(Optimal) Duplication is not Elementary Recursive

ANDREA ASPERTI
Università di Bologna
and
PAOLO COPPOLA and SIMONE MARTINI
Università di Udine

alla fine

Additional Key Words and Phrases: lambda-calculus, optimal reduction,graph rewriting ,complexity, elementary affine logic

1. INTRODUCTION

In the last ten years there has been a steady interest in *optimal reduction* of λ-terms (or, more generally, of functional programs). The very story started, in fact, more than twenty years ago, with Jean-Jacques Lévy's thesis [Lévy 1978]. The problem he attacked was to find an execution strategy for λ-terms minimizing the number of β-reductions. It was already known that no such strategy exists which reduces only one redex (i.e., a single function call) at the time. Lévy's insight was to discover that a *parallel* optimal strategy exists, the one reducing in a single step all the redexes belonging to the same *family*, a crucial notion he introduced. However, Lévy lacked the programming technology to implement his strategy. He showed it was effective, but no one knew at that time what kind of data structure could be used to dynamically maintain the families in such a way that all the redexes of a given family could be somehow shared and, therefore, reduced as a single step. The solution came in 1989, when independently Kathail [Kathail 1990] and

—Preliminary version
—grants...

Name: Andrea Asperti
Address: Via di Mura Anteo Zamboni, 40127 Bologna, Italy
Affiliation: Dipartimento di Scienze dell'Informazione, Università di Bologna
Name: Paolo Coppola and Simone Martini
Address: via delle Scienze 206, 33100 Udine, Italy
Affiliation: Dipartimento di Matematica e Informatica, Università di Udine

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works, requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept, ACM Inc., 1515 Broadway, New York, NY 10036 USA, fax +1 (212) 869-0481, or `permissions@acm.org`.

Lamping [Lamping 1990] gave abstract λ-calculus machines which reduced terms as prescribed by Lévy's optimality theory. Lamping's graph rewriting approach is the one that received most interest, and after his breakthrough other variants of optimal reducers have been proposed, especially by Gonthier, Abadi and Lévy [Gonthier et al. 1992a] and Asperti [Asperti 1995]. We will refer to all of them as the *optimal sharing graph approach*.

All these variants share a common core – the *abstract algorithm* in the terminology of [Asperti and Mairson 1998] – and differ in the way they implement the bookkeeping work needed to maintain the families. The algorithms are described as elegant graph rewriting systems, where any rule rewrites only a pair of facing nodes (and then it can be easily implemented as a constant time operation). The abstract algorithm is responsible for the performing of the *shared β-rule* and for the incremental *duplication* of subterms. It is the job of the bookkeeping part – also called the *oracle* – to maintain in the graph enough distributed information to make the abstract algorithm correct with respect to the standard β-reduction (see Section 2, or [Asperti and Guerrini 1998] for a complete account).

Since all the reduction rules can be implemented as constant time operations, we may take the number of rewriting steps to reach the normal form of a term as the cost of the reduction algorithm. Lévy's theory, given a λ-term M, prescribes the number of shared β-reductions needed to reach the normal form. All optimal sharing graph algorithms perform exactly this number of β-rewritings, but the global number of reductions is greater, since the algorithms also perform both duplication (in the abstract algorithm) and bookkeeping. Is it possible to bound the total work as a (fixed) function of the number of shared β-reductions? It is this question that has been behind several contributions by Asperti, Lawall, and Mairson [Asperti 1996; Lawall and Mairson 1996; Lawall and Mairson 1997], culminated in [Asperti and Mairson 1998]. Define the Kalmár elementary functions $\mathbf{K}_\ell(n)$ as $\mathbf{K}_0(n) = n$ and $\mathbf{K}_{\ell+1}(n) = 2^{\mathbf{K}_\ell(n)}$. Then [Asperti and Mairson 1998] shows that there are λ-terms that can be normalized with n shared β-reductions, and for which the total work needed to reach the normal form with any algorithm (and hence with any optimal sharing graph algorithm) exceeds $\mathbf{K}_\ell(n)$ for any fixed $\ell \geq 0$.

The theorem shows that optimal β reduction cannot be realized as a unit cost operation. However, it does not mean that optimal reduction, as a whole, is unfeasible. By a classical theorem of Statman [Statman 1979], any λ-calculus machine has to take non elementary time in the size of a term to reach the normal form. In the case of optimal reduction, the non elementary bound (as a function of the number of shared β-reductions) is just a consequence of the fact that with this sharing technique the number of redex families is surprisingly low (essentially linear in the size of the term). Hence the most part of the work – that has to be done in any case, by Statman's theorem – is devolved to duplication and bookkeeping. Still, in the case of Lamping's algorithm, it remained the problem to understand if this cost was essentially due to duplication, or to the (pretty complex) bookkeeping technique. In this paper, we prove that already the mere cost of duplication is not elementary. In other words, it is not only Lamping's bookkeeping technique to be unfeasible, but already the plain work of duplication. This is particularly interesting since, by now, there is some good evidence that the number of duplications performed

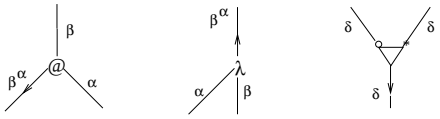

Fig. 1. Lamping's (abstract) algorithm nodes.

by Lamping's algorithm is a lower bound to the complexity of any (sub-optimal[1]) reduction technique [Lawall and Mairson 1997; Asperti and Laneve 1997].

As a working tool, we use in this paper Elementary Affine Logic [Asperti 1998] (EAL), a variant of Elementary Linear Logic, introduced in [Girard 1998]. In particular, we show that the simply typed λ-terms used in [Asperti and Mairson 1998] to code a certain decision problem can be given a type also in EAL. A crucial feature of the λ-terms typeable in EAL is that their reduction does not need the bookkeeping part of the sharing graph algorithm. Therefore, we can refine the first corollary to the main theorem in [Asperti and Mairson 1998] in the following manner:

> *There exists a set of λ-terms E_n : Bool which normalize in no more than n parallel β-steps, where the number of ~~bookkeeping~~ duplication interactions that are required to normalize E_n using Lamping's graph reduction algorithm grows as $\Omega(\mathbf{K}_\ell(n))$ for any fixed integer $\ell \geq 0$.*

The reader may be confused by the outward paradox of typing non elementary terms in an elementary logic. There is no paradox indeed, since the complexity of reducing terms inside EAL is elementary once the depth of the term is fixed. In our case, the depth of the involved terms grows polynomially in the size of the input.

The paper is organized as follows. In Section 2 we introduce Lamping's graph reduction technology. Section 3 presents Elementary Affine Logic, analyzing the complexity of its cut-elimination and discussing the problem of type inference. Section 4 recalls Asperti and Mairson's result and sketches how to type their terms in EAL, obtaining our main result.

2. OPTIMAL REDUCTION

We assume the reader to have some familiarity with the simply typed λ-calculus. We only recall that the set of types is defined inductively by the grammar $\alpha ::= o \mid \alpha \to \alpha$, where o is a fixed base type, and the (untyped) terms are generated by the grammar $M ::= x|(M\ M)|\lambda x.M$. We assume to have an enumerable set of variables, ranged over by x. Types can be assigned to terms in the usual way. Reduction is defined by the β-rule: $(\lambda x.M\ N) \to M[N/x]$. We write $\alpha_1 \to \alpha_2 \to \alpha_3$ for $\alpha_1 \to (\alpha_2 \to \alpha_3)$ and $(M_1\ M_2\ M_3)$ for $((M_1\ M_2)\ M_3)$.

While the details of Lamping's approach for optimal reduction are complex and technical (see the book [Asperti and Guerrini 1998]), the few concepts needed to understand the result of this paper are easy to grasp. The first step in Lamping's

[1] Theoretically, there could be super-optimal implementations, where one *dynamically* looks for common subexpressions. Levy's theory of optimality does not cover these cases, which are neglected as a form of "syntactical coincidence".

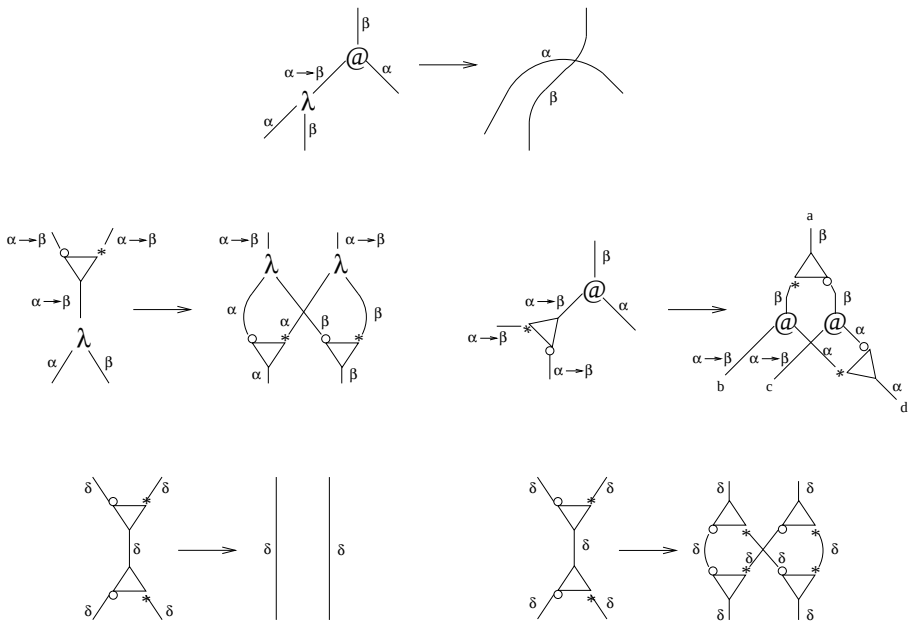

Fig. 2. Rules of the abstract algorithm.

(abstract) algorithm is to represent a term as a labelled graph built out of the nodes given in Figure 1. The arrow exiting a node is the *principal port* of the node. The graph is obtained from the syntax tree of the term by: (i) representing a variable by an arc (a *wire*); (ii) adding an explicit node (the triangular node, called *fan*) for the sharing of a variable occurrence; and (iii) connecting the single wire representing a variable (after sharing) to the node λ which binds the variable. Reduction of the term is represented by graph rewriting. The (typed) rules of the abstract algorithm are displayed in Figure 2. Note that two nodes are rewritten (they *interact*) only when their principal ports face each other. The first rule is the *shared β-rule*; the others, as it should be clear from their shape, perform an incremental (and controlled, to get optimality) duplication of the graph. When the duplication is over, the first of the two fan-fan rules annihilate the fans (intuitively, the two fans originate from the same node in the starting graph); otherwise, in the last rule, one fan duplicates the other. Observe, however, that this presentation of the algorithm is non deterministic, since the two fan-fan rules have the same left-hand side. To make the algorithm deterministic, one may think to label the fans in the original graph and then, throughout the reduction, apply the first fan-fan rule when the two fans have the same label; apply the other otherwise. It is one of the crucial Lamping's observations that this simple technique does not work for the simply typed (and, a fortiori, the type free) terms. A much more complex *bookkeeping* oracle machinery have to be adopted, adding more nodes (brackets, croissants) to the graphs and decorating any node with an index. The full algorithm, therefore, consists of both the abstract algorithm we just sketched and its bookkeeping part, which we omit. We will introduce in the next section a formal system to type terms

$$A \vdash A \ (Ax) \qquad \frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash B} \ (Cut)$$

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C} \ (Perm.) \qquad \frac{\Gamma \vdash C}{\Gamma, !A \vdash C} \ (Weak.)$$

$$\frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B} \ (Contr.) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \ (\multimap, r)$$

$$\frac{\Gamma \vdash A \quad B, \Delta \vdash C}{\Gamma, A \multimap B, \Delta \vdash C} \ (\multimap, l) \qquad \frac{A_1, \ldots, A_n \vdash B}{!A_1, \ldots, !A_n \vdash !B} \ (!)$$

$$\frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B} \ (\epsilon) \qquad \frac{\Gamma, !!A \vdash B}{\Gamma, !A \vdash B} \ (\delta)$$

Fig. 3. Intuitionistic Linear Logic

Fig. 4. Bracket and croissant.

for which the simple labelling of fans *is enough* to get correctness.

3. ELEMENTARY AFFINE LOGIC

3.1 The logical system

Linear Logic [Girard 1987] provides a logical interpretation of Lamping's algorithm as a cut-elimination process for a suitable notation for proofs (proof-nets). This was first hinted at in [Gonthier et al. 1992a] and then developed in [Gonthier et al. 1992b]. We give here an intuitive introduction to the subject, to motivate the use of EAL as a tool towards our result; see, e.g., Chapters 4 and 8 of [Asperti and Guerrini 1998] for a full account on the subject.

Figure 3 presents (in a somewhat non standard way) the intuitionistic fragment of Linear Logic (ILL).

It is a modal system, where the modality (!) is used to mark those formulas on which contraction and weakening may be applied. The standard implicational fragment of Intuitionist Logic may be embedded into ILL via the interpretation $A \to B \equiv !A \multimap B$. By this, and the Curry-Howard isomorphism, for any typed λ-term we may give a corresponding ILL proof. Such proofs may be expressed as graphs (proof-nets), essentially corresponding to the syntax tree of the λ-terms but for the presence of !, which is expressed by the so-called *boxes*. Boxes are given in standard ILL proof-nets as global entities (they are certain subgraphs with specified nodes as interfaces), but they can also be described in a local way, by introducing new nodes (and node indexes) besides the ones corresponding to the connectives of the logic. These new nodes, corresponding to the rules ϵ and δ, are the nodes needed in the oracle of Lamping's algorithm–brackets and croissants, see Figure 4.

A typed term M is translated into an optimal sharing graph $[M]_0$ according

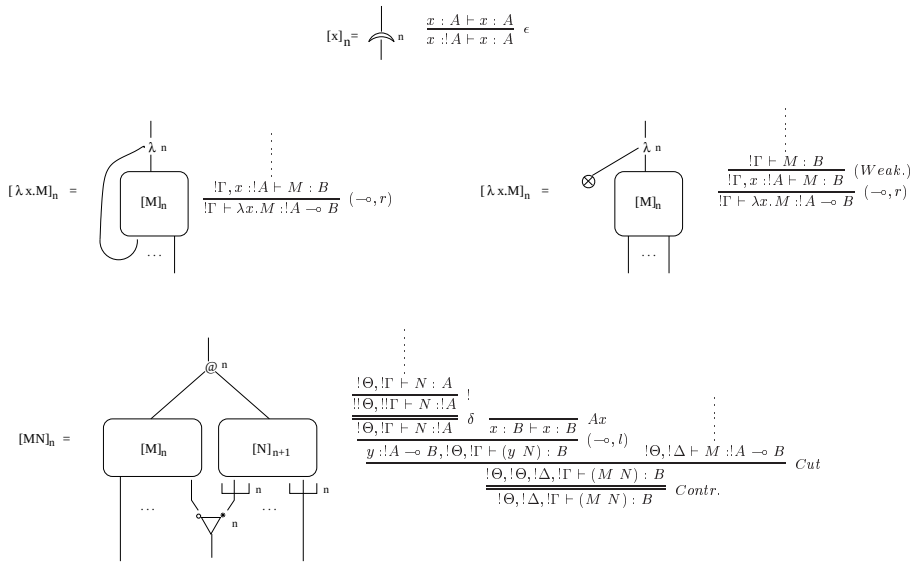

Fig. 5. $[.]_n$ translation.

to the inductive rules of Figure 5, where also the corresponding ILL derivations are given. The rightmost case corresponds to the abstraction of a variable not appearing in the body; the node $\otimes$ is called *eraser* and can be thought of as a fan with zero premises. Observe, in the application case, how the !-rule is translated into a box: the level of the translation increases from $[N]_i$ to $[N]_{i+1}$.

The rationale of this translation is the following. Since contraction (fan) may be performed only on !-formulas, the translation of a variable puts a ! on any variable, getting ready for a *possible* contraction. In the translation of an application, we put a box around the argument N (!-rule) and then, since there is now one more ! in the type of the free variables of N, we apply the rule δ (the bracket), finally contracting the common variables. In view of the isomorphism $A \to B \equiv !A \multimap B$, this technique allows the translation of any typed term (and also of untyped terms, provided we see them as having type $O \equiv O \multimap O$).

