilon;d_1,\ d' = \epsilon;d_1'$. Now suppose $d = \epsilon;d_1$ and $d' = s(\mathbf{d}')$. Then $s = \phi \in \Phi$ and $\epsilon/d' = \{\epsilon\} \neq \varnothing$ which contradicts $d \simeq d'$ and thus $d \sim d'$. □

2.5 LABELED DERIVATIONS. We use Vuillemin's labeling system [25] to mark occurrences along derivations. (See [15] for the λ-calculus.) Given an alphabet E, let the set E^* of words on E be set of labels. A *labeling* μ *of a term* t is a mapping $\mu{:}\mathscr{C}(\Phi, t) \rightarrow E^*$. A labeling ν of the program Σ is an N-tuple $(\nu_1, \nu_2, \ldots, \nu_N)$ of labelings ν_i of the τ_i. Given a labeled term (t, μ) and an immediate derivation $d{:}t \xrightarrow{c} t'$ with $c \in \mathscr{C}(\phi_i, t)$, the corresponding step of labeled derivation $d{:}(t, \mu) \xrightarrow{c} (t', \mu')$ is defined by

$$\mu(c') = \mu(c_1') \quad \text{if } c' \in c_1'/c,$$
$$\mu(c') = \mu(c)\nu_i(c_1') \quad \text{if } c \text{ creates } c' \text{ and } c' = c \cdot c_1'.$$

For example, let $t = \phi(\phi(x))$ and $\Sigma{:}\phi(x) = f(x, \phi(\phi(x)))$ and let us write labels as exponents of occurrences in terms. Suppose $(\Sigma, \nu){:}\phi(x) = f(x, \phi^a(\phi^b(x)))$ and $(t, \mu) = \phi^c(\phi^d(x))$. Then $(t, \mu) \xrightarrow{\epsilon} f(\phi^d(x), \phi^{ca}(\phi^{cb}(\phi^c(x))))$.

Notation 2.5.1. We write $d \equiv_\mu d'$ if d and d' are of the form $d, d'{:}(t, \mu) \xrightarrow{*} (t', \mu')$.

The permutation lemma, Lemma 2.1.1, also holds for labeled derivations.

LEMMA 2.5.1. *If* c_1, c_2 *are in* t, *then* $c_1;(c_2/c_1) \equiv_\mu c_2;(c_1/c_2)$ *for all* μ.

PROOF. By inspection of cases. □

PROPOSITION 2.5.1. *If* $d_1 \sim d_2$, *then* $d_1 \equiv_\mu d_2$ *for all* μ.

PROOF. Obvious from Lemma 2.5.1 and Proposition 2.2.2 built on Lemma 2.1.1 instead of Lemma 2.1.2. □

Definition 2.5.1. Two labels α and β are incompatible, written $\alpha|\beta$, iff there are no γ and δ such that $\alpha\gamma = \beta\delta$. A labeling μ of a term t is *consistent* if the labels $\mu(c)$ and $\mu(c')$ are incompatible for any pair c and c' of nested occurrences in t. A labeling ν of Σ is consistent if all the ν_i are consistent.

With our notation, if μ is consistent and $c < c'$ then $\mu(c)|\mu(c')$. We remark that $\alpha|\beta$ implies $\alpha\gamma|\beta\delta$ and $\gamma\alpha|\delta\beta$ for any γ and δ. The consistency of labelings is a technical condition used to connect labeled terms and the equivalence $\sim$.

LEMMA 2.5.2. *If* μ, ν *are consistent and if* $(t, \mu) \xrightarrow{*} (t', \mu')$, *then* μ' *is consistent.*

PROOF. By inspection of cases in one step of labeled derivation. □

PROPOSITION 2.5.2. *If* μ, ν *are consistent, then for any* d *there is one and only one standard* d' *such that* $d \equiv_\mu d'$.

PROOF. By Theorem 2.4.1, for any d, there is a standard d' such that $d \sim d'$. Hence $d \equiv_\mu d'$ by Proposition 2.5.1. Now suppose that d and d' are standard such that $d \equiv_\mu d'$. More precisely, let $d, d'{:}(t, \mu) \xrightarrow{*} (t', \mu')$. We prove $d = d'$ by induction on $\langle |d| + |d'|, \|t\|\rangle$. If $d = s(\mathbf{d}),\ d' = s(\mathbf{d}')$ or $d = \epsilon;d_1,\ d' = \epsilon;d_1'$, the induction works easily by using

Lemma 2.5.2. Assume now that $d = \epsilon; d_1$ and $d' = s(\mathbf{d}')$. We have $s = \phi \in \Phi$ and $t = \phi(\mathbf{t})$, $t' = \phi(\mathbf{t}')$. Along d', we get $\mu'(\epsilon) = \mu(\epsilon)$. But $d{:}(\phi(\mathbf{t}), \mu) \xrightarrow{\epsilon} (\tau[\mathbf{t}/\mathbf{x}], \mu_1) \xrightarrow{*} (t', \mu')$. Consider any c_1 in $\tau[\mathbf{t}/\mathbf{x}]$. Either $c_1 \in c/\epsilon$ with c internal to some t_i and $\mu_1(c_1) = \mu(c)|\mu(\epsilon)$ by the consistency hypothesis, or c_1 is created by ϵ and $\mu(\epsilon) \leq \mu(c_1)$. Therefore there is no c in $\tau[\mathbf{t}/\mathbf{x}]$ such that $\mu_1(c) \leq \mu(\epsilon)$. Hence no occurrence in t' can have $\mu(\epsilon)$ for a label, which contradicts $\mu(\epsilon) = \mu'(\epsilon)$. □

THEOREM 2.5.1. *$D_1 \sim D_2$ iff $D_1 \equiv_\mu D_2$ for all μ, iff $D_1 \equiv_\mu D_2$ for some consistent μ, ν.*

PROOF. Denote the above clauses (1), (2), and (3), respectively. Then (1) $\Rightarrow$ (2) by Proposition 2.5.1, (2) $\Rightarrow$ (3) by instantiation, and (3) $\Rightarrow$ (1) by Theorem 2.4.1 and Lemma 2.5.2. □

COROLLARY 2.5.1. *$D_1 \leq D_2$ iff, for all μ, ν (or for some consistent μ, ν), one has $D_1{:}(t, \mu) \xrightarrow{*} (t_1, \mu_1)$, $D_2{:}(t, \mu) \xrightarrow{*} (t_2, \mu_2)$, and $(t_1, \mu_1) \xrightarrow{*} (t_2, \mu_2)$.*

For the case of the λ-calculus, all theorems and propositions of Section 2 are valid (see [15]). However some proofs, such as Theorem 2.1.1, are more complicated.

3. *Familes of Occurrences and f-Complete Derivations*

3.1 FAMILY OF OCCURRENCES. In order to compute in an "optimal" way, we must share duplicated objects (see [18, 25]). Sharing residuals is certainly necessary, but not sufficient, as shown by the following example.

Let $\Sigma{:}\phi_1(x) = f(x, x)$, $\phi_2(x) = g(\phi_2(x))$. Let $d{:}t = \phi_1(\phi_2(x)) \xrightarrow{\epsilon} f(\phi_2(x), \phi_2(x)) \xrightarrow{C} f(g(\phi_2(x)), g(\phi_2(x))) = t'$ where $C = \{(f, 1), (f, 2)\}$. The two occurrences $c_1 = (f, 1)(g, 1)$ and $c_2 = (f, 2)(g, 1)$ in t' are not residuals of the same occurrence in a term appearing in d. But it is easy to see that they have to be shared. This can be approached in two ways. The first approach used by Vuillemin [25] is to share them because they have the same label in any labeling of d. Intuitively, residuals of shared occurrences are shared but occurrences are also shared which are created in the same way by shared occurrences. The second approach, used here, does not involve labels. There is a permutation d' of d, namely

$$d'{:}t = \phi_1(\phi_2(x)) \xrightarrow{c'} \phi_1(g(\phi_2(x))) \xrightarrow{\epsilon} f(g(\phi_2(x)), g(\phi_2(x))) = t'$$

such that $c' = (\phi_1, 1)$. Then c_1, c_2 are indeed residuals of a unique occurrence $c = (\phi_1, 1)(g, 1)$ in $\phi_1(g(\phi_2(x)))$, and therefore can be shared in d. The two approaches will be shown equivalent. Thus the behavior of labels will be explained within the classical formalism of residuals.

Notation 3.1.1. From now on, the derivation-occurrence pair (D, c) is an abbreviation for $D{:}t \xrightarrow{*} t'$ and c in t'. We also consider pairs (D, C) where C is a set of occurrences in t'.

Definition 3.1.1. We consider the relations $\leq$ and $\sim$ (read "has for residual" and "belong to the same family as") over pairs (D, c) defined as follows:

$$(D_1, c_1) \leq (D_2, c_2) \text{ if } \exists D, (D_1; D) \sim D_2 \text{ and } c_2 \in c_1/D;$$
$$(D_1, c_1) \sim (D_2, c_2) \text{ if } (D_1, c_1) \leq (D_2, c_2) \text{ or } (D_2, c_2) \leq (D_1, c_1), \text{ or}$$
$$\exists (D, c), (D_1, c_1) \sim (D, c) \sim (D_2, c_2).$$

Hence the second relation is the symmetric and transitive closure of the first one. It is straightforward to check that $\leq$ is a preorder and $\sim$ is an equivalence.