Restrict now the logical system, removing the rules ϵ and δ. We cannot any longer prepare to contract any variable as we did in the translation of Figure 5. The contractible variables will be only the ones that get a ! in their type by means of rule !. And we cannot any longer decrease the number of ! in the type of a variable, since rule δ is no longer there. As a reward, however, there are no brackets and croissants in the translation of any λ-term. We moved from the complete Lamping's algorithm to the abstract one. It remains only to be seen how the application of the two fan-fan rules may be discriminated. It not difficult to see that, in this framework, the level of a node never changes during reduction. Therefore, we may apply the first of the two fan-fan rules (annihilate) if the two fans have the same level; we apply the second one otherwise. In this restricted system, levels behave like labels, and these labels are sufficient to ensure correctness of reduction. If a λ-term M can be typed inside the restricted system obtained by omitting the rules

$$x : A \vdash x : A \ (Ax) \qquad \frac{\Gamma \vdash M : A \quad x : A, \Delta \vdash N : B}{\Gamma, \Delta \vdash N[M/x] : B} \ (Cut)$$

$$\frac{\Gamma \vdash M : C}{\Gamma, x : A \vdash M : C} \ (Weak.) \qquad \frac{\Gamma, x :!A, x :!A \vdash M : B}{\Gamma, x :!A \vdash M : B} \ (Contr.)$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B} \ (\multimap, r) \qquad \frac{\Gamma, x : A, y : B, \Delta \vdash M : C}{\Gamma, y : B, x : A, \Delta \vdash M : C} \ (Perm.)$$

$$\frac{x_1 : A_1, \ldots, x_n : A_n \vdash M : B}{x_1 :!A_1, \ldots, x_n :!A_n \vdash M :!B} \ (!) \qquad \frac{\Gamma \vdash N : A \quad x : B, \Delta \vdash M : C}{\Gamma, f : A \multimap B, \Delta \vdash M[(f\ N)/x] : C} \ (\multimap, l)$$

Fig. 6. Elementary Affine Logic

ϵ and δ, then the (optimal) reduction of M can be performed without any need of the bookkeeping.

The λ-terms which can be given a type in this restricted system, therefore, are the natural choice if we want to bound the duplication work only.

What we have called so-far the restricted system, is a (fragment of) Elementary Linear Logic (ELL), sketched by Girard in [Girard 1998] as part of a work directed to the logical characterization of polynomial and elementary functions. ELL completely describes the elementary functions. From one side, any elementary function can be coded as a proof of ELL; from the other, normalization of ELL proofs is an elementary time procedure, once the box-nesting depth of the proof is fixed.

A more flexible syntax (also for a polynomial logic, LAL) was introduced by one of the authors in [Asperti 1998], by allowing full weakening (see also [?] for further variants of LAL). The resulting system, Elementary Affine Logic (EAL), is presented in Figure 6, where also λ-terms have been added to the rules. We say that *a term M has an EAL type A under the context* Γ iff $\Gamma \vdash M : A$ is derivable in the system of Figure 6. We remark that these terms are not intended as a notation for proofs in EAL; terms for this purpose can be easily obtained from those for LAL in [Asperti 1998]. Here we use EAL just as a typing system for pure λ-terms. The following is subject reduction.

FACT 1. *If* $\vdash M : A$ *and* $M \to_\beta N$, *then* $\vdash N : A$.

Although this may be non obvious at first, there are simply typed terms for which no EAL type can be derived. An example will be given at the end of Section 3.2.

Like ELL, also EAL characterizes the elementary functions: Any elementary function can be represented in EAL, and the normalization procedure is elementary time.

Finally, and this is the crucial issue we need in the following, we stress again that if M gets an EAL type, then M can be reduced by the abstract algorithm by a simple labelling of the fan nodes.

3.2 Decorating terms

It should be clear from the rules of EAL that a derivation of a type in EAL (an *EAL-type*, from now on) for a λ-term M consists of a *skeleton* – given by the derivation of a type for M in the simple type discipline – together with a *box decoration*, introducing a suitable number of !-rules. Such modalities are needed

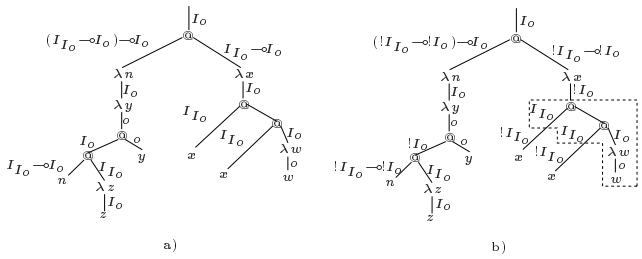

Fig. 7. Type inference in EAL, I

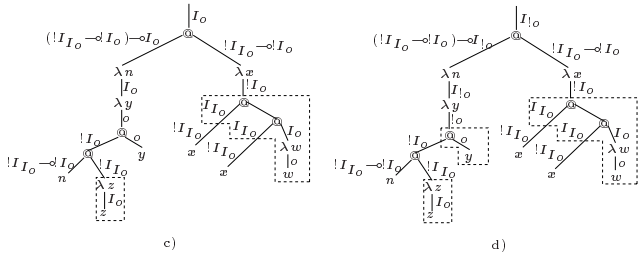

Fig. 8. Type inference in EAL, II

since only !-typed variables can be contracted. Observe, moreover, that a given (simply typed) skeleton has an infinite number of possible decorations, and not all of them are instances of a single, most general one.

Therefore, while the skeleton is trivially obtained, finding the right decoration is the hard part, since the introduction of a box in a portion of the proof forces other boxes to be introduced somewhere else. **Modificare. Citare tlca.** *We conjecture type inference in EAL decidable, but we have not a complete proof of this yet. The typing of the relevant terms in Section 4 is therefore obtained using some heuristics, which we describe in this section.*

We may single out three main steps in the process of type inference: "looking for contractions", "boxing arguments" and "opening boxes". We discuss these steps by going through an easy example.

Let

$$N{=}(\lambda n.\lambda y.((n\ \lambda z.z)\ y)\ \lambda x.(x\ (x\ \lambda w.w)))$$

be the simply typed term to be typed in EAL. We start from the syntax tree of the term, labelled with the types of the simple discipline (just changing $\to$ into $\multimap$) as in Figure 7 (a), where $I_\alpha{=}\alpha \multimap \alpha$ for every type α.

Now *look for contractions*. If all the variables occur only once – they are used linearly – we are done. This is not the case in our example, since x occurs twice in $M{=}\lambda x.(x\ (x\ \lambda w.w))$. As contraction in EAL is admitted only for formulas of type $!\alpha$, we need to introduce a ! before the abstraction of x. Using the usual sequent calculus notation, the *simple* type derivation of M in Figure 7 (a) corresponds to

the following derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{o \vdash o}\,(Ax)}{\vdash I_o}\,(\multimap,r) \quad \dfrac{}{I_o \vdash I_o}\,(Ax)}{I_o \multimap I_o \vdash I_o}\,(\multimap,l) \quad \dfrac{}{I_o \vdash I_o}\,(Ax)}{\mathbf{I_{I_o}, I_{I_o} \vdash I_o}}\,(\multimap,l)}{\mathbf{I_{I_o} \vdash I_o}}\,Contr?}{\vdash I_{I_o} \multimap I_o}\,(\multimap,r)$$

which is not in EAL because the contraction rule is wrong. To obtain a correct EAL derivation we add a !-rule before contraction:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{o \vdash o}\,(Ax)}{\vdash I_o}\,(\multimap,r) \quad \dfrac{}{I_o \vdash I_o}\,(Ax)}{I_o \multimap I_o \vdash I_o}\,(\multimap,l) \quad \dfrac{}{I_o \vdash I_o}\,(Ax)}{I_{I_o}, I_{I_o} \vdash I_o}\,(\multimap,l)}{!I_{I_o}, !I_{I_o} \vdash !I_o}\,(!)}{!I_{I_o} \vdash !I_o}\,(Contr.)}{\vdash !I_{I_o} \multimap !I_o}\,(\multimap,r)$$

The corresponding typing is represented in Figure 7 (b), where the type of x inside the box is I_{I_o}, whereas it is $!I_{I_o}$ outside. The new type for M, however, needs to be propagated in the left branch of the tree for the full term N, or otherwise the topmost application would have the wrong type. As a consequence, the variable n in Figure 7 (b) gets type $!I_{I_o} \multimap !I_o$.

Observe now that the leftmost innermost application is wrong. We need to *box the argument* $\lambda z.z$, which must have type $!I_{I_o}$, as it is shown in Figure 8 (c).

Finally, in order to apply $(n\ \lambda z.z)$ of type $!I_o = !(o \multimap o)$ to y of type o, we need to *open the box*, as in Figure 8 (d). As mentioned above, types inside boxes "lose" one !, in particular $!(o \multimap o)$ becomes $o \multimap o$, allowing us to perform the application.

Observe how a single contraction inside the term M forced us to introduce boxes all over the tree.

The final decoration of Figure 8 (d) represents the following derivation in EAL:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{I_o \vdash I_o}\,(Ax)}{\vdash I_{I_o}}\,(\multimap,r) \quad \dfrac{\dfrac{}{o \vdash o}\,(Ax) \quad \dfrac{}{o \vdash o}\,(Ax)}{I_o, o \vdash o}\,(\multimap,l)}{\vdash !I_{I_o}}\,(!) \quad \dfrac{}{!I_o, !o \vdash !o}\,(!)}{!I_{I_o} \multimap !I_o, !o \vdash !o}\,(\multimap,l)}{!I_{I_o} \multimap !I_o \vdash I_{!o}}\,(\multimap,r)}{\vdash (!I_{I_o} \multimap !I_o) \multimap I_{!o}}\,(\multimap,r) \quad \begin{array}{c}\mathcal{E}\\ \vdots \\ \vdash !I_{I_o} \multimap !I_o\end{array}}{\vdash I_{!o}}\,(Cut)$$

where $\mathcal{E}$ is the derivation given above for the subterm M.

As already mentioned, other decorations for N are possible. First, we may give N the type $!^n I_{!o}$, by adding n !-rules at the end of the derivation. Or, to be more general, we may give N the type $!^n I_{!^m o}$, if we introduce $m \geq 1$!'s before the abstraction in the derivation of M. But there are other possibilities. We may choose to introduce m !'s ("close" m boxes) in Figure 8 (d) after the abstraction of y, obtaining for N the type $!^{n+m} I_o$.

Finally we remark that type inference in EAL is not always possible. For example the simply typed λ-term

$$(\lambda n.(n\ \lambda y.(n\ \lambda z.y))\ \lambda x.(x\ (x\ y)))$$

has no EAL decoration. To see this in a simple way, write the term as a sharing graph and reduce it in the abstract algorithm by matching fans by labels. The sharing graph in normal form is *a cycle*, that is a sharing graph which does not correspond to any λ-term (least to say to y, which is the normal form of the given term). This means that the oracle *is* needed for the reduction of this term, and hence it cannot have a type in EAL.

4. CODING TYPE THEORY INTO EAL PROOFS

4.1 Asperti and Mairson's result

The result of Asperti and Mairson [Asperti and Mairson 1998] is obtained out of three building blocks. The first, and most novel, contribution is that any simply typed term can be "pre-compiled" in a certain way in order to drastically limit the number of its shared β-reductions.

Consider the size of types as the structural size, the size of λ-terms counting 1 for each abstraction and application and counting the size of the type for each variable and finally consider the size of sharing graphs as the number of nodes.

Let x be a variable of type σ. The *η-expansion* $\eta_\sigma(x)$ of x is defined inductively on σ as follows:

$$\begin{aligned}\eta_o(x) &= x\\ \eta_{\alpha_1\to\cdots\to\alpha_n\to o}(x) &= \lambda y_1:\alpha_1.\ldots\lambda y_n:\alpha_n.\\ &\qquad (x\ \eta_{\alpha_1}(y_1)\ \ldots\ \eta_{\alpha_n}(y_n))\end{aligned}$$

Given a simply typed lambda-term M, following the procedure in [Asperti and Mairson 1998], we construct a $\beta\eta$-equivalent variant $\mathbf{or}(M)$ – the *optimal root* of M – replacing every subterm of the form $\lambda x:\sigma.E$, where $\sigma \neq o$, with $\lambda x':\sigma.(\lambda x:\sigma.E\ \eta_\sigma(x'))$. During this process we introduce a certain number – clearly linearly bounded by the size of M – of new *preliminary* β-redexes. $\Delta(M)$ is the sharing graph obtained from $\mathbf{or}(M)$ by reducing all the new preliminary redexes and propagating all sharing nodes to the base type.

THEOREM 1. *For any simply-typed λ-term M, the total number of shared β-reductions in the graph normalization of $\Delta(M)$ is limited by the size of $\Delta(M)$.*

The second ingredient is obtained from Mairson's proof [Mairson 1992] of theorems of Statman and Meyer [Statman 1979; Meyer 1974].

Define $\mathcal{D}_1 = \{\mathbf{true}, \mathbf{false}\}$, and $\mathcal{D}_{k+1} = \mathit{powerset}(\mathcal{D}_k)$. The decision problem for propositional calculus can be naturally generalized to higher-order types by allowing variables and quantifiers to range over values of $\mathcal{D}_k$, for $k \geq 1$. Let x^k, y^k, z^k be variables ranging over $\mathcal{D}_k$; we define *prime formulas* as **true**, **false**, $\mathbf{true} \in y^2$, $\mathbf{false} \in y^2$, and $x^k \in y^{k+1}$. Finally, let a *formula* Φ be built up out of prime formulas, the propositional connectives $\wedge$, $\vee$, $\neg$, and the quantifiers $\forall$ and $\exists$. Statman showed how to reduce the truth of a higher-order formula to the reduction to a given normal form of a suitable typed term. Mairson [Mairson 1992] showed

how to simplify the proof of Statman with a different encoding based on the same basic idea – the quantifier elimination procedure – but much simpler and easy to understand for the use of *list iteration* as a quantifier elimination procedure.

THEOREM 2. *A higher-order formula Φ is true if and only if its typed λ-calculus interpretation $\hat{\Phi} : \mathsf{Bool}$ is $\beta\eta$-equivalent to $\mathtt{true} : \mathsf{Bool}$. Moreover, if Φ only quantifies over universes $\mathcal{D}_i$ for $i \leq k$, then $\hat{\Phi}$ has order at most k, and $|\hat{\Phi}| = O(|\Phi|(2k)!)$.*

Finally, the last step is to show that any elementary time-bounded Turing machine can be encoded into higher-order logic, refining the proof in [Mairson 1992].

THEOREM 3. *Let M be a fixed Turing machine that accepts or rejects an input x in $\mathbf{K}_\ell(|x|)$ steps. Then there exists a formula Φ_x in higher-order logic such that M accepts x iff Φ_x is true. Moreover, Φ_x only quantifies over universes $\mathcal{D}_i$ for $i \leq (\log^* |x|) + \ell + 6$, and has length $O(|x| \log^* |x|)$.*

From the Theorems 1, 2, and 3 the main result of [Asperti and Mairson 1998] is obtained easily. In this paper, we show that Theorems 1 and 2 hold for EAL-typed λ-terms. Then, combining these results with Theorem 3 and the observation we made at the end of Section 3.1, we derive our main result.

THEOREM 4. *There exists a set of λ-terms $E_n : \mathsf{B}$ which normalize in at most n shared β-reductions, where the number of non β-interactions that are required to normalize E_n using Lamping's* abstract *algorithm grows as $\Omega(\mathbf{K}_\ell(n))$ for any fixed integer $\ell \geq 0$.*

4.2 Coding higher-order logic in EAL

We show in this section how higher-order logic can be coded with EAL-typed λ-terms. The (type-free) λ-terms we use are minor variants of those of [Asperti and Mairson 1998], the main technical contribution being the type-inference inside EAL.

REMARK 1. *The modifications to the encoding of [Asperti and Mairson 1998] are the following: (i) we use different terms for the encoding of equality; and (ii) the variable x_1 is not a prime formula. As a consequence of (ii), one has to adopt also a slightly different encoding of a Turing Machine in the proof of Theorem 3. In particular, define $x^1 < y^1 = \exists x^2.true \in x^2 \wedge false \notin x^2 \wedge y^1 \in x^2 \wedge x^1 \notin x^2$ and $x^1 = y^1 = \neg(x^1 < y^1 \vee y^1 < x^1)$.*

Define the type of Booleans as $\mathsf{B} = !o \multimap !o \multimap !o$. We write L^τ_α for the EAL type of generic lists of elements of type α:

$$L^\tau_\alpha = !(\alpha \multimap \tau \multimap \tau) \multimap !(\tau \multimap \tau).$$

Following [Mairson 1992], quantifiers can be encoded by using iteration over lists. Given $n \geq 0$ and some EAL type σ, suppose to have coded with the λ-terms $\hat{Q} : L_\sigma^{!^n\mathsf{B}}$ the set $Q = \{e_1, \ldots, e_m\}$ of elements of type σ; suppose moreover that $\hat{\Psi} : \sigma \multimap !^n\mathsf{B}$ is a term encoding a generic formula Ψ. Then $(\hat{Q}\ \lambda z.(\mathtt{AND}\ (\hat{\Psi}\ z))\ \mathtt{true})$ is the term encoding the formula $\forall z \in Q\ \Psi$. Encoding of prime formulas can be understood as the λ-calculus translation of the following inductive definitions:

$$\begin{aligned} x^1 =_1 y^1 &= x^1 \leftrightarrow y^1 \\ x^{k-1} \in_k y^k &= \exists z^{k-1} \in y^k\ z^{k-1} =_{k-1} x^{k-1} \\ x^k =_k y^k &= \forall z^{k-1} \in \mathcal{D}_{k-1}\ (z^{k-1} \in_k x^k \leftrightarrow z^{k-1} \in_k y^k). \end{aligned}$$

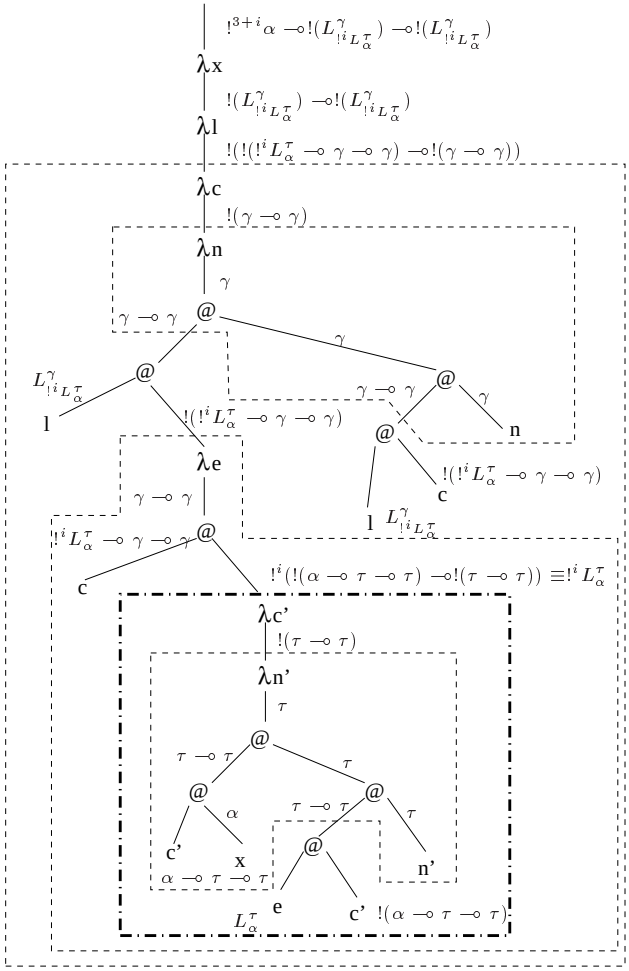

Fig. 9. EAL type of *double*

The quantified formulas $\exists z^{k-1} \in y^k$ in the definition of $\in_k$, and $\forall z^{k-1} \in \mathcal{D}_{k-1}$ in the definition of $=_k$, are encoded by list iteration, as described above.

DEFINITION 1 (SIZE OF TERMS). *Size of* simply typed *lambda terms is defined inductively as follows:*

$$\begin{aligned} ||o|| &= 1 \\ ||\alpha \to \beta|| &= 1 + ||\alpha|| + ||\beta|| \end{aligned}$$

$$\begin{aligned} |x| &= ||\sigma|| \text{ if } x \text{ has type } \sigma \\ |\lambda x.M| &= 1 + |M| \\ |(M\ N)| &= 1 + |M| + |N|. \end{aligned}$$

Given the EAL-typed term $M : A$, size of M is defined as the size of the simply typed term obtained by the usual erasure function $()^*$ applyed to A.

DEFINITION 2 (DEPTH). *Given an EAL-type A,* depth $d(A)$ *is defined inductively as follows:*

$$\begin{aligned} d(o) &= 0 \\ d(A \multimap B) &= \max\{d(A), d(B)\} \\ d(!A) &= 1 + d(A). \end{aligned}$$

Depth $d(M)$ of an EAL-typed term M is the maximum number of nested boxes in the EAL-derivation of M.

In Table 1 is summarized the full encoding with EAL-types, size of terms and depth of EAL-derivations.

NOTE 1 (TO TABLE 1). *(*) Sizes of double and powerset depend on types α, γ, τ and are relevant only to the calculus of $\mathbf{D_k}$. Depth of eq_k is relevant only to the calculus of depth of $member_k$.*
(**) Δ_1 and Δ_k are defined below in Definition 3. M_k is defined in Lemma 8. E^k is definded in the proof of Lemma 8. d is the constant of Theorem 4.1 in [Asperti and Mairson 1998]. $f()$ is defined in Theorem 5.

LEMMA 1 (BOOLEANS). *We can encode booleans in EAL as follows:*

(1) $\mathtt{true} = \lambda x\ y.x :!^n\mathtt{B}$ *for any $n \geq 0$, $|\mathtt{true}| = 3$, $d(\mathtt{true} :!^n\mathtt{B}) = n$ and the deepest type that appears in the EAL-derivation of* $\mathtt{true} :!^n\mathtt{B}$ *is $!^n\mathtt{B}$, of depth $d(!^n\mathtt{B}) = n + 1$;*

(2) $\mathtt{false} = \lambda x\ y.y :!^n\mathtt{B}$ *for any $n \geq 0$, $|\mathtt{false}| = 3$, $d(\mathtt{false} :!^n\mathtt{B}) = n$, deepest type $!^n\mathtt{B}$ of depth $d(!^n\mathtt{B}) = n + 1$;*

(3) $\mathtt{NOT} = \lambda b\ x\ y.(b\ y\ x) :!^n\mathtt{B} \multimap !^n\mathtt{B}$ *for any $n \geq 0$, $|\mathtt{NOT}| = 12$, $d(\mathtt{NOT} :!^n\mathtt{B} \multimap \cdots) = n$, deepest type $!^n\mathtt{B}$ of depth $d(!^n\mathtt{B}) = n + 1$;*

(4) $\mathtt{AND} = \lambda b_1\ b_2\ x\ y.(b_1\ (b_2\ x\ y)\ y) :!^n\mathtt{B} \multimap !^n\mathtt{B} \multimap !^n\mathtt{B}$ *for any $n \geq 0$, $|\mathtt{AND}| = 21$, $d(\mathtt{AND} :!^n\mathtt{B} \multimap \cdots) = n$, deepest type $!^n\mathtt{B}$ of depth $d(!^n\mathtt{B}) = n + 1$;*

(5) $\mathtt{OR} = \lambda b_1\ b_2\ x\ y.(b_1\ x\ (b_2\ x\ y)) :!^n\mathtt{B} \multimap !^n\mathtt{B} \multimap !^n\mathtt{B}$ *for any $n \geq 0$, $|\mathtt{OR}| = 21$, $d(\mathtt{OR} :!^n\mathtt{B} \multimap \cdots) = n$, deepest type $!^n\mathtt{B}$ of depth $d(!^n\mathtt{B}) = n + 1$;*

Table 1. Encoding.

Formula	Term	Type	Size	Depth
`true`	$\lambda x\ y.x$	$\forall n \geq 0.\ !^n\mathtt{B}$	3	n
`false`	$\lambda x\ y.y$	$\forall n \geq 0.\ !^n\mathtt{B}$	3	n
`NOT`	$\lambda b\ x\ y.(b\ y\ x)$	$\forall n \geq 0.\ !^n\mathtt{B} \multimap !^n\mathtt{B}$	12	n
`AND`	$\lambda b_1\ b_2\ x\ y.(b_1\ (b_2\ x\ y)\ y)$	$\forall n \geq 0.\ !^n\mathtt{B} \multimap !^n\mathtt{B} \multimap !^n\mathtt{B}$	21	n
`OR`	$\lambda b_1\ b_2\ x\ y.(b_1\ x\ (b_2\ x\ y))$	$\forall n \geq 0.\ !^n\mathtt{B} \multimap !^n\mathtt{B} \multimap !^n\mathtt{B}$	21	n
`IFF`	$\lambda b_1\ b_2\ x\ y.(b_1\ (b_2\ x\ y)(b_2\ y\ x))$	$\forall n \geq 1.\ !^n\mathtt{B} \multimap !^n\mathtt{B} \multimap !^n\mathtt{B}$	29	n
double	$\lambda x\ l\ c\ n.(l\ \lambda e.(c\ \lambda c'\ n'.\ (c'\ x(e\ c'\ n')))(l\ c\ n))$	$\forall \alpha,\gamma,\tau,\forall i \geq 0.\ !^{3+i}\alpha \multimap !L^{\gamma}_{!^i L^{\tau}_{\alpha}} \multimap !L^{\gamma}_{!^i L^{\tau}_{\alpha}}$	(*)	$3+i$
powerset	$\lambda A^*.(A^*\ double \lambda c\ n.\ (c\ \lambda c'\ n'.n'\ n))$	$\forall \alpha,\gamma,\tau,\forall i \geq 0.\ !^{L^{\gamma}_{!^i L^{\tau}_{\alpha}}}L^{\gamma}_{!^{3+i}\alpha} \multimap !^2 L^{\gamma}_{!^i L^{\tau}_{\alpha}}$	(*)	$4+i$
$\mathbf{D_1}$	$\lambda c\ n.(c\ \mathtt{true}\ (c\ \mathtt{false}\ n))$	$\forall \tau_1, \forall n_0, n_1 \geq 0.\ \Delta_1$ (**)	$\leq 2d$ (**)	$n_0 + n_1 + 1$
$\mathbf{D_k}$	$(powerset\ \mathbf{D_{k-1}})$	$\forall \tau_1,\ldots,\tau_k,\ \forall n_0,\ldots,n_{k-1} \geq 0,$ (**) $\forall n_k \geq 2(k-1).\ \Delta_k$	$\leq d(2k)!$	$k + \sum_{i=0}^{k} n_i$
eq_1	`IFF`	$\forall n \geq 1.\ !^n\mathtt{B} \multimap !^n\mathtt{B} \multimap !^n\mathtt{B}$	29	n
$member_k$	$\lambda x^{k-1}.\lambda y^k.(y^k\ \lambda y^{k-1}.(\mathtt{OR}\ (eq_{k-1}\ x^{k-1}\ y^{k-1}))\ \mathtt{false})$	$\forall n \geq 0.\ !^{n+\max\{2k-5,2\}}M_{k-1} \multimap !^{n+\max\{2k-7,0\}}M_k \multimap !^{n+\max\{4k-9,2(k-1)\}}\mathtt{B}$ (**)	$\leq c * k^2 + d(2(k-1))!$	$n + 2(k-1) + \max\{2k-7, 0\}$
eq_k	$\lambda x^k.\lambda y^k.(\lambda op.(\mathbf{D_{k-1}}\ \lambda z^{k-1}.(\mathtt{AND}\ (\mathtt{IFF}\ (op\ z^{k-1}\ x^k)\ (op\ z^{k-1}\ y^k)))\ \mathtt{true})\ member_k)$	E^k (**)	$\leq c * k^2 + d(2(k-1))!$	(*)
Higher-order logic formulas				
$\mathbf{true} \in x^2$	$(member_2\ \mathtt{true}\ x^2)$	$!^{f(2)}\mathtt{B}$ (**)	$\leq 4c + 2d$	$f(2)$
$\mathbf{false} \in x^2$	$(member_2\ \mathtt{false}\ x^2)$	$!^{f(2)}\mathtt{B}$	$\leq 4c + 2d$	$f(2)$
$x^{k-1} \in x^k$	$(member_k\ x^{k-1}\ x^k)$	$!^{f(k)}\mathtt{B}$	$\leq c * k^2 + d(2(k-1))!$	$f(k)$
$\neg\Phi'$	$(\mathtt{NOT}\ \Psi')$	$!^{f(k_j)+h+\sum_{\ell\in J} 2(k_\ell-1)}\mathtt{B}$	$O(\lvert\neg\Phi'\rvert d(2k_{\mathrm{MAX}})!)$	$f(k_j) + h + \sum_{\ell\in J} 2(k_\ell - 1)$
$\Phi_1 \wedge \Phi_2$	$(\mathtt{AND}\ \Psi_1\ \Psi_2)$	$!^{f(k_j)+h+\sum_{\ell\in J} 2(k_\ell-1)}\mathtt{B}$	$O(\lvert\Phi_1 \wedge \Phi_2\rvert d(2k_{\mathrm{MAX}})!)$	$f(k_j) + h + \sum_{\ell\in J} 2(k_\ell - 1)$
$\Phi_1 \vee \Phi_2$	$(\mathtt{OR}\ \Psi_1\ \Psi_2)$	$!^{f(k_j)+h+\sum_{\ell\in J} 2(k_\ell-1)}\mathtt{B}$	$O(\lvert\Phi_1 \vee \Phi_2\rvert d(2k_{\mathrm{MAX}})!)$	$f(k_j) + h + \sum_{\ell\in J} 2(k_\ell - 1)$
$\forall y^{k_i}\Phi'$	$(\mathbf{D_{k_i}}\ \lambda y^{k_i}.(\mathtt{AND}\ (\lambda x^{k_i}.\Psi'\ y^{k_i}))\ \mathtt{true})$	$!^{f(k_j)+h+\sum_{\ell\in J} 2(k_\ell-1)}\mathtt{B}$	$O(\lvert\forall y^{k_i}\Phi'\rvert d(2k_{\mathrm{MAX}})!)$	$f(k_j) + h + \sum_{\ell\in J} 2(k_\ell - 1)$
$\exists y^{k_i}\Phi'$	$(\mathbf{D_{k_i}}\ \lambda y^{k_i}.(\mathtt{OR}\ (\lambda x^{k_i}.\Psi'\ y^{k_i}))\ \mathtt{false})$	$!^{f(k_j)+h+\sum_{\ell\in J} 2(k_\ell-1)}\mathtt{B}$	$O(\lvert\exists y^{k_i}\Phi'\rvert d(2k_{\mathrm{MAX}})!)$	$f(k_j) + h + \sum_{\ell\in J} 2(k_\ell - 1)$

(6) $\texttt{IFF} = \lambda b_1\ b_2\ x\ y.(b_1\ (b_2\ x\ y)\ (b_2\ y\ x)) : !^n\texttt{B} \multimap !^n\texttt{B} \multimap !^n\texttt{B}$ *for any* $n \geq 1$, $|\texttt{IFF}| = 29$, $d(\texttt{IFF} : !^n\texttt{B} \multimap \cdots) = n$, *deepest type* $!^n\texttt{B}$ *of depth* $d(!^n\texttt{B}) = n + 1$*;*

Proof. All properties are trivially proven simply inspecting the various EAL-derivations. □

Following [Asperti and Mairson 1998], for any k the domain $\mathcal{D}_k$ can be encoded by a λ-term $\mathbf{D_k}$ representing $\mathcal{D}_k$ as the list of its values. Define

$$\begin{aligned} \mathbf{D_1} &= \lambda c.\lambda n.(c\ \texttt{true}\ (c\ \texttt{false}\ n)) \\ \mathbf{D_k} &= (\mathit{powerset}\ \mathbf{D_{k-1}}), \end{aligned}$$

where

$$\begin{aligned} \mathit{powerset} &= \lambda A^*.(A^*\ \mathit{double}\ \lambda c.\lambda n.(c\ \lambda c'.\lambda n'.n'\ n)) \\ \mathit{double} &= \lambda x.\lambda l.\lambda c.\lambda n.(l \\ &\qquad \lambda e.(c\ \lambda c'.\lambda n'.(c'\ x\ (e\ c'\ n'))))\ (l\ c\ n)). \end{aligned}$$

Lemma 2. *For any EAL-type* α, γ *and* τ *and for* $i \geq 0$*:*

—*double has type* $!^{3+i}\alpha \multimap !(L^{\gamma}_{!^i L^{\tau}_{\alpha}}) \multimap !(L^{\gamma}_{!^i L^{\tau}_{\alpha}})$; $d(\mathit{double} : !^{3+i}\alpha \multimap \cdots) = 3 + i$ *and the deepest type in the EAL-type derivation of double is* $!(L^{\gamma}_{!^i L^{\tau}_{\alpha}})$ *of depth* $2 + \max\{d(\gamma), 1 + i + d(\tau), 1 + i + d(\alpha)\}$.

—*powerset has type* $L^{!(L^{\gamma}_{!^i L^{\tau}_{\alpha}})}_{!^{3+i}\alpha} \multimap !!(L^{\gamma}_{!^i L^{\tau}_{\alpha}})$. $d(\mathit{powerset} : L^{!(L^{\gamma}_{!^i L^{\tau}_{\alpha}})}_{\ldots} \multimap \cdots) = 4 + i$ *and the deepest type in the EAL-type derivation of powerset is* $!!(L^{\gamma}_{!^i L^{\tau}_{\alpha}})$ *of depth* $3 + \max\{d(\gamma), 1 + i + d(\tau), 1 + i + d(\alpha)\}$.

Proof. See Figures 9 and 10, where the bold dashed lines stand for i boxes.