PROPOSITION 3.1.1. *The two above relations are consistent with the equivalence of derivations with respect to permutations. Namely, if $D_1 \sim D_1'$ and $D_2 \sim D_2'$ we have*

$$(D_1, c_1) \leq (D_2, c_2) \textit{ iff } (D_1', c_1) \leq (D_2', c_2) \textit{ and}$$
$$(D_1, c_1) \sim (D_2, c_2) \textit{ iff } (D_1', c_1) \sim (D_2', c_2).$$

PROOF. Obvious since $(D_1'; D) \sim D_2'$ when $D_1 \sim D_1'$, $D_2 \sim D_2'$ and $(D_1; D) \sim D_2$. □

PROPOSITION 3.1.2.

(1) *$(D_1, c_1) \leq (D_2, c_2)$ iff $D_1 \leq D_2$ and $c_2 \in c_1/(D_2/D_1)$.*

(2) *If $D_1 \sim D_2$ and if $(D_i, c_i) \leq (D, c)$ for $i = 1,2$, then $c_1 = c_2$.*

(3) *If* $(D_1, c_1) \le (D_2, c_2)$ *and* $D_1 \le D \le D_2$, *then there is a unique* c *such that* $(D_1, c_1) \le (D, c) \le (D_2, c_2)$.

(4) *If* $(D_i, c_i) \le (D, c)$ *for* $i = 1, 2$, *then there are a unique* c' *and some* D' *such that* $D' \sim$ *l.u.b.* $([D_1],[D_2])$ *and* $(D_i, c_i) \le (D', c') \le (D, c)$.

Proof. Obvious once we remark that $D_1 \sim D_2$ implies $c/D_1 = c/D_2$ and that we have by definition $c_1 = c_2$ if $c \in c_1/D$ and $c \in c_2/D$. □

Hence $(\varnothing, c_1) \sim (D, c)$ iff $c \in c_1/D$, i.e. occurrences are in the family of an initial occurrence iff they are residuals of it. But also, created occurrences can be in the same family. We will indicate what is the meaning of the family relation for them. We show that, in a given family, there is only one (d, c) such that d "generates" c. This element is exactly the one for which $|d|$ is minimum. Furthermore, two occurrences will be in the same family iff they are created in the same way.

Definition 3.1.2. The *derivation* d *generates* (d, c) iff $d = \varnothing$ or $d = d_1;c_1$ with d_1 generating (d_1, c_1) and c_1 creating c.

We now adopt some technical but convenient notations concerning derivations in subterms.

Notation 3.1.2. Given c and $d = c_1; c_2; \cdots; c_k$, we use the notation $c \cdot d$ for denoting the derivation $c \cdot c_1; c \cdot c_2; \cdots; c \cdot c_k$ (when it exists). Let $(d, c)*(d', c') = ((d;(c \cdot d')), (c \cdot c'))$. Let $(d', c')\backslash(d, c) = (d'', c'')$ iff $(d, c) = (d', c')*(d'', c'')$.

Note that if d generates (d, c) and if $(d;c) = (c';d')$, then $d = c' \cdot d_1$ and $c = c' \cdot c_1$ for some d_1 and c_1. Note also that $(d, c)*(d', c')$ is defined iff d and d' are of the form $t_1 \overset{*}{\to} t_2$ and $t_1' \overset{*}{\to} t_2'$ and t_1' is the subterm of t_2 at occurrence c. Furthermore, the operation $*$ is associative. Moreover, if c creates c' and if $d' = (d;c)$, then $(d, c)\backslash(d', c')$ is defined (see preliminaries). We can now express an operation on any pair (d, c) which consists in extracting from d the steps contributing to the creation of c.

Definition 3.1.3. The *canonical representative* of (d, c), written $(d, c)_0$, is inductively defined by

$$\begin{aligned}
&(\varnothing, c)_0 = (\varnothing, c),\\
&(d, c)_0 = (d', c_1)_0 \quad \text{if } d = d';c' \text{ and } c \in c_1/c',\\
&(d, c)_0 = (d', c')_0*((d', c')\backslash(d, c)) \text{ if } d = d';c' \text{ and } c' \text{ creates } c.
\end{aligned}$$

We now prove that this definition is consistent with the equivalence of derivations modulo permutations.

Lemma 3.1.1. *If* c_1 *and* c_2 *are in a term* t, *then, for any* c, *one has* $((c_1;(c_2/c_1)), c)_0 = ((c_2;(c_1/c_2)), c)_0$.

Proof. By inspection of cases. □

This lemma is stronger than the permutation lemma, Lemma 2.1.1, because not only residuals but also created occurrences behave well with respect to permutation.

Notation 3.1.3. The *generator* gen(d, c) *of* c *by* d is defined inductively for any pair (d, c) by

$$\begin{aligned}
&\text{gen}(\varnothing, c) = c,\\
&\text{gen}((d';c'), c) = \text{gen}(d', c_1) \quad \text{if } c \in c_1/c',\\
&\text{gen}((d';c'), c) = \text{gen}(d', c') \quad \text{if } c' \text{ creates } c.
\end{aligned}$$

Thus, if $d{:}t \overset{*}{\to} t'$, then gen$(d, c)$ is in t.

Lemma 3.1.2. *For any pair* (d, c), *if* $d = (d_1;d_2)$, *then*

(1) $(d, c)_0 = (d_1, gen(d_2, c))_0 * ((\varnothing, gen(d_2, c))\backslash(d_2, c)_0)$,

(2) $(d, c)_0 = ((d_1;d_2'),c')_0$ *if* $(d_2, c)_0 = (d_2', c')$.

Proof. Obvious by induction on $|d_2|$ and application of Definition 3.1.3 and Notation 3.1.3. □

Proposition 3.1.3. *If* $d_1 \sim d_2$, *then* $(d_1, c)_0 = (d_2, c)_0$.

Proof. By the remark following Proposition 2.2.2, it is enough to consider the case when $d_1 = (d';c_1;(c_2/c_1);d'')$ and $d_2 = (d';c_2;(c_1/c_2);d'')$. Then Lemmas 3.1.1 and 3.1.2 give the result. □

Now, we can speak consistently of $(D, c)_0$, where each step of D consists in the simultaneous derivation of a set of occurrences, since all derivations designated by D are equivalent. Furthermore, canonical representatives are consistent for the family relation.

PROPOSITION 3.1.4 *If* $(D_1, c_1) \sim (D_2, c_2)$, *then* $(D_1, c_1)_0 = (D_2, c_2)_0$.

PROOF. It is enough to consider $(D_1, c_1) \leq (D_2, c_2)$. Then by definition, there is D such that $D_1;D \sim D_2$ and $c_2 \in c_1/D$. Hence, by Lemma 3.1.2 and Proposition 3.1.3, one gets

$$(D_2, c_2)_0 = ((D_1;D), c_2)_0 = (D_1, c_1)_0. \qquad \square$$

We now prove the converse of Proposition 3.1.4 by showing $(D, c) \sim (D, c)_0$ for any (D, c).

Definition 3.1.4. If $D{:}t \xrightarrow{*} t'$ and c is in t, then c and D are *disjoint* iff $D = (C_1;\ C_2;\ \cdots;\ C_n)$ and c, C_i are disjoint for any i such that $1 \leq i \leq n$. Thus $c/D = \{c\}$.

LEMMA 3.1.3. *If* c, D *are disjoint and* $c_1 \in c/D_1$, *then* c_1 *and* D/D_1 *are disjoint. Moreover* $c_1/(D/D_1) = \{c_1\}$ *and* $c_1 \in c/(D_1/D)$.

PROOF. By induction on $|D|$. The only nontrivial case is when $D = C$. By hypothesis we have c and c' disjoint for any c' in C. As residuals of disjoint occurrences are still disjoint, one has c_1 and C/D_1 disjoint too. Thus $c_1/(C/D_1) = \{c_1\}$, i.e. $c_1 \in c/(D_1;(C/D_1))$. But since $D_1;(C/D_1) \sim C;(D_1/C)$ and since residuals are consistent with respect to permutations, we have $c_1 \in c/(D;(D_1/C))$, i.e. $c_1 \in c/(D_1/C)$. $\square$

PROPOSITION 3.1.5. *For any* (D, c), *there is some* (D_1, c_1) *such that* $(D, c) \leq (D_1, c_1)$ *and* $(D, c)_0 \leq (D_1, c_1)$. *Thus* $(D, c)_0 \sim (D, c)$.

PROOF. Consider any singleton derivation d equivalent to D. Then $(D, c)_0 = (d, c)_0$ by Proposition 3.1.4. Use an induction on $|d|$. If $d = \varnothing$, then $(d, c)_0 = (d, c) \leq (d, c)$. Now suppose $d = c';d'$. By induction, there is (D'_1, c'_1) such that $(d', c)_0 \leq (D'_1, c'_1)$ and $(d', c) \leq (D'_1, c'_1)$. Let $(d_0, c_0) = (d, c)_0$ and $(d'_0, c'_0) = (d', c)_0$. If we use Lemma 3.1.2, there are two cases.

First, if c' creates gen(d', c), then $d_0 = c';d'_0$ and $c_0 = c'_0$. Therefore, if $D_1 = c';D'_1$ *and* $c_1 = c'_1$, we have $(d_0, c_0) \leq (D_1, c_1)$ and $(d, c) \leq (D_1, c_1)$.

Second, if c' does not create gen(d', c), then gen$(d', c) \in c_2/c'$ for some c_2. Let $c'_2 =$ gen(d', c). By Lemma 3.1.2, we have $d'_0 = c'_2 \cdot d''_0$, $c'_0 = c'_2 \cdot c''_0$ and $d_0 = c_2 \cdot d''_0$, $c_0 = c_2 \cdot c''_0$. Hence, as c_2/c' is a set of disjoint occurrences, the derivations $c''_2 \cdot d''_0$ are disjoint for all c''_2 in c_2/c'. But these derivations form d_0/c'. Therefore $d'_0 \leq d_0/c'$. Moreover c'_0 and $d_3 = (d_0/c')/d'_0$ are disjoint. Hence $c'_0 \in c'_0/d_3$. But by the permutation lemma, we have $c';d'_0;d_3 \sim d_0;(c'/d_0)$. Thus $((d_0;(c'/d_0)), c'_0)_0 = ((c';d'_0), c'_0)_0 = (d_0, c_0)$ by Lemma 3.1.2. But as d_0 generates (d_0, c_0), we have $(d_0, c_0)_0 = (d_0, c_0)$. Then again by Lemma 3.1.2, we get $c'_0 \in c_0/(c'/d_0)$. Let $D''_1 = D'_1/d'_0$. Since $(d'_0, c'_0) \leq (D'_1, c'_1)$, we have $c'_1 \in c'_0/D''_1$ by Proposition 3.1.2 and therefore $c'_1 \in c'_0/(D''_1/d_3)$ by Lemma 3.1.3 since c'_0 and d_3 are disjoint. Again by Lemma 3.1.3, we get $c'_1 \in c'_1/(d_3/D'_1)$. In short, if $c_1 = c'_1$ and $D_1 = c';D'_1;(d_3/D''_1)$, then $(d, c) \leq (D_1, c_1)$ and $(d_0, c_0) \leq (D_1, c_1)$. $\square$

THEOREM 3.1.1. $(D_1, c_1) \sim (D_2, c_2)$ *iff* $(D_1, c_1)_0 \sim (D_2, c_2)_0$.

PROOF. Obvious from Propositions 3.1.4 and 3.1.5. $\square$

PROPOSITION 3.1.6. *For any* (D, c), *the element* $(D, c)_0 = (d_0, c_0)$ *is the only one in the family of* (D, c) *such that* d_0 *generates* (d_0, c_0). *Furthermore, it is the only one in the family of* (D, c) *such that* $|d_0|$ *is minimum.*

PROOF. By definition, if d_0 generates (d_0, c_0), then $(d_0, c_0)_0 = (d_0, c_0)$. Thus, Theorem 3.1.1 gives the first part. Furthermore, if $(d_0, c_0) = (d, c)_0$ and $(d_0, c_0) \neq (d, c)$, then $|d_0| < |d|$ by Definition 3.1.3. Again, Theorem 3.1.1 gives the second part. $\square$

The next proposition gives the key property for showing that the complete derivations (or derivations with sharing) form a sublattice of the lattice of derivations. This will be crucial in proving the optimality results. (The remark following Proposition 3.1.2 was indeed a special case of this proposition.)

PROPOSITION 3.1.7. *Let* $(D, c)_0 = (d_0, c_0)$. *Then* $d_0 \leq D$ *iff* $(d_0, c_0) \leq (D, c)$.

PROOF. Corollary of Proposition 3.1.2 (clause (1)) and Propositions 3.1.5 and 3.1.2 (clause (4)). $\square$

We now show the connection between families and labels.

Definition 3.1.5. Given any term t, the labeling μ is *elementary* if $\mu:\mathscr{E}(\phi, t) \to E$, i.e. the range of μ is the set of letters E.

LEMMA 3.1.4. *For any label α, if μ and v are injective elementary labelings, there is at most one (d, c) such that $d:(t, u) \overset{*}{\to} (t', \mu')$, d generates (d, c), and $\mu'(c) = \alpha$.*

PROOF. By induction on the length of α. □

THEOREM 3.1.2. *Let $D_1 = (t, \mu) \overset{*}{\to} (t_i, \mu_i)$ for $i = 1, 2$. Then $(D_1, c_1) \sim (D_2, c_2)$ iff $\mu_1(c_1) = \mu_2(c_2)$ when μ, v are injective elementary iff $\mu_1(c_1) = \mu_2(c_2)$ for all μ.*

PROOF. First $(D_1, c_1) \sim (D_2, c_2)$ implies $\mu_1(c_1) = \mu_2(c_2)$ by the definitions. The converse follows from Lemma 3.1.4 and Theorem 3.1.1. □

A more general form of Curry's Property E (Theorem 2.1.1) expresses that complete derivations relative to a finite set of families are upperbounded in length and equivalent (see [15]).

3.2 COMPLETE DERIVATIONS

Definition 3.2.1. The set C of occurrences is *f-complete* with respect to D, in short (D, C) is *f*-complete, iff $C = \{c' | (D, c') \sim (D, c)\}$ for any $c \in C$.

Definition 3.2.2. (D, C) is *d-complete* if C is maximal such that there are D', D'', c with $D \sim D';D''$ and $C = c/D''$.

Definition 3.2.3. (D, C) is *l*-complete if $C = \{c' | \mu'(c') = \mu'(c)\}$ when $c \in C$ and $D:(t, \mu) \overset{*}{\to} (t', \mu')$.

Definition 3.2.4. The derivation D is *f*-complete (respectively *d*- and *l*-complete) if $D \neq \varnothing$ or $D = D_1;C$ when D_1 and (D_1, C) are *f*-complete (respectively *d*- and *l*-complete).

LEMMA 3.2.1. *If D is f-complete, then $(D, c)_0 \leq (D, c)$ for any c.*

PROOF. •By Proposition 3.1.7, it is sufficient to prove $d_0 \leq D$ if $(D, c)_0 = (d_0, c_0)$. We use an induction on $|D|$. If $D = \varnothing$, then $d_0 = \varnothing \leq D$. Otherwise $D = D_1;C_1$ with D_1 and (D_1, C_1) *f*-complete. If $c \in c_1/C_1$, then $(D, c)_0 = (D_1, c_1)_0$ and by induction $d_0 \leq D_1 \leq D$. Otherwise, there is only one c_1 in C_1 which creates c. Let $(d_0', c_0') = (D_1, c_1)_0$. One has $d_0 = d_0';c_0'$ by Definition 3.1.3. But, as (D_1, C_1) is *f*-complete we get $(D_1, c_1)_0 = (D_1, c_2)_0$ for all c_2 in C_1 by Theorem 3.1.1. By induction $d_0' \leq D_1$. Hence $C_1 = c_0'/(D_1/d_0')$ by Propositions 3.1.7 and 3.1.2. So $d_0 \leq D$. □

PROPOSITION 3.2.1. *D is f-complete iff D is d-complete iff D is l-complete when the initial labelings are injective and elementary.*

PROOF. Obvious with the above lemma and Theorem 3.1.2. □

Notation 3.2.1. We shall abbreviate *f*- (*d*- *l*-) complete derivation into *complete derivation*. We denote by $\mathscr{F}\mathscr{D}(t)$ the set of complete derivation starting at t.

PROPOSITION 3.2.2. *The set $[\mathscr{F}\mathscr{D}(t)] = \mathscr{F}\mathscr{D}(t)/\sim$ with the $\leq$ ordering is a sublattice of $[\mathscr{D}(t)]$ with the same l.u.b. operation.*

PROOF. Let us remark that if $(d_0, c_0) = (D_2, c_2)_0$ and $d_0 \leq D_1 \leq D_2$, then there is a c_1 such that $(d_0, c_0) \leq (D_1, c_1) \leq (D_2, c_2)$ by Propositions 3.1.2 and 3.1.7. Suppose now that D_1, D_2 are complete; then it is obvious by induction on $|D_2|$ that $D_1;(D_2/D_1)$ is complete for any D_2 with the use of Lemma 3.2.1. □

Notation 3.2.2. Let the *family* $[D, c]$ be the equivalence class of (D, c) with respect to $\sim$. Notice that the notation $[D, C]$ is unambiguous when $D;C$ is complete. Let $\mathscr{F}(D)$ be the set of families of occurences derived in D.

The next result shows that a family is derived at most once in a complete derivation.

PROPOSITION 3.2.3. *If $D_1;C_1;D_2;C_2;D_3$ is complete, then*

$$[D_1, C_1] \neq [D_1;C_1;D_2, C_2].$$

PROOF. By Lemma 3.2.1 and Proposition 3.1.7. □

Definition 3.2.5. For any $d \in \mathscr{D}(t)$, the complete derivation $\bar{d}$ associated with d is defined by

$$\begin{aligned} &\underline{\varnothing} = \varnothing, \\ &\overline{d;c} = \bar{d};C, \quad \text{where } c = \{c' | (d, c) \sim (\bar{d}, c')\}. \end{aligned}$$

It is straightforward that $\bar{d}$ is complete, that $d \leq \bar{d}$, and that every complete $D \in \mathscr{F}\mathscr{D}(t)$ is equal to $\bar{d}$ for some $d \in \mathscr{D}(t)$. Note that $d \sim d'$ does not imply $\bar{d} \sim \bar{d}'$.

3.3 Outermost Complete Derivations. As announced in the Introduction, the correctness problem leads us to consider outermost derivations. We summarize here the syntactic properties of outermost derivations that we shall need.