—the deapest type in derivation of $\mathit{double} : !^{3+i}\alpha \multimap !(L^{\gamma}_{!^i L^{\tau}_{\alpha}}) \multimap !(L^{\gamma}_{!^i L^{\tau}_{\alpha}})$ is the final type.

$$\begin{aligned} &d(!^{3+i}\alpha \multimap !(L^{\gamma}_{!^i L^{\tau}_{\alpha}}) \multimap !(L^{\gamma}_{!^i L^{\tau}_{\alpha}})) = \\ &\quad = \max\{3 + i + d(\alpha), 1 + \max\{1 + d(\gamma), 1 + i + \max\{1 + d(\tau), 1 + d(\alpha)\}\}\} \\ &\qquad = 2 + \max\{1 + i + d(\alpha), \max\{d(\gamma), 1 + i + \max\{d(\tau), d(\alpha)\}\}\} \\ &\qquad\quad = 2 + \max\{d(\gamma), 1 + i + d(\tau), 1 + i + d(\alpha)\} \\ &\qquad\qquad = d(!(L^{\gamma}_{!^i L^{\tau}_{\alpha}})); \end{aligned}$$

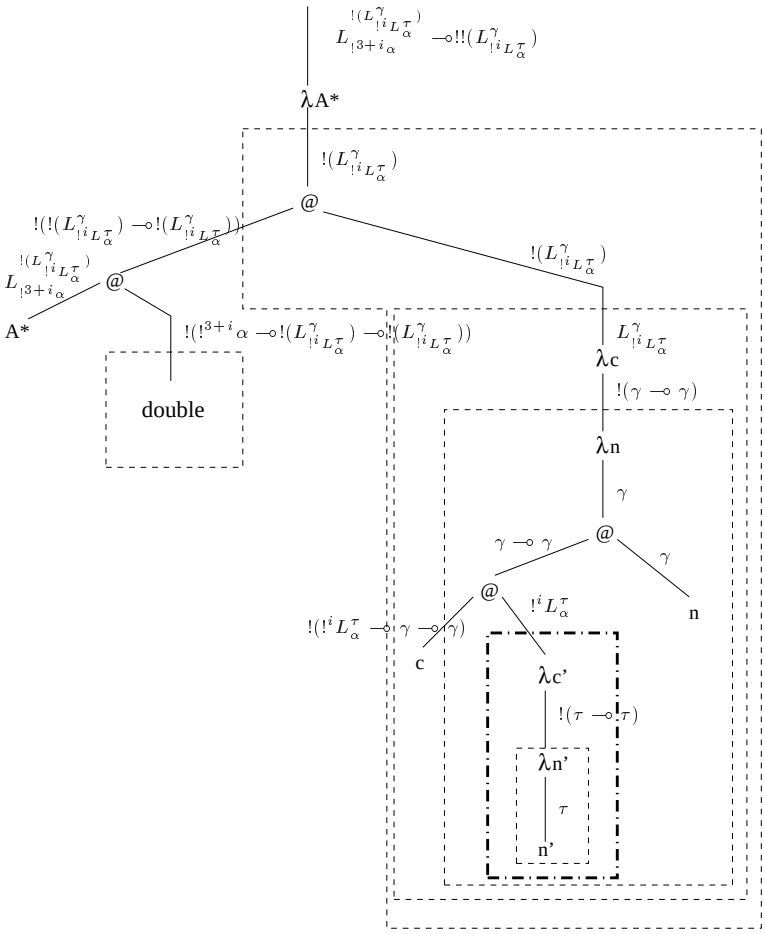

Fig. 10. EAL type of *powerset*

—analogously for *powerset* : $L^{!(L^{\gamma}_{!^i L^{\tau}_{\alpha}})}_{!^{3+i}\alpha} \multimap !!(L^{\gamma}_{!^i L^{\tau}_{\alpha}})$:

$$
\begin{aligned}
& d(L^{!(L^{\gamma}_{!^i L^{\tau}_{\alpha}})}_{!^{3+i}\alpha} \multimap !!(L^{\gamma}_{!^i L^{\tau}_{\alpha}})) = \\
& = \max\left\{ \max\left\{ \begin{array}{r} 1+3+i+d(\alpha), \\ 1+1+\max\{1+d(\gamma), 1+i+\max\{1+d(\tau), 1+d(\alpha)\}\} \\ 2+\max\{1+d(\gamma), 1+i+\max\{1+d(\tau), 1+d(\alpha)\}\} \end{array} \right\}, \right\} \\
& = \max\{1+3+i+d(\alpha), 2+\max\{1+d(\gamma), 1+i+\max\{1+d(\tau), 1+d(\alpha)\}\}\} \\
& = 3+\max\{1+i+d(\alpha), \max\{d(\gamma), \max\{1+i+d(\tau), 1+i+d(\alpha)\}\}\} \\
& = 3+\max\{d(\gamma), 1+i+d(\tau), 1+i+d(\alpha)\} \\
& = d(!!(L^{\gamma}_{!^i L^{\tau}_{\alpha}})).
\end{aligned}
$$

□

Observe how the piling up of !'s is already present at this stage of the encoding – in the type of *powerset* we have two consecutive !'s. Since in EAL is not possible to derive $!\alpha \multimap \alpha$, the number of consecutive !'s will keep increasing; intuitively, n applications of *powerset* will produce $2n$ consecutive !'s in the final type.

Given the type schema

DEFINITION 3.

$$
\begin{aligned}
\Delta_0 &= !^{n_0}\mathtt{B} \\
\Delta_k &= !^{n_k}(L^{\tau_k}_{\Delta_{k-1}},)
\end{aligned}
$$

we can prove the following:

LEMMA 3.

(1) $\forall \tau_1, \ldots, \tau_k$ *types in EAL,* $\forall n_0, \ldots, n_{k-1} \geq 0$, $\forall n_k \geq 2(k-1)$, $\mathbf{D_k}$ *has type* Δ_k.

(2) $d(\mathbf{D_k} : \Delta_k) = k + \sum_{i=0}^{k} n_i$.

(3) The deepest type in the derivation of $\mathbf{D_k}$ *is* Δ_k *of depth* $d(\Delta_k) = \max_{0 \leq j \leq k} \left\{ \sum_{i=j}^{k} (1+n_i) + d(\tau_j) \right\}$.

(4) $|\mathbf{D_k}| \leq d(2k)!$ *where* d *is a fixed constant.*

PROOF. For the proof of the size of $\mathbf{D_k}$ see [Asperti and Mairson 1998].

By induction on k:

$k = 1$ then

(1) $\forall \tau_1$ type in EAL, $\forall n_0 \geq 0$ and $\forall n_1 \geq 2(1-1)$, D_1 has type $\Delta_1 =!^{n_1}(L^{\tau_1}_{!^{n_0}\mathtt{B}})$. The proof is shown in Figure 11.

(2) Looking at Figure 11 is easy to see that depth of $\mathbf{D_1}$ is $1 + n_0 + n_1$.

(3) Again, looking at Figure 11, the deepest type is $!^{n_1}(L^{\tau}_{!^{n_0}\mathtt{B}}) = \Delta_1$ and

$$d(!^{n_1} L^{\tau_1}_{!^{n_0}\mathtt{B}}) = n_1 + 1 + \max\{d(\tau_1), n_0 + 1\}.$$

$k \geq 2$ (1) —if $k = 2$ by lemma 2, *powerset* has type $L^{!(L^{\gamma}_{!^i L^{\tau}_{\alpha}})}_{!^{3+i}\alpha} \multimap !!(L^{\gamma}_{!^i L^{\tau}_{\alpha}})$ for any type α, γ and τ and for $i \geq 0$, and, in particular, when $\alpha =!^{j}\mathtt{B}, j \geq 0$,

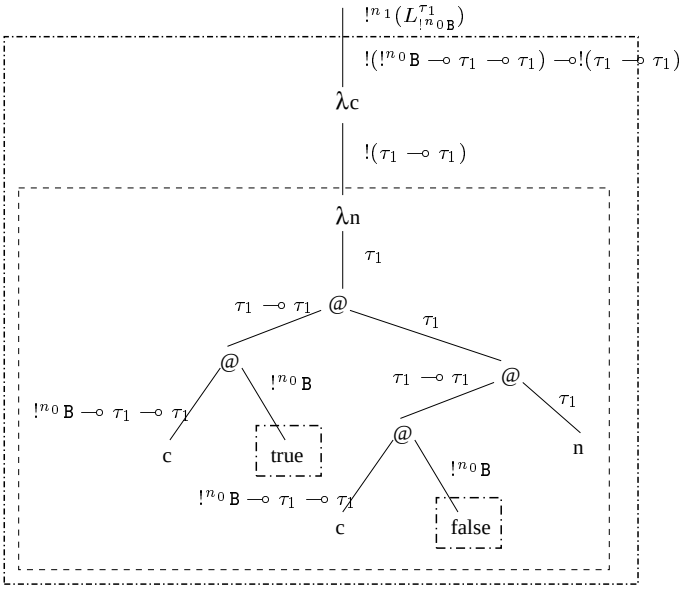

Fig. 11. EAL type of $\mathbf{D_1}$

powerset has type:

$$L^{!(L^{\gamma}_{!^i L^{\tau}_{!^j\mathtt{B}}})}_{!^{3+i}(!^j\mathtt{B})} \multimap !!(L^{\gamma}_{!^i L^{\tau}_{!^j\mathtt{B}}})$$

By the previous point $\mathbf{D_1}$ has type $!^{n_1}(L^{\tau_1}_{!^{n_0}\mathtt{B}})$ for any type τ_1 and $\forall n_0, n_1 \geq 0$. Hence, in particular, for $n_1 = 0, n_0 = 3 + i + j$ and $\tau_1 = !(L^{\gamma}_{!^i(L^{\tau}_{!^j\mathtt{B}})})$, $\mathbf{D_1}$ has type

$$L^{!(L^{\gamma}_{!^i L^{\tau}_{!^j\mathtt{B}}})}_{!^{3+i+j}\mathtt{B}}$$

Hence $\mathbf{D_2} = (\mathit{powerset}\ \mathbf{D_1})$, $\forall m \geq 0$ has type

$$!^{2+m}(L^{\gamma}_{!^i L^{\tau}_{!^j\mathtt{B}}})$$

see Figure 12 where the bold box represents m "normal" boxes.
Then $\forall \tau_1, \tau_2, \forall n_0, n_1 \geq 0, \forall n_2 \geq 2, \mathbf{D_2}$ has type

$$\Delta_2 = !^{n_2}(L^{\tau_2}_{!^{n_1}(L^{\tau_1}_{!^{n_0}\mathtt{B}})}).$$

—if $k > 2$, $\mathbf{D_k} = (\mathit{powerset}\ \mathbf{D_{k-1}})$ by definition.
By inductive hypothesis $\forall \tau_1, \ldots, \tau_{k-1}$ types in EAL, $\forall n_0, \ldots, n_{k-2} \geq 0$, $\forall n_{k-1} \geq 2(k-2)$, $\mathbf{D_{k-1}}$ has type $\Delta_{k-1} = !^{n_{k-1}}(L^{\tau_{k-1}}_{!^{n_{k-2}}(L^{\tau_{k-2}}_{\Delta_{k-3}})})$.
Similar to the previous case, for $\alpha = !^j(L^{\tau_{k-2}}_{\Delta_{k-3}})$, $n_{k-1} = 2(k-2), n_{k-2} =$

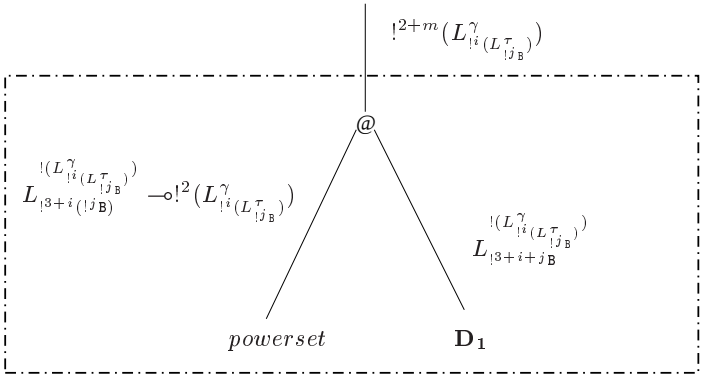

Fig. 12. EAL type of $\mathbf{D_2}$

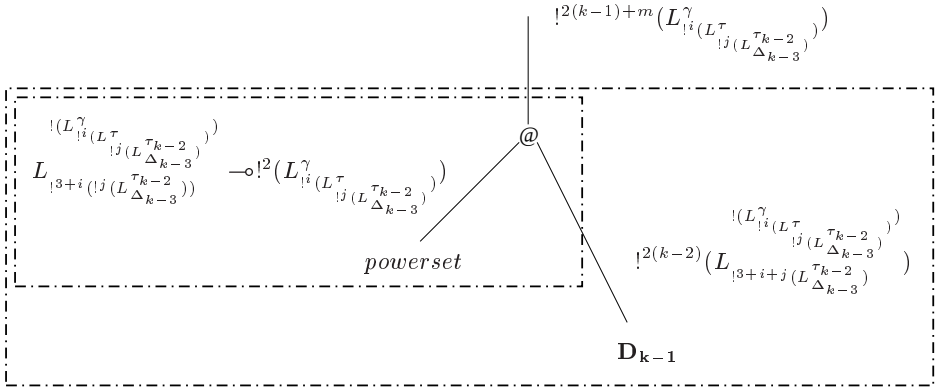

Fig. 13. EAL type of $\mathbf{D_k}$

$3+i+j$ and $\tau_{k-1} = !(L^{\gamma}_{!^i(L^{\tau}_{!^j(L^{\tau_{k-2}}_{\Delta_{k-3}})})})$ one has $\forall m \geq 0$, $\mathbf{D_k}$ of type

$$!^{2(k-1)+m}(L^{\gamma}_{!^i(L^{\tau}_{!^j(L^{\tau_{k-2}}_{\Delta_{k-3}})})})$$

(see figure 13, where the inner bold box represents $2(k-2)$ normal boxes and the outer one represents m normal boxes).

Hence $\forall \tau_1, \ldots, \tau_k, \forall n_0, \ldots, n_{k-1} \geq 0, \forall n_k \geq 2(k-1), \mathbf{D_k}$ has type

$$\Delta_k = !^{n_k}(L^{\tau_k}_{!^{n_{k-1}}(L^{\tau_{k-1}}_{!^{n_{k-2}}(L^{\tau_{k-2}}_{\Delta_{k-3}})})}).$$

(2) For $k \geq 2$ one has:

$$\begin{aligned}\mathbf{D_k} = (&powerset_{k-1}^{L^{\cdots}_{!^{3+n_{k-1}}(!^{n_{k-2}}(L^{\cdots}_{\Delta_{k-3}}))}}\\ &(powerset_{k-2}^{L^{\cdots}_{!^{3+(3+n_{k-1}+n_{k-2})}(!^{n_{k-3}}(L^{\cdots}_{\Delta_{k-4}}))}}\\ &\quad\cdots\\ &(powerset_1^{L^{\cdots}_{!^{3+(3+\cdots(3+n_{k-1}+n_{k-2})\cdots+n_1)}(!^{n_0}\mathtt{B})}}\\ &\quad\mathbf{D_1}^{L^{\cdots}_{!^{3+(3+\cdots(3+n_{k-1}+n_{k-2})\cdots+n_1)+n_0}\mathtt{B}}})\cdots))\end{aligned}$$

where the subscripts at every *powerset* are just notation for distinguish various instances.
Looking at Figure 13, every application of $powerset_j$ in the construction above is inside $2(j-1)$ boxes, hence depth of $\mathbf{D_k}$ is

$$\max\left\{\begin{array}{l} d(\mathbf{D_1} : L^{\cdots}_{!^{3(k-1)+\sum_{i=0}^{k-1} n_i}\mathtt{B}}),\\ \max_{1\leq j\leq k-1}\{d(powerset_j : L^{\cdots}_{!^{3(k-j)+\sum_{i=j-1}^{k-1} n_i}\ldots}) + 2(j-1)\}\end{array}\right\} + m$$

but

$$d(powerset_j : L^{\cdots}_{!^{3(k-j)+\sum_{i=j-1}^{k-1} n_i}\ldots}) + 2(j-1) > d(powerset_{j+1} : L^{\cdots}_{!^{3(k-j-1)+\sum_{i=j}^{k-1} n_i}\ldots}) + 2j$$

indeed

$$1 + 3(k-j) + \sum_{i=j-1}^{k-1} n_i + 2(j-1) > 1 + 3(k-j-1) + \sum i = j^{k-1} n_i + 2j$$

$$-1 + n_{j-1} > -2$$

and by hypothesis every n_i is greater than zero. Hence depth of $\mathbf{D_k}$ is

$$\begin{aligned}&\max\left\{\begin{array}{l} d(\mathbf{D_1} : L^{\cdots}_{!^{3(k-1)+\sum_{i=0}^{k-1} n_i}\mathtt{B}}),\\ d(powerset_1 : L^{\cdots}_{!^{3(k-1)+\sum_{i=0}^{k-1} n_i}\ldots})\end{array}\right\} + m\\ &= \max\left\{\begin{array}{l} 1 + 3(k-1) + \sum_{i=0}^{k-1} n_i,\\ 1 + 3(k-1) + \sum_{i=0}^{k-1} n_i\end{array}\right\} + m\\ &= 1 + m + 3(k-1) + \sum_{i=0}^{k-1} n_i\\ &= 1 + n_k + (k-1) + \sum_{i=0}^{k-1} n_i\\ &= k + \sum_{i=0}^{k} n_i.\end{aligned}$$