Definition 3.3.1. An occurrence c in t is *outermost* if it is not contained in another occurrence c' in t. A derivation $D \in \mathscr{D}(t)$ is *outermost* iff at least one outermost occurrence is derived at each step of D.

The main property of outermost occurrences and outermost derivations are the following ones.

Lemma 3.3.1. *Let c be outermost in t; let $t \xrightarrow{C} t'$ where $c \notin C$. Then $c/C = \{c\}$ and c is outermost in t'.*

Proof. Straightforward. □

Proposition 3.3.1. *Let d be outermost. Then $d \leq D$ implies $\mathscr{F}(d) \subset \mathscr{F}(D)$. Moreover $\mathscr{F}(d) = \mathscr{F}(\bar{d})$.*

Proof. By induction on $|d|$. Let $d = d_1;c$. Then $\mathscr{F}(d_1) \subset \mathscr{F}(D)$ since $d_1 \leq d \leq D$. Furthermore $c/(D/d_1) = \varnothing$. Hence $D/d_1 = D_1';C;D_2'$ with $c \in C$ by Lemma 3.3.1. And $[d_1, c] = [(d_1;D_1'), c]$. By Definitions 2.2.1 and 3.1.1, one has $[(d_1;D_1'), c] \in \mathscr{F}(D)$. Thus $\mathscr{F}(d) \subset \mathscr{F}(D)$. Now consider $D = \bar{d}$. As $d \leq \bar{d}$, we get $\mathscr{F}(d) \subset \mathscr{F}(\bar{d})$. And $\mathscr{F}(\bar{d}) \subset \mathscr{F}(d)$ is obvious by the definition of $\bar{d}$. □

Proposition 3.3.2. *If d is outermost, then $\bar{d}$ is outermost.*

Proof. By induction on $|d|$. Let $d = d_1;c$. We have two cases. First $d \leq \bar{d}_1$ and $\mathscr{F}(d) \subset \mathscr{F}(\bar{d}_1)$ by Proposition 3.3.1. Then $\bar{d} = \bar{d}_1$ by Proposition 3.2.3, and $\bar{d}$ is outermost by induction. Second $c/(\bar{d}_1/\mathrm{d}_1) \neq \varnothing$. Then $\bar{d} = \bar{d}_1;C$ where $c \in C$ and c is outermost by Lemma 3.3.1. Hence $\bar{d}$ is outermost. □

Proposition 3.3.3. *Let $\bar{d}$ be outermost. Then $\bar{d} \leq \bar{d}'$ is equivalent to $\mathscr{F}(\bar{d}) \subset \mathscr{F}(\bar{d}')$.*

Proof. That $\bar{d} \leq \bar{d}'$ implies $\mathscr{F}(\bar{d}) \subset \mathscr{F}(\bar{d}')$ is shown as in Proposition 3.3.1. Conversely, assume $\mathscr{F}(\bar{d}) \subset \mathscr{F}(\bar{d}')$. Then $\mathscr{F}(\bar{d}';(\bar{d}/\bar{d}')) \subset \mathscr{F}(\bar{d}') \cup \mathscr{F}(\bar{d}') = \mathscr{F}(\bar{d}')$. But $\bar{d}';(\bar{d}/\bar{d}')$ is complete by Proposition 3.2.2. Hence $\bar{d}/\bar{d}' = \varnothing$ by Proposition 3.2.3. □

4. *Minimal and Optimal Computations*

4.1 Semantics of Recursive Programs. We first study infinite derivations. We then define interpretations in terms of complete F-algebras and eventually we define the computations of a program.

Definitions: Ordered Sets. We consider partially ordered sets $(D, \subset, \bot)$ having a least element $\bot$. A totally ordered subset of D is called a *chain*. When they exist, l.u.b. and g.l.b. are denoted by the symbols $\cup$ and $\cap$. A subset ∇ of D is *directed* iff it is nonempty and satisfies $\forall \alpha, \beta \in \nabla$, $\exists \gamma \in \nabla$ such that $\alpha \subset \gamma$ and $\beta \subset \gamma$, and D is a *complete partial order*, or is a *c.p.o.*, if every directed set $\nabla \subset D$ has a l.u.b. or *limit* $\cup\nabla$. Two elements $\alpha, \beta \in D$ are *joinable*, written $\alpha \uparrow \beta$, iff $\exists \gamma \in D$ such that $\alpha \subset \gamma$ and $\beta \subset \gamma$, and a c.p.o. D is *consistently complete* if any two joinable elements α and β have a l.u.b. $\alpha \cup \beta$; then any subset $X \subset D$ has a g.l.b. $\cap X = \cup\{\alpha \mid \forall x \in X, \alpha \subset x\}$ since the latter set is directed.

A mapping $h: D \to D'$ is *monotonic* if $\alpha \subset \beta$ implies $h(\alpha) \subset' h(\beta)$. If D and D' are c.p.o.'s, a monotonic mapping $h: D \to D'$ is *continuous* iff $h(\cup\nabla) = \cup h(\nabla)$ holds for every directed $\nabla \subset D$ (note that the set $\cup h(\nabla)$ is directed).

Definitions: Completions of an Ordered Set [4]. Let $(D, \subseteq, \bot)$ be an ordered set. A *completion* D^∞ of D is a complete set containing D (up to monotonic injection) and such that for any complete D', every monotonic mapping $h: D \to D'$ extends in a unique way into a continuous mapping $h^\infty: D^\infty \to D'$. All completions are equal up to unique isomorphism. A standard completion is the *completion by ideals*: Call *ideal* a directed set $\nabla \subset D$ such that $\forall \alpha \in \nabla$, $\forall \beta \in D$, $\beta \subset \alpha$ implies $\beta \in \nabla$, and let D^∞ be the set of ideals of D ordered by inclusion. Then D^∞ has least element $\{\bot\}$, each $\alpha \in D$ is represented by the ideal $\{\beta \in D \mid \beta \subset \alpha\}$, and the l.u.b. of a directed set $\nabla \subset D^\infty$ is the union of the elements of ∇. Every monotonic mapping $h:D \to D'$ extends in a unique way to a continuous mapping $h^\infty:D^\infty \to D'$, where $h^\infty(I) = \cup\{h(d) \mid d \in I\}$.

We keep the same ordering denotation $\subset$ for a completion D^∞ of D.

Definitions: Infinite Derivations. Let $\mathscr{D}^\infty(t)$ be the set of infinite set derivations from t, and $\mathscr{F}\mathscr{D}^\infty(t)$ be the set of infinite complete derivations from t, defined as in Section 3.2. We denote by δ a (singleton) infinite derivation, and by $\bar{\delta}$ its associated complete derivation (see Definition 3.2.4). We denote by Δ an arbitrary infinite set derivation. Let Δ_i be the i first steps of Δ, and define $\Delta \leq_\infty \Delta'$ to hold iff $\forall i \geq 0$, $\exists j \geq 0$ such that $\Delta_i \leq \Delta'_j$. Then $\leq_\infty$ is a preorder, and we denote by $\sim_\infty$ its associated equivalence and we write $[\Delta]$ for the class of Δ and $[\mathscr{D}^\infty(t)]$ for $\mathscr{D}^\infty(t)/\sim_\infty$.

Call a *complete lattice* a lattice L such that any proper subset of L has a g.l.b. and a l.u.b.

PROPOSITION 4.1.1 *The structure $\langle[\mathscr{D}^\infty(t)], \leq_\infty\rangle$ is a completion of $\langle\mathscr{D}(t), \leq\rangle$ and is a complete lattice, of which $\langle[\mathscr{F}\mathscr{D}^\infty(t)], \leq\rangle$ is a sublattice.*

PROOF. Notice that $[\mathscr{D}^\infty(t)]$ contains $\mathscr{D}(t)$ and that the relation $\leq$ is the restriction of $\leq_\infty$ to $\mathscr{D}(t)$. Moreover we have from the definitions $\Delta = \cup\{\Delta_k \mid k \geq 0\}$ and the first part of the proposition follows. Using the completion by ideals, it is easily seen that a completion of an upper semilattice is a complete lattice, which shows the second part. The third part is obvious from Proposition 2.2.3. □

We denote therefore $\leq_\infty$ and $\sim_\infty$ by $\leq$ and $\sim$. It is easy to see that the *full or Kleene derivation $K(t)$* (see [20]), which consists in deriving at each step all occurrences of unknown function symbols, belongs to the maximum class of $[\mathscr{D}^\infty(t)]$.