(3) for $\mathbf{D_k} :!^{2(k-1)+m} L^{\tau_k}_{\Delta_{k-1}}$, looking at Figure 13, we prove first that the deep-

est type is $!^{2(k-1)+m}L^{\tau_k}_{\Delta_{k-1}}$, indeed

$$d(!^{2(k-1)}L^{\tau_k}_{!^{n_{k-1}}L^{\tau_{k-1}}_{!^{n_{k-2}}L^{\tau_{k-2}}_{\Delta_{k-3}}}}) =$$

$$= 2(k-1)+1+\max\left\{\begin{array}{c} d(\tau_k), \\ n_{k-1}+1+\max\left\{\begin{array}{c} d(\tau_{k-1}), \\ n_{k-2}+1+\max\{d(\tau_{k-2}),d(\Delta_{k-3})\}\end{array}\right\}\end{array}\right\}$$

$$= 2(k-1)+1+\max\left\{\begin{array}{c} d(\tau_k), \\ n_{k-1}+1+\max\left\{\begin{array}{c} d(\tau_{k-1}), \\ n_{k-2}+1+d(\tau_{k-2}), \\ n_{k-2}+1+d(\Delta_{k-3})\end{array}\right\}\end{array}\right\}$$

$$= 2(k-1)+1+\max\left\{\begin{array}{c} d(\tau_k), \\ n_{k-1}+1+d(\tau_{k-1}), \\ n_{k-1}+n_{k-2}+2+d(\tau_{k-2}), \\ n_{k-1}+n_{k-2}+2+d(\Delta_{k-3})\end{array}\right\}$$

$$= 2(k-2)+3+\max\left\{\begin{array}{c} d(\tau_k), \\ n_{k-1}+1+d(\tau_{k-1}), \\ n_{k-1}+n_{k-2}+2+d(\tau_{k-2}), \\ n_{k-1}+n_{k-2}+2+d(\Delta_{k-3})\end{array}\right\}$$

$$= 2(k-2)+1+\max\left\{\begin{array}{c} 1+1+d(\tau_k), \\ 1+1+n_{k-1}+1+d(\tau_{k-1}), \\ 3+n_{k-1}+n_{k-2}+1+d(\tau_{k-2}), \\ 3+n_{k-1}+n_{k-2}+1+d(\Delta_{k-3})\end{array}\right\}$$

$$= 2(k-2)+1+\max\left\{\begin{array}{c} 1+1+\max\left\{\begin{array}{c} d(\tau_k), \\ n_{k-1}+1+d(\tau_{k-1}), \\ n_{k-1}+n_{k-2}+2+d(\tau_{k-2}) \\ n_{k-1}+n_{k-2}+2+d(\Delta_{k-3})\end{array}\right\} \\ 3+n_{k-1}+n_{k-2}+1+d(\tau_{k-2}), \\ 3+n_{k-1}+n_{k-2}+1+d(\Delta_{k-3})\end{array}\right\}$$

$$= 2(k-2)+1+\max\left\{\begin{array}{c} 1+1+\max\left\{\begin{array}{c} d(\tau_k), \\ n_{k-1}+1+d(\tau_{k-1}), \\ n_{k-1}+1+d(\Delta_{k-2})\end{array}\right\} \\ 3+n_{k-1}+n_{k-2}+1+d(\tau_{k-2}), \\ 3+n_{k-1}+n_{k-2}+1+d(\Delta_{k-3})\end{array}\right\}$$

$$= 2(k-2)+1+\max\left\{\begin{array}{c} 1+1+\max\left\{\begin{array}{c} d(\tau_k), \\ d(\Delta_{k-1})\end{array}\right\} \\ 3+n_{k-1}+n_{k-2}+1+\max\left\{\begin{array}{c} d(\tau_{k-2}), \\ d(\Delta_{k-3})\end{array}\right\}\end{array}\right\}$$

$$= d(!^{2(k-2)}L^{!L^{\tau_k}_{\Delta_{k-1}}}_{!^{3+n_{k-1}+n_{k-2}}L^{\tau_{k-2}}_{\Delta_{k-3}}})$$

Then we have depth of type of $\mathbf{D_{k-i}}$ equals to depth of type of $\mathbf{D_{k-i-1}}$ in the derivation of $\mathbf{D_k}$. Moreover depth of type of $\mathbf{D_{k-i}}$ is greater then depth of type of $powerset_{k-i}$, hence the deepest type is type of $\mathbf{D_k}$.

Then, assuming $\tau_0 = o$, we prove:

$$d(\Delta_k) = \max_{0\le j\le k}\left\{\sum_{i=j}^{k}(1+n_i)+d(\tau_j)\right\}$$

By induction on k:
—$d(\Delta_0) = d(!^{n_0}\mathtt{B}) = n_0 + 1$;
—

$$\begin{aligned} d(\Delta_k) &= 1 + n_k + \max\{d(\tau_k), d(\Delta_{k-1})\} \\ &\stackrel{\text{IH}}{=} \max\left\{1+n_k+d(\tau_k), 1+n_k+\max_{0\le j\le k-1}\left\{\sum_{i=j}^{k-1}(1+n_i)+d(\tau_j)\right\}\right\} \\ &= \max\left\{1+n_k+d(\tau_k), \max_{0\le j\le k-1}\left\{\sum_{i=j}^{k}(1+n_i)+d(\tau_j)\right\}\right\} \\ &= \max_{0\le j\le k}\left\{\sum_{i=j}^{k}(1+n_i)+d(\tau_j)\right\}. \end{aligned}$$

□

Prime formulas are encoded by the following terms.

$$\begin{aligned} eq_1 &= \mathtt{IFF} \qquad \text{with } n = 1 \\ member_k &= \lambda x^{k-1}.\lambda y^k.(y^k\ \lambda y^{k-1}.(\mathtt{OR} \\ &\qquad (eq_{k-1}\ x^{k-1}\ y^{k-1}))\ \mathtt{false}) \\ eq_k &= \lambda x^k.\lambda y^k.(\lambda op.(\mathbf{D_{k-1}}\ \lambda z^{k-1}.(\mathtt{AND} \\ &\qquad (\mathtt{IFF}\ (op\ z^{k-1}\ x^k)\ (op\ z^{k-1}\ y^k)))\ \mathtt{true}) \\ &\qquad member_k) \end{aligned}$$

Observe that $member_k$ is defined for $k \ge 2$.

LEMMA 4. *Let*

$$\begin{aligned} M_1 &= \mathtt{B} \\ M_k &= L^{!^{2k-3}\mathtt{B}}_{!M_{k-1}}. \end{aligned}$$

Then

(1) $member_k$ has EAL-type

$$!^{m+\max\{2k-5,2\}}M_{k-1} \multimap !^{m+\max\{2k-7,0\}}M_k \multimap !^{m+\max\{4k-9,2(k-1)\}}\mathtt{B}$$

for any $m \ge 0$.

(2) $\exists c \ge 0.\exists d \ge 0.\quad |member_k| \le c * k^2 + d(2(k-1))!$.

(3) $d(member_k :!^{m+\max\{2k-5,2\}}M_{k-1} \multimap \cdots) = m + 2(k-1) + \max\{2k-7, 0\}$.

(4) The deepest type in $member_k$ is $!^{m+\max\{4k-9,2(k-1)\}}\mathtt{B}$ of depth $\max\{4k-8, 2k-1\}$.

In order to prove Lemmma 4 we need to introduce the following (for $k \geq 2$):

$$\begin{aligned} eq'_1 &= \texttt{IFF} \\ member'_k &= \lambda x^{k-1}.\lambda y^k.(y^k\ \lambda y^{k-1}.(\texttt{OR}\ (eq'_{k-1}\ x^{k-1}\ y^{k-1}))\ \texttt{false}) \\ eq'_k &= \lambda x^k.\lambda y^k.(\lambda op.(d_{k-1}\ \lambda z^{k-1}.(\texttt{AND}\ (\texttt{IFF}\ (op\ z^{k-1}\ x^k) \\ &\qquad (op\ z^{k-1}\ y^k)))\ \texttt{true})\ member'_k) \end{aligned}$$

Note that $member'_k$, for $k \geq 3$ and eq'_k, for $k \geq 2$, are not closed terms, due to the presence of the unbound variable d_{k-1} in the definition of eq'_k. In particular, for $k \geq 3$, $FV(member'_k) = \{d_1, \ldots, d_{k-2}\}$ and, for $k \geq 2$, $FV(eq'_k) = \{d_1, \ldots, d_{k-1}\}$.

Moreover notice that:

$$\begin{aligned} member_k &= member'_k[\mathbf{D_1}/d_1, \ldots, \mathbf{D_{k-2}}/d_{k-2}] \\ eq_k &= eq'_k[\mathbf{D_1}/d_1, \ldots, \mathbf{D_{k-1}}/d_{k-1}] \end{aligned}$$

Definition 4. *We define types $E^k = E^k_1 \multimap E^k_1 \multimap E^k_3$ of eq'_k, for $k \geq 1$, and $M^k = M^k_1 \multimap M^k_2 \multimap M^k_3$ of $member'_k$, for $k \geq 2$ as follows:*

$$\begin{aligned} E^1_1 &= !\texttt{B} & & & E^1_3 &= !\texttt{B} \\ M^k_1 &= !E^{k-1}_1 & M^k_2 &= L^{E^{k-1}_3}_{E^{k-1}_1} & M^k_3 &= !E^{k-1}_3 \\ E^k_1 &= !M^k_2 & & & E^k_3 &= !M^k_3 \end{aligned}$$

Hence

$$\begin{aligned} E^1 &= !\texttt{B} \multimap !\texttt{B} \multimap !\texttt{B} \\ M^k &= !E^{k-1}_1 \multimap L^{E^{k-1}_3}_{E^{k-1}_1} \multimap !E^{k-1}_3 \\ E^k &= !M^k_2 \multimap !M^k_2 \multimap !M^k_3 \end{aligned}$$

Lemma 5. $\forall n \geq 0\ \forall k \geq 1\ \exists n_0, \ldots, n_k \geq 0\ \exists \tau_1, \ldots, \tau_k$ *s.t.* $\Delta_k = !^n(L^{E^k_3}_{E^k_1})$

Proof. By induction on k:

$k = 1$: $\Delta_1 \overset{n_0=1}{=} !^{n_1}(L^{\tau_1}_{!\texttt{B}}) \overset{\tau_1=E^1_3}{=} !^{n_1}(L^{E^1_3}_{!\texttt{B}}) = !^{n_1}(L^{E^1_3}_{E^1_1})$

$k > 1$: $\Delta_k = !^{n_k}(L^{\tau_k}_{\Delta_{k-1}}) \overset{\tau_k=E^k_3}{=} !^{n_k}(L^{E^k_3}_{\Delta_{k-1}}) \overset{\text{IH}}{=} !^{n_k}(L^{E^k_3}_{!(L^{E^{k-1}_3}_{E^{k-1}_1})}) =$

$= !^{n_k}(L^{E^k_3}_{!M^k_2}) = !^n(L^{E^k_3}_{E^k_1})$.

□

Lemma 6. $\forall k \geq 1\ \exists n_0, \ldots, n_{k-1} \geq 0\ \exists \tau_1, \ldots, \tau_k$ *such that* $\forall n_k \geq 0\ \Delta_k = !^{n_k}(L^{M^{k+1}_3}_{M^{k+1}_1})$.

Proof. By induction on k:

$k = 1$: $\Delta_1 = !^{n_1}(L^{\tau_1}_{!^{n_0}\texttt{B}}) \overset{n_0=2}{=} !^{n_1}(L^{\tau_1}_{!^2\texttt{B}}) \overset{\tau_1=M^2_3}{=} !^{n_1}(L^{M^2_3}_{!^2\texttt{B}}) = !^{n_1}(L^{M^2_3}_{M^2_1})$

$k > 1$: $\Delta_k = !^{n_k}(L^{\tau_k}_{\Delta_{k-1}}) \overset{lemma\ 5}{=} !^{n_k}(L^{\tau_k}_{!^2(L^{E^{k-1}_3}_{E^{k-1}_1})}) = !^{n_k}(L^{M^{k+1}_3}_{M^{k+1}_1})$.

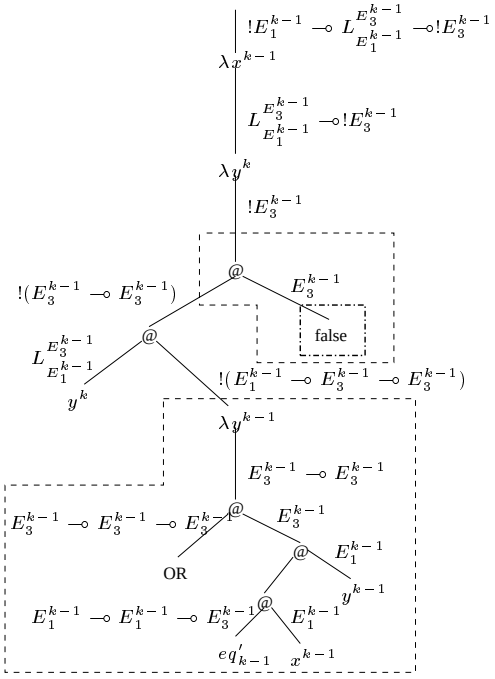

Fig. 14. EAL type of $member'_k$

□

LEMMA 7. *(1) $member'_k$ has type M^k in EAL, with $FV(member'_k) = \{d_1, \ldots, d_{k-2}\}$, d_i of type $!^{2(k-2-i)+1}(L_{M_1^{i+1}}^{M_3^{i+1}})$ for $k \geq 2$, moreover $d(member'_k : M^k) = 2(k-1)$;*

(2) eq'_k has type E^k in EAL, with $FV(eq'_k) = \{d_1, \ldots, d_{k-1}\}$, d_i of type $!^{2(k-1-i)}(L_{M_1^{i+1}}^{M_3^{i+1}})$ for $k \geq 1$, moreover $d(eq'_k : E^k) = 2(k-1)+1$.

PROOF. By mutual induction on k.
First of all, note that $\forall k \geq 2 \quad M_3^k \equiv !^{2(k-1)}\mathtt{B}$ and $\forall k \geq 1 \quad E_3^k \equiv !^{2(k-1)+1}\mathtt{B}$.
The proof for $member'_k$ is shown in figure 14, where OR has type

$$!^{2(k-2)+1}\mathtt{B} \multimap !^{2(k-2)+1}\mathtt{B} \multimap !^{2(k-2)+1}\mathtt{B}$$

for the observation above, and the bold box represents $2(k-2)+1$ normal boxes. Note that eq'_{k-1} has free variables $d_1 :!^{2(k-3)}(L_{M_1^2}^{M_3^2}), \ldots, d_{k-2} : L_{M_1^{k-1}}^{M_3^{k-1}}$ and it is inside a box, hence $member'_k$ has the same free variables with the same types, but with one more !, that is $d_1 :!^{2(k-3)+1}(L_{M_1^2}^{M_3^2}), \ldots, d_{k-2} :!(L_{M_1^{k-1}}^{M_3^{k-1}})$.

The proof for eq'_k is shown if figure 15. Note that $member'_k$ has free variables $d_1 :!^{2(k-3)+1}(L_{M_1^2}^{M_3^2}), \ldots, d_{k-2} :!(L_{M_1^{k-1}}^{M_3^{k-1}})$ and it is inside a box, hence eq'_k has free variables, $d_{k-1} : L_{M_1^k}^{M_3^k}$ plus the variables of $member'_k$ with the same type with one

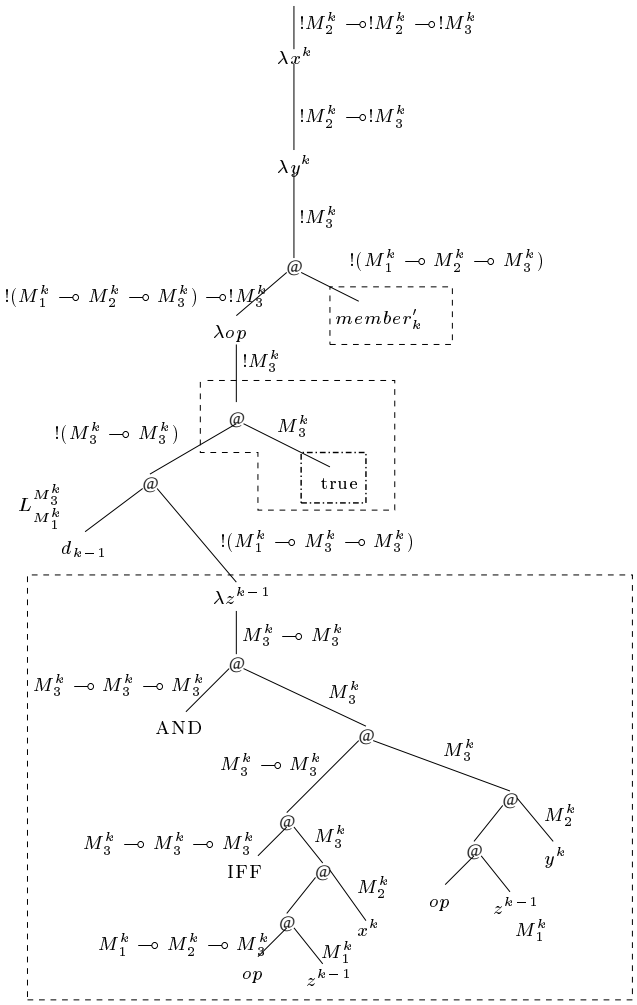

Fig. 15. EAL type of eq'_k

more !, that is $d_1 :!^{2(k-2)}(L^{M_3^2}_{M_1^2}), \ldots, d_{k-1} : L^{M_3^k}_{M_1^k}$.