Definitions: Ordered and Complete F-Algebras [20]. An *ordered (complete) F-algebra* is an F-algebra $I = \langle D_I, \subset, \perp, f_i^I\rangle$ where D_I is ordered and has least element $\perp$, and the mappings f^I are monotonic (continuous). A *morphism* of ordered (complete) F-algebras is a morphism θ of the underlying F-algebras which preserves $\perp$ (i.e. satisfies $\theta(\perp) = \perp$), and is monotonic (continuous). A *free ordered (complete) F-algebra generated by V* is such that for any ordered (complete) F-algebra I, any mapping $\nu:v \to D_I$ extends in a unique way into a morphism of ordered (complete) F-algebras (see "The Free F-Algebra" in Section 1.3). To construct a free ordered F-algebra $M_\Omega(F, V)$, consider a new nullary symbol Ω and order $M(F\cup\{\Omega\}, V)$ by the least ordering $<$ such that $\Omega < a$ for all a and $f(\mathbf{a}) < f(\mathbf{a}')$ for all $f \in F$ and $\mathbf{a}$, $\mathbf{a}'$ such that $\mathbf{a} < \mathbf{a}'$ (this ordering can be seen as the "initial segment" ordering on trees; see the Introduction). We let $M_\Omega = \langle M(F \cup \{\Omega\}, V), <, \Omega, f_i\rangle$ and denote by a, a', ... the elements of M_Ω. It is easily seen that the completion by ideals $M_\Omega^\infty = \langle M(F \cup \{\Omega\}, V)^\infty, <, \Omega, f_i^\infty\rangle$ is a free complete F-algebra generated by V. Its elements are denoted by A, A', ... and can be viewed as infinite trees written on F, V, Ω and ordered by the "initial segment" ordering. The functions f_i^∞ are simply denoted f_i. For any $A \in M_\Omega^\infty$, complete F-algebra I, and $\nu: V \to D_I$, $(I, \nu)A$ is the value of A when the f_i are interpreted by the f_i^I, the x_i by the $\nu(x_i)$, and Ω by $\perp$.

Definitions: Symbolic Value of a Term. We associate with any $t \in M = M(F \cup \Phi, V)$ its symbolic value $\omega(t) \in M_\Omega(F, V)$ by replacing all occurrences of elements of Φ by Ω:

(i) $\omega(x_i) = x_i$,

(ii) $\omega(f(\mathbf{t})) = f(\omega(\mathbf{t}))$ for all $f \in F$,

(iii) $\omega(\phi(\mathbf{t})) = \Omega$ for all $\phi \in \Phi$.

LEMMA 4.1.1. *If $t \xrightarrow{*} t'$, then $\omega(t) < \omega(t')$.*

PROOF. Obvious by induction on the length of the derivation and on the size of t. □

Definitions: Programs and Values of Derivations. We call *program* a triple $P = (\Sigma, I, \nu)$ where Σ is a program scheme, I is a complete F-algebra called the *interpretation*, and $\nu: V \to D_I$ is a *data mapping*. If $P = (\Sigma, I, \nu)$ is a program and if $t \in M$, then the set $\{(I, \nu)\omega(t') \mid t \xrightarrow{*} t'\}$ is directed in D_I by Lemma 4.1.1 and the Church-Rosser property. This set has a limit $P(t)$ in D_I, called the *value computed by P on t*. For $I = M_\Omega^\infty$ and $\nu: x \to x$, we write $\Sigma(t) = (\Sigma, M_\Omega^\infty, \nu)(t) = \cup\{\omega(t') \mid t \xrightarrow{*} t'\}$ for all $\Sigma(t)$ *the symbolic value computed by Σ on t*. Let $\Delta: t_0 \to t_1 \to t_2 \to \cdots \in \mathscr{D}^\infty(t)$. Then the set $\{(I, \nu)\omega(t_i) \mid i \in N\}$ is a chain in D_I, and has a limit $(I, \nu)\Delta$ called *the value computed by Δ in (I, ν)*. As before, the *symbolic value computed by Δ* is $\Sigma(\Delta) = \cup\{\omega(t_i) \mid i \in N\} \in M_\Omega^\infty$.

Given P, Δ and $\alpha \in D_I$, we say that Δ *computes* α if $\alpha \subset (I, \nu)\Delta$. We call *correct* (respectively *symbolically correct*) any Δ which computes $P(t)$ (respectively $\Sigma(t)$). It is clear from a remark above that the Kleene derivation $K(t)$ is always correct.

4.2 Least Computations in M_Ω^∞. We now restrict our attention to the symbolic interpretation M_Ω^∞, and first prove the existence of least computations of every approximation of $\Sigma(t)$.

Definition 4.2.1. Let $t \in M$ and let $A \in M_\Omega^\infty$ satisfy $A < \Sigma(t)$. Then $\Delta \in \mathscr{L}^\infty(t)$ is *least for A* iff it computes A and satisfies $\Delta \leq \Delta'$ for any Δ' which computes A.

Lemma 4.2.1. *If $\Sigma(t) \neq \Omega$, then there exists a least $n \geq 0$ such that $\omega(\epsilon^n) \neq \Omega$. Every standard or outermost derivation $d \in \mathscr{D}(t)$ such that $\omega(d) \neq \Omega$ is of the form $d = \epsilon^n; f(d_1, d_2, \ldots, d_k)$ where $t \xrightarrow{\epsilon^n} f(t_1, t_2, \ldots, t_k)$ and $d_i \in \mathscr{D}(t_i)$ for $1 \leq i \leq k$.*

Proof. If $\Sigma(t) \neq \Omega$, then there exists a standard $d \in \mathscr{D}(t)$ such that $\omega(d) \neq \Omega$. If $t = f(\mathbf{t})$ then $n = 0$ satisfies the conditions. Otherwise $t = \phi(\mathbf{t})$, and any standard or outermost d' such that $\omega(d') \neq \Omega$ must begin with ϵ. The result follows by induction on $|d|$. □

Definition 4.2.2. Let $t \in M$ and $a \in M_\Omega$ satisfy $a < \Sigma(t)$. Let $d_a(t) \in \mathscr{D}(t)$ be constructed by induction on $\|a\|$ by

(i) $d_a(t) = \varnothing$ if $a = \Omega$ or $a = x \in V$ or $a = f \in F^0$,

(ii) $d_a(t) = \epsilon^n, f(d_{a_1}(t_1), d_{a_2}(t_2), \ldots, d_{a_k}(t_k))$ if $a = f(a_1, a_2, \ldots, a_k)$ and $t \xrightarrow{\epsilon^n} f(t_1, t_2, \ldots, t_k)$.

Proposition 4.2.1. *If $t \in M$ and $a < \Sigma(t)$, then $d_a(t)$ is standard, outermost, and least for a.*

Proof. Obvious by induction on $\|a\|$ from the definitions and Lemma 4.2.1. □

We extend the result to infinite derivations.

Definition 4.2.3. Let $t \in M$ and $A \in M_\Omega^\infty$ satisfy $A < \Sigma(t)$. Let $S = \{a_n | n \geq 1\} \subset M_\Omega$ be an increasing chain with limit A in M_Ω^∞ (such a chain always exists by the definition of a completion). Let $\delta_S(t) \in \mathscr{D}^\infty(t)$ be such that $\delta_S(t) = d_{a_1}(t_0); d_{a_2}(t_1); \cdots; d_{a_k}(t_{k-1}); \cdots$ where $t_0 = t$ and $d_{a_k}(t_{k-1}): t_{k-1} \xrightarrow{*} t_k$ is constructed as in Definition 4.2.2. Denote $\delta_S^k(t) = d_{a_1}(t_0); d_{a_2}(t_1); \cdots; d_{a_k}(t_{k-1})$ and $\delta_S^0(t) = \varnothing$.

Theorem 4.2.1. *If S is a chain of limit A, then $\delta_S(t)$ is least for A and also for any A' such that $A < A' < \omega(\delta_S(t))$.*

Proof. Let A' be such that $A < A' < \omega(\delta_S(t))$, let δ such that $\omega(\delta) > A'$, and assume by induction $\delta \geq \delta_S^k(t)$ which holds trivially for $k = 0$. Then by the definition of $\leq$ on $\mathscr{L}^\infty(t)$, there exists $\delta' \in \mathscr{D}(t_k)$ such that $\delta \sim \delta_S^k(t); \delta'$. But $\omega(\delta') = \omega(\delta) > A' > a_{k+1}$ implies $\delta' \geq d_{a_{k+1}}(t_k)$ by Proposition 4.2.1, and therefore $\delta \geq \delta_S^k(t); d_{a_{k+1}}(t_k) = \delta_S^{k+1}(t)$. □

Note that since $\delta_S(t)$ is a singleton derivation, all occurrences which it derives are outermost. It is intuitively clear that $\delta_S(t)$ performs no useless steps, which can be of two kinds:

—rewriting of an occurrence contained in an outermost occurrence which is useless for computing A: This is avoided since $\delta_S(t)$ is least for A.

—rewriting of an occurrence which will disappear when rewriting some outermost occurrence which does not use all its arguments (as in $K(x, y) = x$): This is avoided since $\delta_S(t)$ is outermost.

4.3 Optimal Computations in M_Ω^∞. As we said in the Introduction, since $\delta_S(t)$ performs no useless steps, we can think of constructing some "optimal computation" from it. But, for the definition of optimality, we need to compare various cost measures corresponding to various implementations using different data structures.

We simply consider implementations as sets of *physical computations* K, and we assume that we can associate to any K some derivation $r(K)$ and some integer cost(K). We restrict our attention to implementation where the only shared objects correspond to occurrences of the same family in the associated derivations. This will be expressed by requiring the following inequality:

$$\text{cost}(K) \geq \text{card}(\mathscr{F}(r(K))),$$

where card(X) is the number of elements in the set X.

The dag structure of [18] described in the Introduction or the delay rule of [25] are particular implementations of this kind. They are exactly implement complete derivations, so that $r(K)$ is complete for any physical computation K. The natural cost measure for such implementations is $\text{cost}(K) = |r(K)|$. We have $\text{cost}(K) \geq \text{card}(\mathscr{F}(\overline{r(K)}))$ by Proposition 3.2.3.