Looking at figures 15 and 14,

$$d(eq'_1 : E^1) = 1$$

$$\begin{aligned}
d(member'_k : M^k) &= \max\{d(\texttt{false} : E_3^{k-1}), d(\texttt{OR} : E_3^{k-1} \multimap \cdots), d(eq'_{k-1} : E^{k-1})\} + 1 \\
&= \max\{d(\texttt{false} :!^{2(k-2)+1}\texttt{B}), d(\texttt{OR} :!^{2(k-2)+1}\texttt{B} \multimap \cdots), d(eq'_{k-1} : E^{k-1})\} + 1 \\
&= \max\{2(k-2)+1, d(eq'_{k-1} : E^{k-1})\} + 1 \\
&\overset{\text{IH}}{=} \max\{2(k-2)+1, 2(k-2)+1\} + 1 \\
&= 2(k-1)
\end{aligned}$$

$$
\begin{aligned}
d(eq'_k : E^k) &= \max \left\{ \begin{array}{l} d(\mathtt{true} : M_3^k), d(\mathtt{AND} : M_3^k \multimap \cdots), \\ d(\mathtt{IFF} : M_3^k \multimap \cdots), d(member'_k : M^k) \end{array} \right\} + 1 \\
&= \max \left\{ \begin{array}{l} d(\mathtt{true} :!^{2(k-1)}\mathtt{B}), d(\mathtt{AND} :!^{2(k-1)}\mathtt{B} \multimap \cdots), \\ d(\mathtt{IFF} :!^{2(k-1)}\mathtt{B} \multimap \cdots), d(member'_k : M^k) \end{array} \right\} + 1 \\
&= \max\{2(k-1), d(member'_k : M^k)\} + 1 \\
&\stackrel{\text{IH}}{=} \max\{2(k-1), 2(k-1)\} + 1 \\
&= 2(k-1) + 1.
\end{aligned}
$$

□

LEMMA 8. $\forall k \geq 2 \quad \forall n \geq max\{2k-7, 0\}$ *member*$_k$ *has type* $!^n M_1^k \multimap !^n M_2^k \multimap !^n M_3^k$ *in EAL.*

PROOF. By the previous lemma $member'_k$ has type M^k in EAL, with free variables $d_1 :!^{2(k-3)+1}(L_{M_1^2}^{M_3^2}), \ldots, d_{k-2} :!(L_{M_1^{k-1}}^{M_3^{k-1}})$, and by lemma 3, $\forall \tau_1, \ldots, \tau_j$ types in EAL, $\forall n_0, \ldots, n_{j-1} \geq 0$, $\forall n_j \geq 2(j-1)$, $\mathbf{D_j}$ has type Δ_j, in particular, by Lemma 6, $\mathbf{D_j}$ has type

$$!^{max\{2(j-1), 2(k-2-j)+1\}}(L_{M_1^{j+1}}^{M_3^{j+1}}).$$

The proof is shown in figure 16, where

(1) $i = \lceil \frac{k}{2} \rceil$ is the lowest i s.t. $2(i-1) > 2(k-2-i)+1$,

(2)

$$
\begin{aligned}
\mathbf{D_1} &: !^{2(k-3)+1}(L_{M_1^2}^{M_3^2}) \\
&\vdots \\
\mathbf{D_{i-1}} &: !^{2(k-2-i+1)+1}(L_{M_1^i}^{M_3^i}) \\
\mathbf{D_i} &: !^{2(i-1)}(L_{M_1^{i+1}}^{M_3^{i+1}}) \\
&\vdots \\
\mathbf{D_{k-2}} &: !^{2(k-3)}(L_{M_1^{k-1}}^{M_3^{k-1}})
\end{aligned}
$$

(3) the bold box inserted at step three represent $m = \left\{ \begin{array}{ll} 1 & k \text{ even} \\ 3 & k \text{ odd} \end{array} \right.$ boxes,

(4) $p = max\{2k-7, 0\}$.

The other bold boxes represent four normal boxes. □

PROOF OF LEMMA 4. (1) We need to prove that $\forall k \geq 2$, $\forall n \geq \max\{2k-7, 0\}$, $\exists m \geq 0$, such that

$$!^n M_1^k \multimap !^n M_2^k \multimap !^n M_3^k = !^{m+\max\{2k-5,2\}} M_{k-1} \multimap !^{m+\max\{2k-7,0\}} M_k \multimap !^{m+\max\{4k-9,2(k-1)\}}\mathtt{B}.$$

and viceversa.

It is equivalent to prove:

(a) $!^n M_3^k = !^{m+\max\{4k-9,2(k-1)\}}\mathtt{B}$,

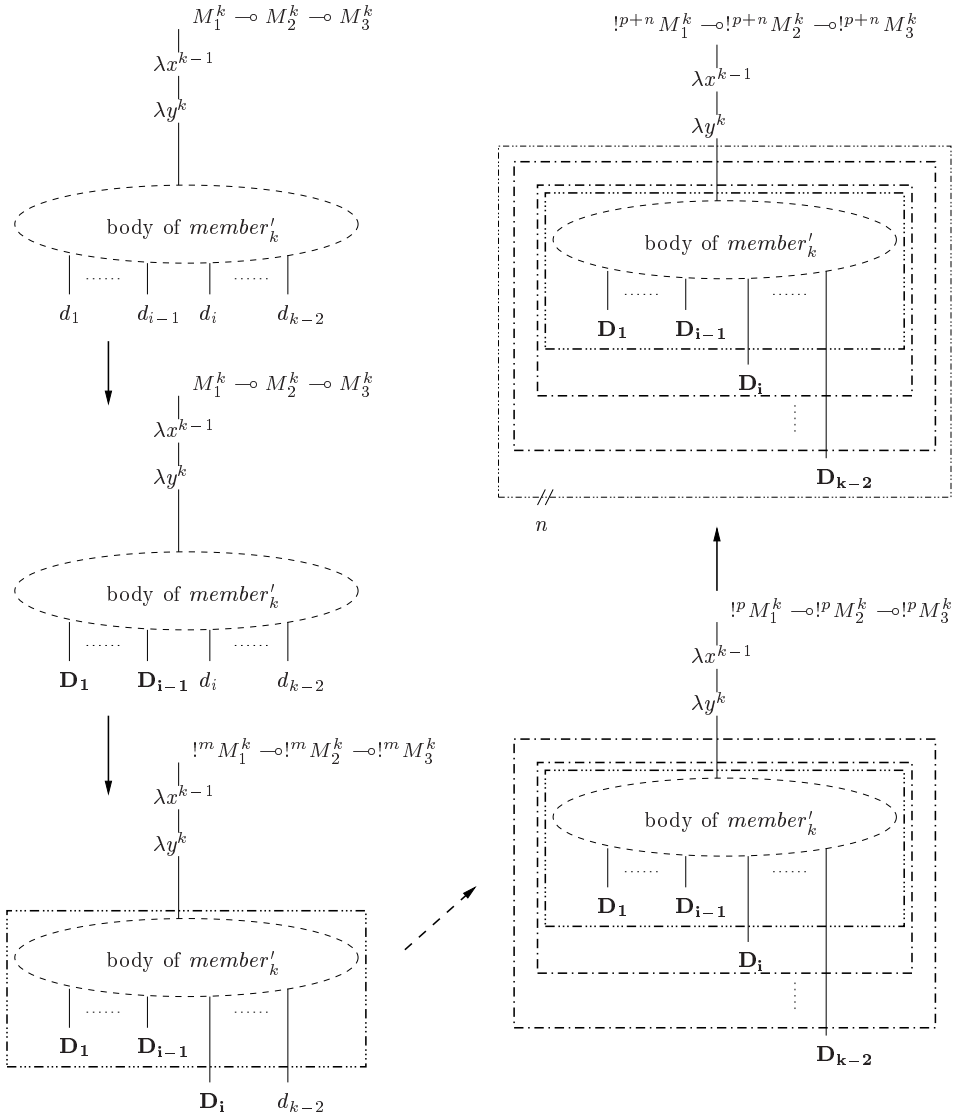

Fig. 16. EAL type of $member_k$

(b) $!^n M_2^k = !^{m+\max\{2k-7,0\}} M_k$,

(c) $!^n M_1^k = !^{m+\max\{2k-5,2\}} M_{k-1}$.

(a) $!^n M_3^k = !^{m+\max\{2k-7,0\}+2(k-1)}\mathtt{B} = !^{m+\max\{4k-9,2(k-1)\}}\mathtt{B}$;

(b) by induction on $k \geq 2$ we prove $M_2^k = M_k$:

$(k=2)$ $M_2^2 = L_{!\mathtt{B}}^{!\mathtt{B}} = L_{!M_1}^{!^{4-3}\mathtt{B}} = M_2$;

$(k>2)$ $M_2^k = L_{!M_2^{k-1}}^{!^{2(k-2)+1}\mathtt{B}} \overset{\text{IH}}{=} L_{!M_{k-1}}^{!^{2k-3}\mathtt{B}} = M_k$.

Hence trivially $!^n M_2^k = !^{m+\max\{2k-7,0\}} M_k$;

(c) $!^{m+\max\{2k-7,0\}} M_1^k = !^{m+\max\{2k-7,0\}} !! M_2^{k-1} = !^{m+\max\{2k-7,0\}+2} M_{k-1} = !^{m+\max\{2k-5,2\}} M_{k-1}$.

(2) First notice the following:

$$\begin{aligned}
(E_1^1)^* &= (\Delta_0)^* = (\mathtt{B})^* \\
(E_1^k)^* &= (L^{\mathtt{B}}_{E_1^{k-1}})^* = (L^{\mathtt{B}}_{\Delta_{k-2}})^* = (\Delta^{\mathtt{B}}_{k-1})^* \\
(M_1^2)^* &= (\Delta_0)^* = (\mathtt{B})^* \\
(M_1^k)^* &= (L^{\mathtt{B}}_{E_1^{k-2}})^* = (L^{\mathtt{B}}_{\Delta_{k-3}})^* = (\Delta^{\mathtt{B}}_{k-2})^*
\end{aligned}$$

and

$$\begin{aligned}
||(\Delta_k^\tau)^*|| &= k(4+4||(\tau)^*||) + 5 \\
||(\mathtt{B})^*|| &= ||(\Delta_0^o)^*|| = ||(\Delta_0)^*|| = 5 \\
||(\Delta_k^{\mathtt{B}})^*|| &= 24k + 5 \leq 24(k+1)
\end{aligned}$$

hence

$$\begin{aligned}
|member_k| &= 8 + |\mathtt{OR}| + |eq_{k-1}| + |\mathtt{false}| \\
&\quad + |y^k| + |x^{k-1}| + |y^{k-1}| \\
&= 32 + |eq_{k-1}| + ||(\Delta^{\mathtt{B}}_{k-1})^*|| + 2||(\Delta^{\mathtt{B}}_{k-2})^*|| \\
&= 32 + 24(k-1) + 5 + 48(k-2) + 10 + |eq_{k-1}| \\
&= 72k - 73 + |eq_{k-1}| \\
|eq_k| &= 14 + |\mathtt{AND}| + |\mathtt{IFF}| + |\mathtt{true}| + |\mathbf{D_{k-1}}| \\
&\quad + |member_k| + |op| + 2|z^{k-1}| + |x^k| + |y^k| \\
&= 14 + 21 + 29 + 3 + |\mathbf{D_{k-1}}| + |member_k| \\
&\quad + ||(\Delta^{\mathtt{B}}_{k-2} \multimap \Delta^{\mathtt{B}}_{k-1} \multimap \mathtt{B})^*|| + 2||(\Delta^{\mathtt{B}}_{k-2})^*|| + 2||(\Delta^{\mathtt{B}}_{k-1})^*|| \\
&= 69 + |\mathbf{D_{k-1}}| + |member_k| + ||(\mathtt{B})^*|| + 3||(\Delta^{\mathtt{B}}_{k-2})^*|| + 3||(\Delta^{\mathtt{B}}_{k-1})^*|| \\
&= 144k - 112 + |\mathbf{D_{k-1}}| + |member_k| \\
&\leq 144k - 112 + d(2(k-1))! + |member_k|
\end{aligned}$$

then

$$\begin{array}{rcl}
|member_2| & = & 72 * 2 - 73 + 29 = 100 \\
|member_k| & \leq & c_k * k + d(2(k-2))! + |member_{k-1}| \\
& \leq & c_k * k + c_{k-1} * (k-1) + d(2(k-2))! + d(2(k-3))! \\
& & \quad + |member_{k-2}| \\
& \cdots & \\
& \leq & \sum_{i=3}^{k} c_i i + \sum_{i=1}^{k-2} d(2i)! \\
& \leq & c * k^2 + d(2(k-1))!
\end{array}$$

(3) Looking at Figure 16 and by lemma 8 we have depth

$$= \max \left\{ \begin{array}{l} dpt(member'_k : M^k) + \max\{2k-7, 0\}, \\ \max_{1 \leq j < \lceil k/2 \rceil} \{dpt(\mathbf{D_j} :!^{2(k-2-j)+1} L^{\cdots}_{M_1^{j+1}})\} + \max\{2k-7, 0\} \\ \max_{\lceil k/2 \rceil \leq j \leq k-2} \{dpt(\mathbf{D_j} :!^{2(j-1)} L^{\cdots}_{M_1^{j+1}}) + 4(k-2-j)\} \end{array} \right\} + m$$

First we need to show that for all $k \geq 1 \quad E_1^k = \Delta_{k-1}$ with $n_{k-1} = \cdots = n_0 = 1$ (we use notation $\Delta_{k-1}^{(1,\ldots)}$) and for all $k \geq 2 \quad M_1^k = \Delta_{k-2}$ with $n_{k-2} = 2$ and $n_{k-3} = \cdots = n_0 = 1$ (we use $\Delta_{k-2}^{(2,1,\ldots)}$). By induction on k:

$$\begin{array}{l}
E_1^1 =!\mathtt{B} = \Delta_0^{(1,\ldots)} \\
E_1^k =!L_{E_1^{k-1}} \stackrel{\text{IH}}{=}!L_{\Delta_{k-2}^{(1,\ldots)}} = \Delta_{k-1}^{(1,\ldots)}
\end{array}$$

then, using result above:

$$\begin{array}{l}
M_1^2 =!E_1^1 =!^2\mathtt{B} = \Delta_0^{(2)} \\
M_1^k =!E_1^{k-1} =!^2 M_2^{k-1} =!^2 L_{E_1^{k-2}} =!^2 L_{\Delta_{k-3}^{(1,\ldots)}} = \Delta_{k-2}^{(2,1,\ldots)}
\end{array}$$

then, in the calculus of $d(\mathbf{D_j})$ for $d(member_k)$, we have $\sum_{\ell=0}^{j-1} n_\ell = j+1$.

Hence

$$
\begin{aligned}
&d(member_k :!^{\max\{2k-5,2\}+n} M_{k-1} \multimap \cdots) = \\
&= \max\left\{\begin{array}{l} d(member'_k : M^k) + \max\{2k-7,0\}, \\ \max_{1\le j<\lceil k/2\rceil}\{d(\mathbf{D_j} :!^{2(k-2-j)+1} L^{\cdots}_{\Delta^{(2,1,\ldots)}_{j-1}})\} + \max\{2k-7,0\} \\ \max_{\lceil k/2\rceil\le j\le k-2}\{d(\mathbf{D_j} :!^{2(j-1)} L^{\cdots}_{\Delta^{(2,1,\ldots)}_{j-1}}) + 4(k-2-j)\} \end{array}\right\} + m \\
&= \max\left\{\begin{array}{l} 2(k-1) + \max\{2k-7,0\}, \\ \max_{1\le j<\lceil k/2\rceil}\{d(\mathbf{D_j} :!^{2(j-1)+2(k-2j-1)+1} L^{\cdots}_{\Delta^{(2,1,\ldots)}_{j-1}})\} + \max\{2k-7,0\} \\ \max_{\lceil k/2\rceil\le j\le k-2}\{d(\mathbf{D_j} :!^{2(j-1)} L^{\cdots}_{\Delta^{(2,1,\ldots)}_{j-1}}) + 4(k-2-j)\} \end{array}\right\} + m \\
&= \max\left\{\begin{array}{l} 2(k-1) + \max\{2k-7,0\}, \\ \max_{1\le j<\lceil k/2\rceil}\{1+2(k-2j-1)+1+3(j-1)+\sum_{\ell=0}^{j-1} n_\ell\} + \max\{2k-7,0\} \\ \max_{\lceil k/2\rceil\le j\le k-2}\{1+3(j-1)+\sum_{\ell=0}^{j-1} n_\ell + 4(k-2-j)\} \end{array}\right\} + m \\
&= \max\left\{\begin{array}{l} 2(k-1) + \max\{2k-7,0\}, \\ \max_{1\le j<\lceil k/2\rceil}\{1+2(k-2j-1)+1+3(j-1)+(j+1)\} + \max\{2k-7,0\} \\ \max_{\lceil k/2\rceil\le j\le k-2}\{1+3(j-1)+(j+1)+4(k-2-j)\} \end{array}\right\} + m \\
&= \max\left\{\begin{array}{l} 2(k-1) + \max\{2k-7,0\}, \\ \max_{1\le j<\lceil k/2\rceil}\{2(k-1)\} + \max\{2k-7,0\} \\ \max_{\lceil k/2\rceil\le j\le k-2}\{4k-9\} \end{array}\right\} + m \\
&= \max\left\{\begin{array}{l} 2(k-1) + \max\{2k-7,0\}, \\ 2(k-1) + \max\{2k-7,0\} \\ 4k-9 \end{array}\right\} + m \\
&= m + 2(k-1) + \max\{2k-7,0\}
\end{aligned}
$$

(4) In the derivation of $member_k$ the deepest type is the deepest between the type of $member'_k$ and types of various $\mathbf{D_i}$s.