Let us see that those implementations which exactly implement complete derivations (dag's or delay rules) are indeed the best possible ones under our constraints. If K is a physical computation in some implementation, let $d = r(K)$. We have $\omega(d) < \omega(\bar{d})$ and also $|\bar{d}| \leq \text{cost}(K)$. This follows since $\text{card}(\mathscr{F}(d)) \leq \text{cost}(K)$ by hypothesis and $|\bar{d}| = \text{card}(\mathscr{F}(\bar{d}))$ by Proposition 3.2.3. Hence by directly implementing $\bar{d}$ at cost $|\bar{d}|$ as with dag's, we get a better result with a lower cost.

Therefore optimality results only need to be investigated within complete derivations with cost measure $|\bar{d}|$. But remembering that $|\bar{d}| = \text{card}(\mathscr{F}(\bar{d}))$, we shall compare derivations not only by their lengths (i.e. $|\bar{d}| \leq |\bar{d}'|$) but also by the set of families they derive (i.e. $\mathscr{F}(\bar{d}) \subset \mathscr{F}(\bar{d}')$). This will give a more precise optimality result which extends to the infinite case where no cost measure in the usual sense is definable.

Definition 4.3.1. Let $t \in M$ and $a < \Sigma(t)$. Then $\bar{d} \in \mathscr{F}\mathscr{D}(t)$ is *optimal* for a if $a < \omega(\bar{d})$ and $|\bar{d}| \leq |\bar{d}'|$ for any $\bar{d}' \in \mathscr{F}\mathscr{D}(t)$ such that $a < \omega(\bar{d}')$. Let $A < \Sigma(t)$. *Then* $\bar{\delta} \in \mathscr{F}\mathscr{D}^\infty(t)$ is *$\mathscr{F}$-optimal* for A if $A < \omega\,(\bar{\delta})$ and $\mathscr{F}(\bar{\delta}) \subset \mathscr{F}(\bar{\delta}')$ for any $\bar{\delta}' \in \mathscr{F}\mathscr{D}^\infty(t)$ such that $A < \omega(\bar{\delta}')$.

THEOREM 4.3.1. *Let $t \in M$ and $a < \Sigma(t)$, let $a' \in M_\Omega$ such that $a < a' < \omega(\overline{d_a(t)})$. Then $\overline{d_a(t)}$ is least for a and a' in $\mathscr{F}\mathscr{D}(t)$ and is optimal and $\mathscr{F}$-optimal for a and a'.*

PROOF. Let $\bar{d} \in \mathscr{F}\mathscr{D}(t)$ such that $a < \omega(\bar{d})$. Then $d_a(t) \leq \bar{d}$ by Proposition 4.2.1; hence $\mathscr{F}(\overline{d_a(t)}) = \mathscr{F}(d_a(t)) \subset \mathscr{F}(\bar{d})$ by Proposition 3.3.1 and $\overline{d_a(t)} \leq \bar{d}$ and $|\overline{d_a(t)}| \leq |\bar{d}|$ by Propositions 3.3.3 and 3.2.3. □

THEOREM 4.3.2. *Let $t \in M$ and $A < \Sigma(t)$. Let S be an increasing chain in M_Ω with limit S in M_Ω^∞; let $A' \in M_\Omega^\infty$ such that $A < A' < \omega(\overline{\delta_S(t)})$. Then $\overline{\delta_S(t)}$ is least for A and A' in $\mathscr{F}\mathscr{D}^\infty(t)$ and is optimal for A and A'.*

PROOF. Straightforward extension of Theorem 4.3.1 to the infinite case (see Theorem 4.2.1). □

Let us now see how to effectively construct optimal derivations:

Definition 4.3.2 (see Barendregt [1]). A *computation rule* in M_Ω^∞ is a total recursive mapping $\mathscr{C}$ which associates a subset C of $\mathscr{C}(\Phi, t)$ to any $t \in M$. The derivation $\Delta_{\mathscr{C}}(t)$ defined by $\mathscr{C}$ on t is

$$\Delta_{\mathscr{C}}(t): t \xrightarrow{\mathscr{C}(t)} t_1 \xrightarrow{\mathscr{C}(t_1)} t_2 \to \cdots \xrightarrow{\mathscr{C}(t_n)} t_{n+1} \to \cdots.$$

Notice that $\mathscr{C}(t_n)$ may be empty for some t_n.

LEMMA 4.3.1. *$\Sigma(t) = \Omega$ holds iff there exists $d \in \mathscr{D}(t)$ such that $d = \epsilon^{N+1}$ (see Notation* 2.4.1).

PROOF. By induction on N. □

Definition 4.3.3. Let the *parallel useful outermost computation rule* π be defined by

$$\pi(\phi(\mathbf{t})) = \begin{cases} \varnothing & \text{if } \Sigma(\phi(\mathbf{t})) = \Omega, \\ \{\epsilon\} & \text{otherwise;} \end{cases}$$

$$\pi(f(\mathbf{t})) = \bigcup_{i=1}^{n} i.\,\pi(t_i).$$

Note that π is indeed a computation rule since $\Sigma(t) = \Omega$ is decidable by Lemma 4.3.1. Although $\bar{d}$ was only defined for singleton derivations d, it is easy to define $\overline{\Delta_\pi(t)}$. But the cost of one set derivation in $\overline{\Delta_\pi(t)}$ is the number of families of rewritten occurrences.

The following result extends a result in Downey and Sethi [9].

PROPOSITION 4.3.1. *The derivation $\Delta_\pi(t)$ is correct for any t, and $\delta \in \mathscr{D}^\infty(t)$ is correct iff $\delta \geq \Delta_\pi(t)$. The complete derivation $\overline{\Delta_\pi(t)}$ is optimal in M_Ω^∞ for any t.*

PROOF. By construction, we have $\Delta_\pi(t) \sim \delta_S(t)$ where S is the symbolic computation of $\Delta_\pi(t)$. The results follow from Theorem 4.2.1 and Proposition 4.2.1, since $\mathcal{F}(\Delta_\pi(t)) = \mathcal{F}(\delta_S(t))$. □

It is also easy to see that any optimal $\bar{\delta} \in \mathcal{D}^\infty(t)$ is a "true permutation" of $\overline{\Delta_\pi(t)}$, i.e. can be obtained from $\overline{\Delta_\pi(t)}$ by a sequence of permutations of the form $C_1;(C_2/C_1) \sim C_2;(C_1/C_2)$ with $C_1/C_2 \neq \varnothing$ and $C_2/C_1 \neq \varnothing$.

4.4 OPTIMAL COMPUTATIONS IN INTERPRETATIONS. We assume that we are given a program $P = (\Sigma, I, \nu)$ and an initial term t, and we investigate the existence and effectiveness of optimal computations of approximation α of $P(t)$.

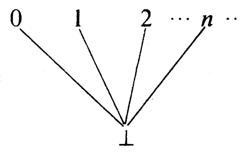
Definition 4.4.1. Let $t \in M$ and $\alpha \subset P(t)$ in D_I. Then $\delta \in \mathcal{D}^\infty(t)$ is *least for* α if it computes α and satisfies $\delta \leq \delta'$ for any δ' which computes α. If $\bar{d} \in \mathcal{F}\mathcal{D}(t)$, then $\bar{d}$ is *optimal for* α if it computes α and satisfies $|\bar{d}| \leq |\bar{d}'|$ for any $\bar{d}' \in \mathcal{F}\mathcal{D}^\infty(t)$ which computes α. If $\bar{\delta} \in \mathcal{F}\mathcal{D}^\infty(t)$, then $\bar{\delta}$ is $\mathcal{F}$*-optimal for* α if it computes α and satisfies $\mathcal{F}(\bar{\delta}) \subset \mathcal{F}(\bar{\delta}')$ for any $\bar{\delta}' \in \mathcal{F}\mathcal{D}^\infty(t)$ which computes α. We abbreviate optimal for $P(t)$ and $\mathcal{F}$-optimal for $P(t)$ into *optimal* and $\mathcal{F}$*-optimal*.

Several difficulties appear when trying to define "optimal computation rules" in interpretations:

—The process of determining the next occurrences to be rewritten may involve computations in the interpretation. We must assume that these computations are effective. (Another approach would be to consider simplifications as in [26].)

—No computation rule can always generate optimal derivations, since the empty derivation is optimal if and only if $P(t) = \perp$, which is undecidable in general.

—An approximation α of $P(t)$ may have no least computations and several optimal but not $\mathcal{F}$-optimal computations. Let us give an example over the flat domain of integers:

$N_\perp =$ 

Here optimal derivations always exist since every value $\alpha \subset P(t)$ is always computed by some finite derivation.

Let *pmult* be the "parallel multiplication" defined by

$$\begin{aligned} &pmult(\perp, \perp) = pmult(\perp, n+1) = pmult(n+1, \perp) = \perp,\\ &pmult(n, n') = n \times n',\\ &pmult(0, \perp) = pmult(\perp, 0) = 0. \end{aligned}$$

Consider the program scheme $\Sigma{:}\phi(x) = x$ and the initial term $t = pmult(\phi(0), \phi(0))$. Then $P(t) = 0$ and $P(t)$ has no least derivation and two optimal but not $\mathcal{F}$-optimal derivations $t \to pmult(0, \phi(0))$ and $t \to pmult(\phi(0), 0)$. Moreover no rule can optimally compute $t = pmult(t_1, t_2)$ for all t_1 and t_2: If $t_1 = 0$ and $t_2 \neq 0$, the optimal derivations of t are given by those of t_1, and symmetrically if $t_1 \neq 0$ and $t_2 = 0$. If $t_1 = t_2 = 0$, then the optimal derivations of t are given by the shortest optimal derivation of t_1 and t_2. But $t_1 = 0$ and $t_2 = 0$ are again undecidable. In fact even to be correct, any rule must evaluate t_1 and t_2 in parallel: This is why *pmult* is called "parallel multiplication"; see [25, 26].