First notice that $M_k = M_2^k = \Delta_{k-1}$ with $n_{k-1} = 0$ and $n_{k-2} = \cdots = n_0 = 1$ and $\tau_{k_i} =!^{2k_i-3}\mathtt{B}$, then

$$
\begin{aligned}
d(M_k) = d(M_2^k) = d(\Delta_{k-1}^{\tau_{k_i} =!^{2k_i-3}\mathtt{B}}{}^{(0,1,\ldots)}) &= \\
&= \max_{0\le j\le k-1}\left\{\sum_{i=j}^{k-1}(1+n_i) + d(\tau_j)\right\} \\
&= \max\left\{\max_{1\le j\le k-1}\left\{\sum_{i=j}^{k-2} 2 + 2(j-1)\right\}, \sum_{i=0}^{k-1}(1+n_i)\right\} \\
&= \max\left\{\max_{1\le j\le k-1}\{1+2(k-1)-2j+2(j-1)\}, 1+2(k-1)\right\} \\
&= 2k-1. \quad (1)
\end{aligned}
$$

Hence depth of type of $member_k$ is:

$$\begin{aligned} d(!^{\max\{2k-5,2\}}M_{k-1} \multimap !^{\max\{2k-7,0\}}M_k \multimap !^{\max\{4k-9,2(k-1)\}}\mathtt{B}) = \\ = \max\left\{\begin{array}{c}\max\{2k-5,2\}+d(M_{k-1})\\ \max\{2k-7,0\}+d(M_k)\\ \max\{4k-9,2(k-1)\}+1\end{array}\right\} \\ = \max\left\{\begin{array}{c}\max\{2k-5,2\}+2(k-1)-1)\\ \max\{2k-7,0\}+2k-1\\ \max\{4k-9,2(k-1)\}+1\end{array}\right\} \\ = \max\{4k-8,2k-1,4k-8,2k-1,4k-8,2k-1\} \\ = \max\{4k-8,2k-1\} = d(!^{\max\{4k-9,2(k-1)\}}\mathtt{B}) \quad (2) \end{aligned}$$

By lemma 8, the $\mathbf{D_i}$s have type $!^{\max\{2(i-1),2(k-2-i)+1\}}(L_{M_1^{i+1}}^{M_3^{i+1}})$ for $1 \le i \le k-2$.

$$\begin{aligned} d(!^{\max\{2(i-1),2(k-2-i)+1\}}(L_{M_1^{i+1}}^{M_3^{i+1}})) = \\ = d(!^{\max\{2(i-1),2(k-2-i)+1\}}(L_{!^2M_2^i}^{!^{2i}\mathtt{B}})) \\ = \max\{2(i-1),2(k-2-i)+1\}+1+\max\{2i+1,2+2i-1\} \\ = \max\{2(i-1),2(k-2-i)+1\}+2i+2 \\ = \max\{4i,2k-1\} = 2k-1. \quad (3) \end{aligned}$$

Hence the deepest type is $!^{\max\{4k-9,2(k-1)\}}\mathtt{B}$, of depth $\max\{4k-8,2k-1\}$. □

Putting together all the ingredients of the encoding, we obtain our main technical result, which establishes Theorem 2 for EAL-typed terms.

THEOREM 5. *Define* $f(k) = \left\{\begin{array}{l} 2k-1 \;\; 1 \le k \le 3 \\ 4k-9 \;\; k \ge 4 \end{array}\right.$.

Let $\Psi[x_1^{k_1},\ldots,x_n^{k_n}]$ *be the term encoding an arbitrary formula* Φ *with free variables* $x_{i+1}^{k_{i+1}},\ldots,x_n^{k_n}$, $0 \le i \le n$, *and* m *quantifiers over* $\mathcal{D}_1,\ldots,\mathcal{D}_i$. *Then*

$$\exists J \subseteq \{1,\ldots,i\},\ \exists j \in \{1,\ldots,n\},\ \exists d \le m,$$

such that in EAL

$$\Psi[x_1^{k_1},\ldots,x_n^{k_n}] :!^{f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1)}\mathtt{B}$$

and free variables $(i+1 \le h \le n)$

$$x_h^{k_h} :!^{\max\{2k_h-7,1\}+f(k_j)-f(k_h)+d+\sum_{\ell\in J}2(k_\ell-1)}M_{k_h}.$$

Moreover

(1) $d(\Psi[x_1^{k_1},\ldots,x_n^{k_n}] :!^{f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1)}\mathtt{B}) = f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1)$;

(2) the deepest type in the EAL-derivation of $\Psi[x_1^{k_1},\ldots,x_n^{k_n}]$ *is the final one, of depth* $\le f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1)+1$;

(3) size of $\Psi[x_1^{k_1},\ldots,x_n^{k_n}]$ *is* $O(|\Phi|(2k_{\mathrm{MAX}})!)$ *where* k_{MAX} *is the greatest* k *such that* $member_k$ *appears in* $\Psi[x_1^{k_1},\ldots,x_n^{k_n}]$.

PROOF. **base** : all base cases have no quantifiers, hence $m = d = 0$ and $J = \emptyset$. Hence the thesis becomes $\Psi :!^{f(k_j)}\mathtt{B}$ with free variables $x_i^{k_i} :!^{\max\{2k_i-7,1\}+f(k_j)-f(k_i)}M_{k_i}$ and

(1) $d(\Psi) = f(k_j)$;
(2) the deepest type is the final one of depth $\leq f(k_j)+1$;
(3) size of Ψ is $O(|\Phi|(2k)!)$.

By cases:

—$\Psi[x^2] = (member_2\ \mathtt{true}\ x^2)$
by lemma 4 $\Psi[x^2] :!^{f(2)}\mathtt{B}$ with $x^2 :!M_2$.

(1)

$$\begin{aligned} d(\Psi[x^2] :!^3\mathtt{B}) &= \max\{d(member_2 :!^{2+1}M_1 \multimap \cdots), d(\mathtt{true} :!^3\mathtt{B})\} \\ &= \max\{1+2(2-1), 3\} = f(2). \end{aligned}$$

Then the thesis holds with $k_j = 2$.

—$\Psi[x^2] = (member_2\ \mathtt{false}\ x^2)$
analogous to the previous one.

—$\Psi[x^{k-1}, x^k] = (member_k\ x^{k-1}\ x^k)$
by lemma 4 $\Psi[x^{k-1}, x^k] :!^{m_3}\mathtt{B}$ for every $m_3 \geq \max\{4k-9, 2(k-1)\}$. $f(k) \geq \max\{4k-9, 2(k-1)\}$ hence $\Psi[x^{k-1}, x^k] :!^{f(k)}\mathtt{B}$. Again by lemma 4 $x^k :!^{m_2}M_k$ for any $m_2 \geq \max\{2k-7, 0\}$, and in particular $x^k :!^{\max\{2k-7,1\}+f(k)-f(k)}M_k$, and $x^{k-1} :!^{m_1}M_{k-1}$ for any $m_1 \geq \max\{2k-5, 2\}$, and in particular $\max\{2(k-1)-7, 1\}+f(k)-f(k-1) \geq \max\{2k-5, 2\}$.

(1)

$$\begin{aligned} d(\Psi[x^{k-1}, x^k] :!^{f(k)}\mathtt{B}) &= d(member_k :!^{\max\{2k-5,3\}}M_{k-1}) \\ &= \begin{cases} 2(k-1)+1 & k \leq 3 \\ 4k-9 & k \geq 4 \end{cases} \\ &= f(k) \end{aligned}$$

Than the thesis holds with $k_j = k$.

Moreover,

(1) **da rendere nascosto**
(2) considering the EAL-derivations, the deepest type is the type of $member_k$ of depth $\max\{4k-8, 2k-1\} \leq f(k)+1$ by Lemma 8.
(3) Finally, by Lemma 8, size of Ψ is $O(|\Phi|(2k)!)$.

inductive step :

—$\Psi[x_1^{k_1}, \ldots, x_n^{k_n}] = (\mathtt{NOT}\ \Psi'[x_1^{k_1}, \ldots, x_n^{k_n}])$
By inductive hypothesis $\Psi[x_1^{k_1}, \ldots, x_n^{k_n}] :!^{f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1)}\mathtt{B}$.

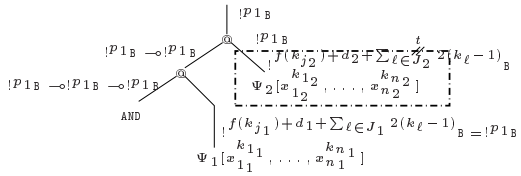

Fig. 17. EAL type of an AND formula

(1)

$$\begin{aligned}
&d(\Psi[x_1^{k_1}\dots x_n^{k_n}] :!^{f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1)}\mathtt{B}) = \\
&\quad = \max\{d(\mathtt{NOT} :!^{f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1)}\mathtt{B}), d(\Psi' :!^{f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1)}\mathtt{B})\} \\
&\quad = \max\{f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1), d(\Psi' :!^{f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1)}\mathtt{B})\} \\
&\quad \overset{\text{IH}}{=} \max\{f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1), f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1)\} \\
&\quad = f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1)
\end{aligned}$$

(2) It is easy to see that the deepest type is the final one, of depth

$$d(!^{f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1)}\mathtt{B}) = f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1)+1.$$

(3) By inductive hypothesis $|\Psi'[x_1^{k_1},\dots,x_n^{k_n}]|$ is $O(|\Phi'|(2k_{\mathrm{MAX}})!)$ and by Lemma 1 $|\mathtt{NOT}| = 12$ hence the thesis holds.

—$\Psi[x_1^{k_1},\dots,x_{i_2}^{k_{i_2}},\dots,x_{i_1}^{k_{i_1}},\dots,x_n^{k_n}] = (\mathtt{AND}\ \Psi_1[x_1^{k_1},\dots,x_{i_1}^{k_{i_1}}]\ \Psi_2[x_{i_2}^{k_{i_2}},\dots,x_n^{k_n}])$ by inductive hypothesis $\exists J_1\subseteq\{1,\dots,i\}$, $\exists d_1\le m$, $\exists j_1\in\{1,\dots,n\}$ such that $\Psi_1 :!^{f(k_{j_1})+d_1+\sum_{\ell\in J_1}2(k_\ell-1)}\mathtt{B}$ and

$$x_h^{k_h} :!^{\max\{2k_h-7,1\}+f(k_{j_1})-f(k_h)+d_1+\sum_{\ell\in J_1}2(k_\ell-1)} M_{k_h}$$

for $1\le h\le i_1$, and exists $J_2\subseteq\{1,\dots,i\}$, $\exists d_2\le m$, $\exists j_2\in\{1,\dots,n\}$ such that $\Psi_2 :!^{f(k_{j_2})+d_2+\sum_{\ell\in J_2}2(k_\ell-1)}\mathtt{B}$ and

$$x_h^{k_h} :!^{\max\{2k_h-7,1\}+f(k_{j_2})-f(k_h)+d_2+\sum_{\ell\in J_2}2(k_\ell-1)} M_{k_h}$$

for $i_2\le h\le n$.

spiegare la figura con l'and

Without loss of generality suppose

$$f(k_{j_1})+d_1+\sum_{\ell\in J_1}2(k_\ell-1)\ge f(k_{j_2})+d_2+\sum_{\ell\in J_2}2(k_\ell-1)$$

then we can construct $f(k_{j_1})+d_1+\sum_{\ell\in J_1}2(k_\ell-1)-(f(k_{j_2})+d_2+\sum_{\ell\in J_2}2(k_\ell-$

1)) boxes [2] around Ψ_2. Then

$$\Psi_2 :!^{\begin{array}{l} f(k_{j_2})+d_2+\sum_{\ell\in J_2} 2(k_\ell-1)+ \\ +f(k_{j_1})+d_1+\sum_{\ell\in J_1} 2(k_\ell-1)- \\ -(f(k_{j_2})+d_2+\sum_{\ell\in J_2} 2(k_\ell-1)) \end{array}} \mathtt{B}$$

hence

$$\Psi_2 :!^{f(k_{j_1})+d_1+\sum_{\ell\in J_1} 2(k_\ell-1)}\mathtt{B}$$

For free variables is analogous.

(1) As above, suppose

$$f(k_{j_1}) + d_1 + \sum_{\ell\in J_1} 2(k_\ell - 1) \geq f(k_{j_2}) + d_2 + \sum_{\ell\in J_2} 2(k_\ell - 1)$$

and that we need to add $f(k_{j_1}) + d_1 + \sum_{\ell\in J_1} 2(k_\ell - 1) - (f(k_{j_2}) + d_2 + \sum_{\ell\in J_2} 2(k_\ell - 1))$ boxes around $\Psi_2[x_{i_2}^{k_{i_2}}, \ldots, x_n^{k_n}]$ as in Figure 17. Then

$$\begin{array}{l} d(\Psi[x_1^{k_1}\ldots x_n^{k_n}] :!^{f(k_{j_1})+d_1+\sum_{\ell\in J_1} 2(k_\ell-1)}\mathtt{B}) = \\ = \max\left\{\begin{array}{l} d(\mathtt{AND} :!^{f(k_{j_1})+d_1+\sum_{\ell\in J_1} 2(k_\ell-1)}\mathtt{B}), \\ d(\Psi_1[x_1^{k_1}, \ldots, x_{i_1}^{k_{i_1}}] :!^{f(k_{j_1})+d_1+\sum_{\ell\in J_1} 2(k_\ell-1)}\mathtt{B}), \\ \left(\begin{array}{l} d(\Psi_2[x_{i_2}^{k_{i_2}}, \ldots, x_n^{k_n}] :!^{f(k_{j_2})+d_2+\sum_{\ell\in J_2} 2(k_\ell-1)}\mathtt{B})+ \\ \quad +f(k_{j_1}) + d_1 + \sum_{\ell\in J_1} 2(k_\ell - 1)- \\ \quad -(f(k_{j_2}) + d_2 + \sum_{\ell\in J_2} 2(k_\ell - 1)) \end{array}\right) \end{array}\right\} \\ = \max\left\{\begin{array}{l} f(k_{j_1}) + d_1 + \sum_{\ell\in J_1} 2(k_\ell - 1), \\ f(k_{j_1}) + d_1 + \sum_{\ell\in J_1} 2(k_\ell - 1), \\ \left(\begin{array}{l} f(k_{j_2}) + d_2 + \sum_{\ell\in J_2} 2(k_\ell - 1)+ \\ \quad +f(k_{j_1}) + d_1 + \sum_{\ell\in J_1} 2(k_\ell - 1)- \\ \quad -(f(k_{j_2}) + d_2 + \sum_{\ell\in J_2} 2(k_\ell - 1)) \end{array}\right) \end{array}\right\} \\ = f(k_{j_1}) + d_1 + \sum_{\ell\in J_1} 2(k_\ell - 1) \end{array}$$

(2) It is easy to see that the deepest type is the final one, of depth

$$d(!^{f(k_j)+d+\sum_{\ell\in J} 2(k_\ell-1)}\mathtt{B}) = f(k_j) + d + \sum_{\ell\in J} 2(k_\ell - 1) + 1.$$

(3) By inductive hypothesis $|\Psi'[x_1^{k_1}, \ldots, x_n^{k_n}]|$ is $O(|\Phi'|(2k_{\mathrm{MAX}})!)$ and by Lemma 1 $|\mathtt{AND}| = 21$ hence the thesis holds.

—$\Psi[x_1^{k_1}, \ldots, x_{i_2}^{k_{i_2}}, \ldots, x_{i_1}^{k_{i_1}}, \ldots, x_n^{k_n}] = (\mathtt{OR}\ \Psi_1[x_1^{k_1}, \ldots, x_{i_1}^{k_{i_1}}]\ \Psi_2[x_{i_2}^{k_{i_2}}, \ldots, x_n^{k_n}])$ as above.

—$\Psi[x_1^{k_1}, \ldots, x_n^{k_n}] = (\mathbf{D}_{\mathbf{k_i}}\ \lambda y^{k_i}.(\mathtt{AND}\ (\lambda x^{k_i}.\Psi'[x_1^{k_1}, \ldots, x_n^{k_n}]\ y^{k_i}))\ \mathtt{true})$ by inductive hypothesis Ψ' has m' quantifiers over $\mathcal{D}_1, \ldots, \mathcal{D}_{i'}$ and there exist

$$\begin{array}{rcl} J' & \subseteq & \{1, \ldots, i'\}, \\ j' & \in & \{1, \ldots, n\}, \\ d' & \leq & m' \end{array}$$

[2] Remember that "constructing n boxes around a term" is always possible in EAL for the presence of the !-introduction rule.

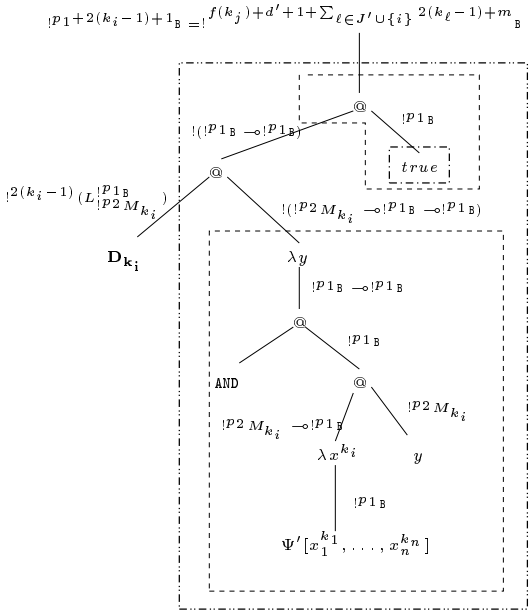

Fig. 18. EAL type of an arbitrary formula

such that

$$\Psi' :!^{f(k_{j'})+d'+\sum_{\ell\in J'} 2(k_\ell-1)}\mathsf{B}$$

with free variables

$$x_h^{k_h} :!^{\max\{2k_h-7,1\}+f(k_{j'})-f(k_h)+d'+\sum_{\ell\in J'} 2(k_\ell-1)} M_{k_h}.$$

Now Ψ has $m = m' + 1$ quantifiers over $\mathcal{D}_1, \ldots, \mathcal{D}_{i'}, \mathcal{D}_i$, then we can type Ψ in EAL adding $2(k_i - 1)$ boxes as in Figure 18 where the bold box represents $2(k_i - 1)$ boxes, $p_1 = f(k_j) + d' + \sum_{\ell\in J'} 2(k_\ell - 1)$ and $p_2 = \max\{2k_i - 7, 1\} + f(k_j) - f(k_i) + d' + \sum_{\ell\in J'} 2(k_\ell - 1)$. Hence the thesis holds with $J = J' \cup \{i\}, j = j'$ and $d = d' + 1$.

(1)

$$d(\Psi[x_1^{k_1}\dots x_n^{k_n}] :!^{f(k_{j'})+(d'+1)+\sum_{\ell\in J'\cup\{h\}}2(k_\ell-1)}\mathtt{B}) =$$
$$= \max\left\{\begin{array}{l} d(\mathtt{true} :!^{f(k_{j'})+d'+\sum_{\ell\in J'}2(k_\ell-1)}\mathtt{B}) + 2(k_h-1)+1,\\ d(\mathtt{AND} :!^{f(k_{j'})+d'+\sum_{\ell\in J'}2(k_\ell-1)}\mathtt{B}) + 2(k_h-1)+1,\\ d(\Psi'[x_1^{k_1}\dots x_n^{k_n}] :!^{f(k_{j'})+d'+\sum_{\ell\in J'}2(k_\ell-1)}\mathtt{B}) + 2(k_h-1)+1,\\ d(\mathbf{D_{k_h}} :!^{2(k_h-1)} L^{\cdots}_{!^{\max\{2k_h-7,1\}+f(k_{j'})-f(k_h)+d'+\sum_{\ell\in J'}2(k_\ell-1)}M_{k_h}}) \end{array}\right\}$$
$$= \max\left\{\begin{array}{l} f(k_{j'})+d'+\sum_{\ell\in J'}2(k_\ell-1)+2(k_h-1)+1,\\ f(k_{j'})+d'+\sum_{\ell\in J'}2(k_\ell-1)+2(k_h-1)+1,\\ f(k_{j'})+d'+\sum_{\ell\in J'}2(k_\ell-1)+2(k_h-1)+1,\\ 1+3(k_h-1)+\sum_{\ell=0}^{k_h-1}n_\ell \end{array}\right\}$$

but $!^{\max\{2k_h-7,1\}+f(k_{j'})-f(k_h)+d'+\sum_{\ell\in J'}2(k_\ell-1)}M_{k_h} =$
$= \Delta_{k_h-1}^{(\max\{2k_h-7,1\}+f(k_{j'})-f(k_h)+d'+\sum_{\ell\in J'}2(k_\ell-1),1,\dots)}$ hence

$$= \max\left\{\begin{array}{l} f(k_{j'})+(d'+1)+\sum_{\ell\in J'\cup\{h\}}2(k_\ell-1),\\ 1+3(k_h-1)+\max\{2k_h-7,1\}+f(k_{j'})-\\ \quad -f(k_h)+d'+\sum_{\ell\in J'}2(k_\ell-1)+(k_h-1) \end{array}\right\}$$
$$= \max\left\{\begin{array}{l} f(k_{j'})+(d'+1)+\sum_{\ell\in J'\cup\{h\}}2(k_\ell-1),\\ f(k_{j'})+(d'+1)+\sum_{\ell\in J'\cup\{h\}}2(k_\ell-1)+\\ \quad +2(k_h-1)+\max\{2k_h-7,1\}-f(k_h) \end{array}\right\}$$

$$2(k_h-1)+\max\{2k_h-7,1\}-f(k_h)=0 \text{ then}$$

$$= f(k_{j'})+(d'+1)+\textstyle\sum_{\ell\in J'\cup\{h\}}2(k_\ell-1)$$

(2) Looking at Figure 18, the deepest type is again the final one, indeed

$$d(!^{2(k_i-1)}(L^{!^{p_1}\mathtt{B}}_{!^{p_2}M_{k_i}})) =$$
$$= 2(k_i-1)+1+\max\{p_1+1, p_2+d(M_{k_i})\}$$
$$= 2(k_i-1)+1+\max\left\{\begin{array}{l} f(k_j)+d'+\sum_{\ell\in J'}2(k_\ell-1)+1,\\ \max\{2k_i-7,1\}+f(k_j)-f(k_i)+d'+\\ \quad +\sum_{\ell\in J'}2(k_\ell-1)+2k_i-1 \end{array}\right\}$$
$$\text{but } f(k_i)=2(k_i-1)+\max\{2k_i-7,1\}, \text{ then}$$
$$= 2(k_i-1)+1+\max\left\{\begin{array}{l} f(k_j)+d'+\sum_{\ell\in J'}2(k_\ell-1)+1\\ f(k_j)+d'+\sum_{\ell\in J'}2(k_\ell-1)+1 \end{array}\right\}$$
$$= 2(k_i-1)+1+f(k_j)+d'+\sum_{\ell\in J'}2(k_\ell-1)+1$$
$$= f(k_j)+(d'+1)+\sum_{\ell\in J'\cup\{i\}}2(k_\ell-1)+1$$
$$= d(!^{f(k_j)+d+\sum_{\ell\in J}2(k_\ell-1)}\mathtt{B}) \quad (4)$$

(3) An arbitrary formula Φ is built up from `AND`, `OR`, `true`, `false`, `IFF`, `NOT`, $\mathbf{D_i}$, $member_j$ all bounded by $O(d(2k_{\mathrm{MAX}})!)$, then the size of the

term encoding Φ is $O(|\Phi|d(2k_{\mathrm{MAX}})!)$.

—$\Psi[x_1^{k_1},\ldots,x_n^{k_n}] = (\mathbf{D}_{\mathbf{k_i}}\ \lambda y^{k_i}.(\mathtt{OR}\ (\lambda x^{k_i}.\Psi'[x_1^{k_1},\ldots,x_n^{k_n}]\ y^{k_i}))\ \mathtt{false})$ as above.

□

The following bound on the number of !'s in the type of a lambda term encoding an arbitrary formula provides a limitation also for the box-nesting depth.

COROLLARY 1. *Let Ψ be a term encoding an arbitrary formula. Then Ψ has type in EAL $!^t\mathtt{B}$ with $t = O(n \cdot k_{\mathrm{MAX}})$ where k_{MAX} is the greatest k such that $member_k$ appears in Ψ and n is the number of quantifiers in Ψ.*

PROOF. By Theorem 5 Ψ has type $!^t\mathtt{B}$ with $t = f(k_j) + m + \sum_{\ell\in J} 2(k_\ell - 1) \leq f(k_{\mathrm{MAX}}) + n + 2n(k_{\mathrm{MAX}} - 1)$. □

In order to obtain the desired result on complexity of duplication, it remains to be shown that the pre-compilation of the λ-terms given by eta-expansion can be performed inside EAL.

THEOREM 6. *If M has an EAL type, so does $\mathbf{or}(M)$.*

PROOF. It is sufficient to prove that for any EAL type σ the η-expansion $\eta(x)_\sigma$ is always typeable in EAL, as it is described in Figure 20. □

THEOREM 7. *Let Ψ a term encoding an arbitrary formula. Then $d(\mathbf{or}(\Psi)) = O(n \cdot k_{\mathrm{MAX}})$, where n is the number of quantifiers in Ψ and k_{MAX} is the greatest k such that $member_k$ appears in Ψ.*

PROOF. It is sufficient to investigate depth of types in the derivation of Ψ. Indeed, depth of $\mathbf{or}(\Psi)$ cannot exceed depth of Ψ plus maximal depth of a type in the derivation of Ψ, because, in the worst case, we can η-expand the deepest variable of deepest type in Ψ and $d(\eta(x)_\sigma)$ is trivially $d(\sigma)$.

□

This is what is needed to obtain Theorem 1 for EAL-typed terms, and, hence, our Theorem 4.

5. CONCLUSIONS

In this paper, we have shown that the complexity result for parallel beta-reduction obtained in [Asperti and Mairson 1998] can be obtained with respect to just (optimal) duplication, and not only (as one could have expected) to the complex machinery required for the management of redex-families. This casts a new and different light on Asperti and Mairson's work; in particular, it helps to understand why it does not convey any negative result for optimal reduction as a technique for the normalization of lambda-terms. In any reduction of a lambda term there is, potentially, a non elementary work of duplication that has to be done, and even an optimal reduction technique, as parsimonious as it could be, cannot do any magic.

A lot of interesting issues are still to be investigated, starting from a theoretical comparison of the performance of Lamping's technique versus more traditional

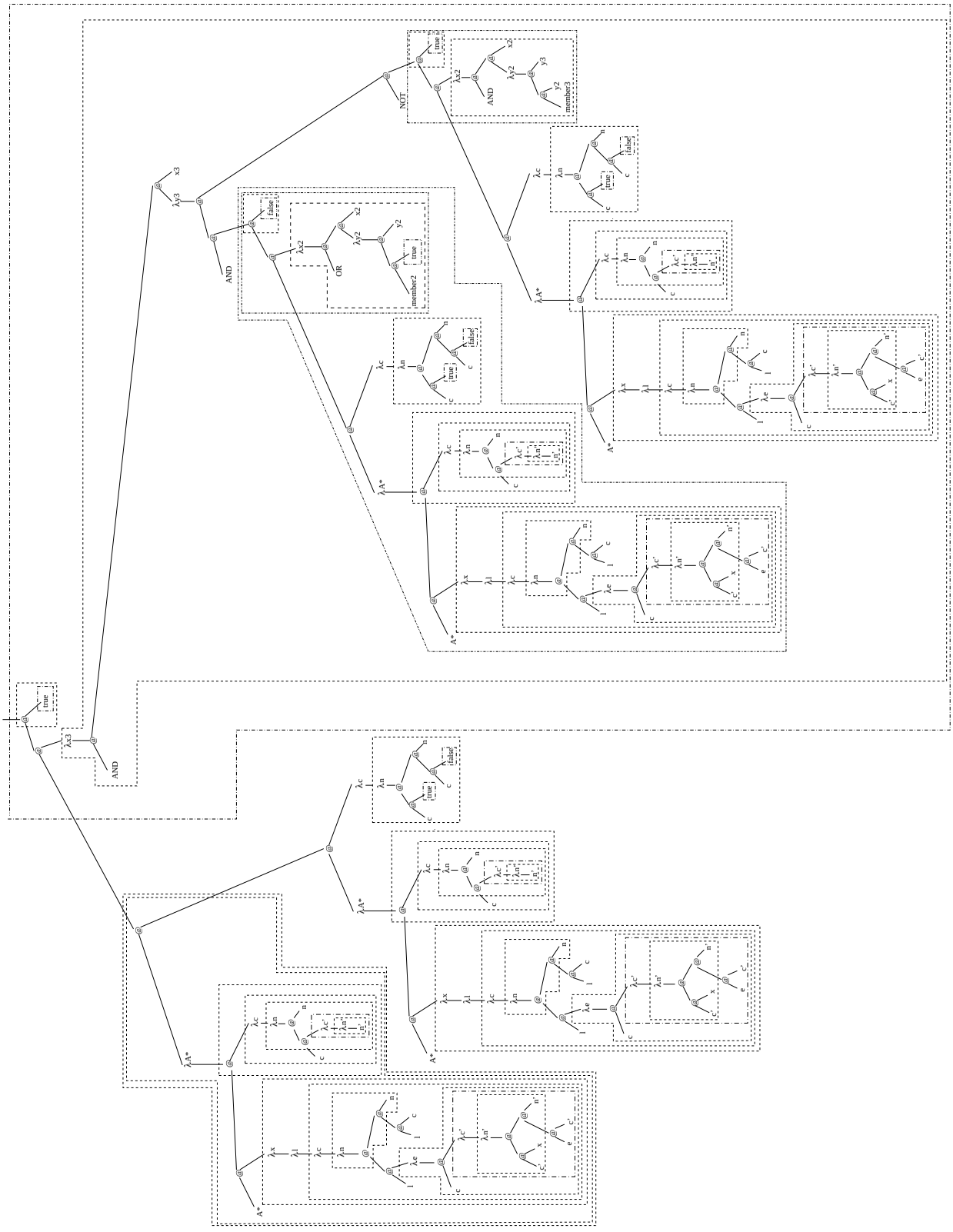

Fig. 19. $\forall x^3(\exists x^2\ true \in x^2\ \wedge\ \neg(\forall x^2\ x^2 \in x^3))$

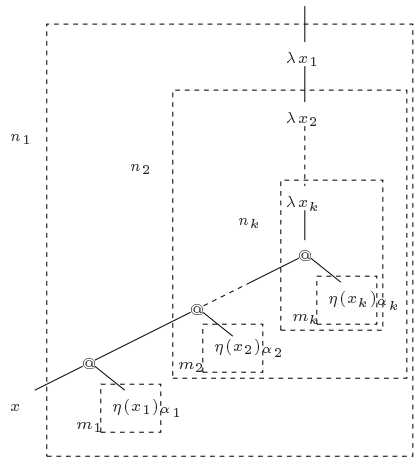

Fig. 20. $\eta(x)_{!^{n_1}(!^{m_1}\alpha_1 \multimap !^{n_2}(!^{m_2}\alpha_2 \multimap \cdots !^{n_k}(!^{m_k}\alpha_k \multimap !^{m_{k+1}}o)\cdots))}$

implementations. This comparison has been hindered so far by the "bookkeeping" work, whose formal investigation is just too complex with the current state of the art. But now we have a huge set of lambda terms (i.e. all the terms of EAL) that can be reduced without the need of bookkeeping, providing a main and very promising arena for a theoretical investigation of performance issues.

REFERENCES

Asperti, A. 1995. Linear logic, comonads and optimal reductions. *Fundamentae Informaticae 22*, 3–22.

Asperti, A. 1996. On the complexity of beta-reduction. In *Conference Record of POPL '96: The* 23rd *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (St. Petersburg Beach, Florida, 21–24 Jan. 1996), pp. 110–118.

Asperti, A. 1998. Light affine logic. In *Proc. of Symposium on Logic in Computer Science* (1998), pp. 300–308.

Asperti, A. and Guerrini, S. 1998. *The Optimal Implementation of Functional Programming Languages*, Volume 45 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Asperti, A. and Laneve, C. 1997. On the dynamics of sharing graphs. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela Eds., *Automata, Languages and Programming, 24th International Colloquium*, Volume 1256 of *Lecture Notes in Computer Science* (Bologna, Italy, 7–11 July 1997), pp. 259–269. Springer-Verlag.

Asperti, A. and Mairson, H. G. 1998. Parallel beta reduction is not elementary recursive. In *Conference Record of POPL '98: The 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (San Diego, California, 19–21 Jan. 1998), pp. 303–315.

Girard, J.-Y. 1987. Linear logic. *Theoretical Computer Science 50*, 1–102.

Girard, J.-Y. 1998. Light linear logic. *Information and Computation 204*, 143–175.

Gonthier, G., n Abadi, M., and Lévy, J.-J. 1992a. The geometry of optimal lambda reduction. In *Conference record of the 19th Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages* (January 1992), pp. 15–26.

Gonthier, G., n Abadi, M., and Lévy, J.-J. 1992b. Linear logic without boxes. In *Proc. of the 7th Annual IEEE Symposium on Logic in Computer Science* (June 1992), pp. 223–234.

KATHAIL, V. K. 1990. *Optimal Interpreters for Lambda-calculus Based Functional Programming Languages*. Ph. D. thesis, MIT.

LAMPING, J. 1990. An algorithm for optimal lambda calculus reduction. In ACM Ed., *POPL '90. Proceedings of the seventeenth annual ACM symposium on Principles of programming languages, January 17–19, 1990, San Francisco, CA* (New York, NY, USA, 1990), pp. 16–30. ACM Press.

LAWALL, J. L. AND MAIRSON, H. G. 1996. Optimality and inefficiency: What isn't a cost model of the lambda calculus? *ACM SIGPLAN Notices 31*, 6 (June), 92–101.

LAWALL, J. L. AND MAIRSON, H. G. 1997. On global dynamics of optimal graph reduction. In *Proceedings of the 1997 ACM SIGPLAN International Conference on Functional Programming*, Volume 32 (Amsterdam, The Netherlands, 9–11 June 1997), pp. 188–195. ACM Press.

LÉVY, J.-J. 1978. *Réductions Correctes et Optimales dans le Lambda Calcul*. Ph. D. thesis, University of Paris VII.

MAIRSON, H. G. 1992. A simple proof of a theorem of Statman. *Theoretical Computer Science 103*, 2 (Sept.), 387–394.

MEYER, A. R. 1974. The inherent computational complexity of theories of ordered sets. In *Proceedings of the International Congress of Mathematicians* (1974), pp. 477–482.

STATMAN, R. 1979. The typed λ-calculus is not elementary recursive. *Theoretical Computer Science 9*, 73–81.