Hence we cannot hope to have simple characterizations of optimal derivations and rules except in particular classes of interpretations. We shall first give a sufficient condition for least and $\mathcal{F}$-optimal derivations to exist.

Definition 4.4.2. Let I be an interpretation and $\nu{:}\, V \to D_I$ be a data mapping. Denote $c_{I,\nu}(\alpha, A) = \cap\{A' < A \mid (I, \nu)A' \supset \alpha\}$ for any $A \in M_\Omega^\infty$ and $\alpha \subset (I, \nu)A$ in D_I. Then I is *projective* iff $(I, \nu)(c_{I,\nu}(\alpha, A)) \supset \alpha$ holds for every ν and A. We then call $c_{I,\nu}(\alpha, A)$ the *canonical form of A for* α and $c_{I,\nu}((I, \nu)A, A)$ the *canonical form of A*. Most of the time, I and ν will be assumed from the context. Then we write $c(\alpha, A) = c_{I,\nu}(\alpha, A)$ and $c(A) =$

$c_{I,\nu}((I, \nu)A, A)$. We write $A \supset \alpha$ for $(I, \nu)\, A \supset \alpha$ and $A \supset A'$ for $(I, \nu)A \supset (I, \nu)A'$. We use the relation $A \equiv A'$ for $(I, \nu)A = (I, \nu)A'$. We denote by $\subsetneq$ and $\underset{\neq}{<}$ the strict orderings associated with $\subset$ and $<$, so that $\alpha \subsetneq \beta$ means $\alpha \subset \beta$ and $\alpha \neq \beta$.

LEMMA 4.4.1. *If $\alpha \subset \alpha' \subset A$, then $c(\alpha, A) < c(\alpha', A)$. If $A < A'$ and $\alpha \subset A$, then $c(\alpha, A) = c(\alpha, A')$.*

PROOF. Straightforward. □

THEOREM 4.4.1. *Let I projective and $\nu: V \to D_I$. Let $t \in M$ and $\alpha \in D_I$ such that $\alpha \subset P(t)$; let S be any chain in M_Ω of limit $c_{I,\nu}(\alpha, \Sigma(t))$ in M_Ω^∞. Let $\alpha', \alpha'' \in D_I$ satisfy $\alpha \subset \alpha' \subset (I, \nu)(\delta_S(t))$ and $\alpha \subset \alpha'' \subset (I, \nu)(\overline{\delta_S(t)})$. Then $\delta_S(t)$ is least for α and α' in $\mathscr{D}^\infty(t)$ and $\delta_S(t)$ is least and $\mathscr{F}$-optimal for α and α'' in $\mathscr{F}\mathscr{D}^\infty(t)$.*

PROOF. Computing α in D_I is nothing but computing $c(\alpha, \Sigma(t))$ in M_Ω^∞; the result follows from Theorems 4.2.1 and 4.3.1 and from $c(\alpha', \Sigma(t)) < \omega(\delta_S(t))$ and $c(\alpha'', \Sigma(t)) < \omega(\delta_S(t))$ implied by Lemma 4.4.1. □

When computing an $\alpha \subset f(t_1, t_2, \dots, t_n)$ in a projective interpretation, the computations of $t_1, t_2, \dots, t_n$ are in some sense independent; it is impossible to compute less of t_1 by computing more of t_2, as it was when computing *pmult*(t_1, t_2) when $t_1 = t_2 = 0$. Hence the projectivity condition forbids "parallel functions" like *pmult*: We indeed have no canonical form for *pmult*$(0, 0)$.

The projectivity condition may be restricted to only hold on finite terms if D_I is algebraic (see [4, 7]).

Definition 4.4.3. An element $\alpha \in D_I$ is *isolated* iff for all directed sets $\nabla \subset D_I$, $\alpha \subset \cup\nabla$ implies $\exists\beta \in \nabla$ such that $\alpha \subset \beta$. Let $\mathscr{A}(\alpha) = \{\beta \in D_I | \beta$ isolated and $\beta \subset \alpha\}$. Then D_I is *algebraic* iff for every $\alpha \in D_I$ the set $\mathscr{A}(\alpha)$ is directed and has l.u.b. α. As an example, the set M_Ω^∞ is algebraic and has for isolated elements the finite trees of M_Ω.

PROPOSITION 4.4.1. *If D_I is algebraic, then I is projective iff $(I, \nu)(c_{I,\nu}(\alpha, a)) \supset \alpha$ holds for every $a \in M_\Omega$ and α isolated such that $\alpha \subset (I, \nu)a$.*

PROOF. Let α isolated and assume $A \supset \alpha$. Then since $(I, \nu)A = \cup\{(I, \nu)\, a | a < A\}$ there exists $a < A$ such that $a \supset \alpha$. If $A' < A$ is such that $A' \supset \alpha$, there exists $a' < A'$ such that $a' \supset \alpha$. Then $c(\alpha, a) = c(\alpha, a \cup a') = c(\alpha, a') < a' < A'$ holds as in Lemma 4.4.1. Hence $c(\alpha, A) = c(\alpha, a) \supset \alpha$. Now for any $\alpha \in D_I$, consider $A' = \cup\{c(\beta, A) | \beta \in \mathscr{A}(\alpha)\}$, where the set on the right is directed since $\mathscr{A}(\alpha)$ is too. If $A'' \supset \alpha$, then $A'' \supset \beta$ and $A'' > c(\beta, A)$ for all $\beta \in \mathscr{A}(\alpha)$, so that $A'' > A'$. But $(I, \nu)A' = \cup\{(I, \nu)c(\beta, A) | \beta \in \mathscr{A}(\alpha)\} \supset \cup\{\beta | \beta \in \mathscr{A}(\alpha)\} = \alpha$ since D_I is algebraic and eventually $A' = c(\alpha, A) \supset \alpha$. □

In this case we also have the converse of Theorem 4.4.1:

PROPOSITION 4.4.2. *If D_I is algebraic and I is not projective, then there exist a program P, a term t, and an $\alpha \subset P(t)$ such that α has no $\mathscr{F}$-optimal derivation.*

PROOF. There exist ν, a, and $\alpha \subset a$ such that $c(\alpha, a) \supsetneq \alpha$, so that the set $\{a' < a | a' \supset \alpha\}$ has at least two distinct minimal elements b and b', with $b \cap b' \supsetneq \alpha$. With additional variables y_i, we can write $b \cap b' = b_1[\mathbf{\Omega}/\mathbf{y}]$ and $b = b_1[\mathbf{c}/\mathbf{y}]$ and $b' = b_1[\mathbf{c}'/\mathbf{y}]$ with $\mathbf{c} \cap \mathbf{c}' = \mathbf{\Omega}$. Take then $\Sigma: \phi(x) = x$ and $t = b_1[\phi(c_1 \cup c_1')/y_1, \phi(c_2 \cup c_2')/y_2, \dots, \phi(c_k \cup c_k')/y_k]$: There is no optimal computation of α from t for (Σ, I, ν). □

Notice that the projectivity condition mixes syntax and semantics. We now investigate purely semantic conditions on the f^I which make I projective.

A first condition is Vuillemin's sequentiality condition, used in Vuillemin's optimality proof [25]. No algebraicity condition is needed.

Definition 4.4.4. A function $h: D_I^k \to D_I$ is *sequential* iff $\forall\alpha \in D_I^k$, $\exists i$, $1 \leq i \leq k$, such that $\beta \in D_I^k$, $\beta_i = \alpha_i$ and $\beta \supset \alpha$ implies $h(\beta) = h(\alpha)$. An interpretation I is *sequential* if all functions f^I are.

All indexes i satisfying the definition at a given α are called *sequentiality indexes* of h at α: To increase the output of h, one must increase its input on these indexes. Functions like **if then else**, constants, or usual arithmetic functions on the "flat integers" (see Introduction) are sequential. Vuillemin [25, 26] shows that composition and limits of sequential functions are also sequential.

The "parallel or" described above is not sequential, and M_Ω^∞ is *not* sequential since we have together $f(a, \Omega) \gneq f(\Omega, \Omega)$ and $f(\Omega, a) \gneq f(\Omega, \Omega)$: Here f has no sequentiality index on (Ω, Ω).

Notation 4.4.1. If c is an occurrence in $A \in M_\Omega^\infty$, we define the subterm $c \downarrow A$ of A at the occurrence c by

$$\epsilon \downarrow A = A,$$

$$(f, i)c' \downarrow f(A_1, A_2, \ldots, A_n) = c' \downarrow A_i.$$

LEMMA 4.4.2. *Let $a \in M_\Omega$, let I be sequential, and let $\nu : V \to D_I$. Then either $(I, \nu)A = (I, \nu)a$ for any $A > a$ or there exists c such that $c \downarrow a = \Omega$ and $c \downarrow A \neq \Omega$ for any $A > a$ such that $(I, \nu)\, a \subsetneq (I, \nu)A$.*

PROOF. By induction on $\|a\|$. The result is obvious is $\|a\| = 1$. Let $a = f(a_1, a_2, \ldots, a_n)$, and let k be a sequentiality index of f at $(I, \nu)a_1, (I, \nu)a_2, \ldots, (I, \nu)a_n$. Determine c_k for a_k by induction. Then $(f, k)c_k$ answers the questions. □

PROPOSITION 4.4.3. *Every sequential interpretation is projective.*

PROOF. Let I be sequential, $\nu : V \to D_I$ and $A \in M_\Omega^\infty$. Let $\alpha \subset (I, \nu)A$. We have two cases:

Case 1. $\exists a <$ A such that $a \supset \alpha$. Let $A' <$ A such that $\alpha \subset A'$. Then $a \cap A' \supset \alpha$; otherwise there exists, by Lemma 4.4.2, c such that $c \downarrow a \cap A' = \Omega$, $c \downarrow a \neq \Omega$, and $c \downarrow A' \neq \Omega$, which is impossible by definition of g.l.b. Since the set $\{a' < a \mid a' \supset \alpha\}$ is finite and closed by g.l.b., we have $c(\alpha, a) \supset \alpha$. But also $c(\alpha, A) = c(\alpha, a)$, so that $c(\alpha, A) \supset \alpha$.

Case 2. $\forall a < A$, $a \underset{\neq}{\overset{\supset}{}} \alpha$. Then $\alpha = (I, \nu)A$ and $c(\alpha, A) = c(A)$. From case 1 we deduce that $A' \equiv A$ and $a < A$ imply $A' > c(a)$ and that $c(a) \equiv a$. Then $c(A) > \cup\{c(a) \mid a < A\}$; hence $(I, \nu)c(A) = \cup\{(I, \nu)c(a) \mid a < A\} = \cup\{(I, \nu)a \mid a < A\} = (I, \nu)A$, which concludes the proof. □

Least computations of $\alpha \subset P(t)$ from t are easily constructed in Vuillemin's sequential interpretations: Build a derivation $t = t_0 \to t_1 \to t_2 \to \cdots \to t_n \to \cdots$ by selecting no occurrence in t_n if $\omega(t_n) \supset \alpha$ and by selecting an occurrence of an unknown function symbol which corresponds to an occurrence c in $\omega(t_n)$ determined as in Lemma 4.4.2 otherwise. Vuillemin's delay rule implements the corresponding complete derivation (see [25, 26]). We leave to the reader to see that this complete derivation $\bar\delta_k(t)$ has the following uniform optimality property:

$$\forall \bar d \in \mathscr{F}\mathscr{D}(t), \quad \exists k \geq 0, \quad (I,\nu)\bar d \subset (\mathrm{I}, \nu)\bar\delta_k(t) \quad \text{and} \quad \mathscr{F}(\bar\delta_k(t)) \subset \mathscr{F}(\bar d) \quad \text{and} \quad |\bar\delta_k(t)| \leq |\bar d|.$$

This property was indeed taken as the optimality definition in [25, 26]. However, it is specific to Vuillemin's sequential interpretations and is not true in the Herbrand interpretation: For $\Sigma : \phi(x) = x$, no computation of $f(\phi(x), \phi(x))$ has this property.

We now show that the class of stable interpretations defined and studied in [2, 3] is also projective.

Definition 4.4.5. Let D be an algebraic c.p.o. such that two elements α and β have a g.l.b. $\alpha \cap \beta$. Then $h : D^k \to D$ is *stable* iff $\forall \alpha, \beta \in D^k$, $\alpha \uparrow \beta$ implies $h(\alpha \cap \beta) = h(\alpha) \cap h(\beta)$. An interpretation I is stable iff every f^I is.

PROPOSITION 4.4.4. *Every stable interpretation is projective.*

PROOF. Show by induction on $\|a\|$ that $a \uparrow a'$ implies $(I, \nu)(a \cap a') = (I, \nu)\, a \cap (I, \nu)a'$ and apply Proposition 4.4.1. □

A particular class of stable interpretations is the class of Kahn and Plotkin's sequential interpretations on concrete date types [13]. This class contains M_Ω^∞ (which is not sequential in Vuillemin's sense).

The design of optimal computation rules for these interpretations is especially important in coroutine systems (see [12]), but is beyond the scope of this paper.

5. *Conclusion*

Only recursive programs schemes have been considered in this paper, with a single operation, i.e. the substitution of recursive function ocurrences. But in usual programming languages, there are more rules for evaluating expressions. For instance, there are rules for "**if then else**" or "+" .... These rules are called simplifications by Vuillemin. Here, we have treated them at a metalevel via projective interpretations. But a more realistic approach, especially for cost measures, would be to include them as rewriting rules. This means that one has to be able to handle more complicated rewriting rules than the one considered here. Thus, a unifying or axiomatized point of view should be adopted to preserve the syntactic results of this paper. This seems possible along the lines of O'Donnell [21] or Rosen [22]. Furthermore, the axiomatized approach should include the λ-calculus where much of the syntactic part of this paper is valid. Also for a practical outcome, the algorithms corresponding to the derivation strategies should be studied. For the formalism of this paper, a lazy evaluator with a delay mechanism is easily designed. But the implementation of such evaluators may not be so easy for other rewriting systems such as the λ-calculus.

REFERENCES

(Note. References [14, 16, 24, 27] are not cited in the text.)

1. Barendregt, H.P., Bergstra, J., Klop, J.W., and Volken, H. Degrees, reductions and representability in the λ-calculus. Math. Dept. Rep., U. of Utrecht, Utrecht, Netherlands, Jan. 1976.
2. Berry, G. Bottom-up computations of recursive programs. *R.A.I.R.O. Informatique Théorique 10*, 3 (March 1976), 47–82.
3. Berry, G. Les calculs minimaux et optimaux des programmes et leurs interprétations stables. To appear.
4. Birkhoff, G. Lattice theory. In Coll. Publications, Vol. 25, Amer. Math. Soc., Providence, R.I., 3rd ed., 1967.
5. Cadiou, J.M. Recursive definitions of partial functions and their computation. Ph.D. Th., Comptr. Sci. Dept., Stanford, U. Stanford, Calif., 1972.
6. Church, A. The calculi of lambda-conversions. Annals of Math. Studies, No. 6, Princeton U., Princeton, N.J., 1941.
7. Courcelle, B., and Nivat, M. Algebraic families of interpretations. Proc. 17th Annual Symp. Foundations Comptr. Sci., Houston, Tex., 1976.
8. Curry, H.B., and Feys, R. *Combinatory Logic, Vol. 1.* North-Holland Pub., Co., Amsterdam, 1958.
9. Downey, P.J., and Sethi, R. Correct computation rules for recursive languages. *SIAM J. Comptng. 5*, 3 (Sept. 1976), 378–401.
10. Henderson, P., and Morris, J.H. A lazy evaluator. Conf. Rec. 3rd ACM Symp. Principles of Programming Languages, Jan. 1976, pp. 95–103.
11. Hindley, R. Reductions of residuals are finite. Math. Dept., U. of Swansea, Swansea, England, 1976.
12. Kahn, G., and McQueen, D. Coroutines and networks of parallel processes. Information Processing 77, North-Holland Pub. Co., Amsterdam, 1977, pp. 993–998.
13. Kahn, G., and Plotkin, G. Concrete data types. To appear.
14. Lévy, J.J. An algebraic interpretation of the $\lambda\beta K$-calculus and a labelled λ-calculus. *Theoret. Comptr. Sci. 2*, 1 (1976), 97–114.
15. Lévy, J.J. Réductions correctes et optimales dans le λ-calcul. Th. de Doctorat d'Etat, Université Paris VII, Paris, 1978.
16. Milner, R. Fully abstract models of typed λ-calculi. To appear in *Theoret. Comptr. Sci.*
17. Mitschke, G. *The standardisation theorem for the λ-calculus.* Math. Dept., U. of Swansea, Swansea, England, 1975.
18. Montangero, C., Pacini, G., and Turini, F. Graph representation and computation rules for typeless recursive program schemes. LNCS 14, Proc. 2nd Colloq. Automata, Languages and Programming, Saarbrücken, Springer-Verlag, 1974.
19. Morris, J.H. Lambda calculus models of programming languages. Ph.D. Th., M.I.T., Cambridge, Mass., 1968.
20. Nivat, M. On the interpretations of recursive programs schemes. In *Symposia Matematica, Vol. XV.* Inst. Nazionale di Alta Matematica, Rome, Italy, 1975, pp. 225–281.
21. O'Donnell, M. Reduction strategies in subtree replacement systems. Ph.D. Th., Cornell U., Ithaca, N.Y., 1976.
22. Rosen, B.K. Tree-manipulating systems and Church-Rosser theorems. *J. ACM 20*, 1 (Jan. 1973), 160–187.
23. Scott, D. Outline of a mathematical theory of computation. Programming Research Group, Mono. No. 2, Oxford, 1970.
24. Trakhtenbrot, M.B. On representation of sequential and parallel functions. Computational Mathematics and Programming (collected papers), Novosibirsk, USSR, 1974.

25. VUILLEMIN, J. Proof techniques for recursive programs. Ph.D. Th., Comptr. Sci. Dept., Stanford U., Stanford, Calif., 1973.
26. VUILLEMIN, J. Syntaxe, sémantique et axiomatique d'un langage de programmation simple. Th. de Université de Paris VII, 1974.
27. WADSWORTH, C.P. Semantics and pragmatics of the λ-calculus. Ph.D. Th., Oxford U., Oxford, England, 1971.

RECEIVED JULY 1977; REVISED FEBRUARY 1978