Objects and their Lambda Calculus

Athanassios Tzouvaras
Dept. of Mathematics, University of Thessaloniki,
540 06 Thessaloniki, Greece
E-mail: tzouvara@ccf.auth.gr

Abstract

A kind of parallel typed lambda calculus is presented based on the language and structure of objects. The term "object" is used here in a sense different from that related to the expression "object-oriented language (programming)". By "objects" here we mean any class of entities which (a) are resource dependent and (b) combine to each other (via some fitness relation) to form more complex ones.

Two operators, λ and its dual $\bar{\lambda}$, are used, and two operations, a binary one, $\odot$, for juxtaposition, and an n-ary one, $|$, for every n, for branching. The construct $\lambda v.x$ represents, roughly, a receiving scheme producing copies of x when fed with proper objects y to fill the empty place v of x, while the dual construct $\bar{\lambda} y.z$ represents a sending scheme that throws y out z in proper surroundings. The interaction of these two constructs takes place when they are matched together by $\odot$ and yields an exchange of resources in a way that preserves the total amount of them. The calculus captures such notions as *concurrency*, *interaction* and *branching* in a way analogous to that of [4] and [5], but with a quite different meaning of the operations. What is described here is "situations" of coexisting entities rather than computations, and resource- preserving transformations between them. The terms are shown to have unique normal forms. Object structures that model the simple theory of objects are extended here in suitable graph structures that provide sound and complete semantics of the calculus.

Key words. Structure of objects, λ-calculus, concurrency, branching.

1 Introduction.

By "objects" we shall mean throughout any class of entities which (a) are resource dependent and (b) combine to each other (via some specific fitness relation) to form more complex ones. One may think of them as "material" entities existing in time and consuming resources, in contrast to abstract, timeless entities of set theory. Therefore the theory of objects referred to here has very little in common e.g. with L. Cardelli's theory of primitive objects, and surely is not designed for applications to programming. What then is its use?

Originally a formal study of the structure of objects was taken up in [7] and [8] with the purpose to treat philosophical questions concerning material objects and their identity by logical means. Soon however it was made clear to the author that the behavior of such objects obey rules which not long ago had been isolated by J.-Y. Girard as mere syntax, namely linear rules. That material objects have their own logic (of existence and change), and this is a fragment of linear logic, was shown in [9] and [**?**], where objects were formally represented by *multisets*. So one can think of objects as the natural semantics of the multiplicative part of linear logic, exactly as abstract sets is the natural boolean semantics of classical logic.

In the present paper we extend the preceding idea to λ-calculus. Namely, if classical λ-calculus encodes the principles of formation and action of abstract functions, how would a system of "material" transformations, respecting resource consumption and fitness conditions, look like? In fact a system of concurrent typed λ-calculus is presented based on the language and structure of objects that captures parallely, interaction and branching. Two basic operators are used, a *receiver* λ and a *sender* $\bar{\lambda}$, and two operations, a binary one "$\odot$" for parallel existence (coexistence) and a binary one "$|$" for branching existence. Since $|$ is associative it can be generalized to an n-ary operation $t_1|\cdots|t_n$ for every n. $(\lambda v.x)$ represents an object x in which the empty place v is *activated*, i.e., it is ready to receive as input an object y of the type of v. $(\bar{\lambda}y.z)$ on the other hand, represents an object z whose part y is *activated*, i.e., is ready to leave z leaving behind an empty place v. These two constructs interact when matched together by $\odot$, i.e., when the term

$$(\lambda v.x \odot \bar{\lambda}y.z)$$

makes sense, and are transformed (reduced) to the term

$$(x[y/v] \odot z[v/y]).$$

Thus the basic reduction rule of our calculus is the following analog of β-conversion

$$(\lambda v.x \odot \bar{\lambda} y.z) = (x[y/v] \odot z[v/y]).$$

The calculus has some points in common with that of [4] and especially [5], from which the notational machinery is borrowed. However, it differs essentially in the *semantics* and the properties of the operations. To be specific Boudol's calculus is computational in character, while ours might be called "situational", as it describes branching *situations* of coexisting entities and their transformations. Our axioms and reduction rules aim to capture *resource-preserving* transformations of such situations.

The paper is organized as follows: In section 2 we outline the theory of objects. Sections 3 and 4 contain the main Section 4 contains the formal systems $\lambda^{\mathbf{o}}$ and $\lambda^{\mathbf{o}}\theta$, the notions of reductions and the normalization results. Section 5 presents a graph-theoretic semantics of these calculi and the soundness and completeness results.

2 The structure of objects

The language L of the formal theory of objects considered below will contain the following symbols:

1) *Object variables* $x, y, z, \ldots$.

2) *Set variables* $X, Y, Z, \ldots$ ranging over sets of objects.

3) A binary relation symbol F for the *fitness relation*.

4) A binary (partial) operation symbol "$\cdot$" for the *assembly or plugging operation*.

5) A binary relation symbol *Type* for the predicate "x, y are of the same type".

For simplicity we may write xy instead of $x \cdot y$. (This is what in [7] and [8] is denoted by $x \Box y$). The intended meaning of "$\cdot$" is that whenever xy is defined and $xy = z$, then z is the new object resulting by plugging together x and y.

We make free use of the concepts and notation of intuitive set theory including those of natural numbers. Throughout $m, n, k, i, j, ...$ will range over positive integers.

Our intuition about objects draws mainly from the class of *artificial* objects (i.e., objects made by humans) rather than that of *natural* ones. (The reader can find in [7] an informal discussion about similarities and differences between natural and artificial objects). This is reflected of course on the principles we adopt concerning their behavior.

Below we introduce the axioms of objects, step by step, together with the relevant notions involved and the necessary discussion.

$$\text{(O1)} \qquad xFy \Longleftrightarrow (\exists z)(xy = z).$$

The partiality of "$\cdot$" stems of course from the fact that not any object fits (or matches) with any other in order to produce a new entity. The assembly operation is commutative,

$$\text{(O2)} \qquad xy = yx,$$

though not associative. Thus in general $x(yz) \neq (xy)z$.

Definition 2.1 x is said to be an *immediate part* of y, in symbols $x <_0 y$, if for some z, $y = xz$. x is a *proper part* of y, if $x < y$, where $<$ is the transitive closure of $<_0$, i.e., if there are objects $z_1, z_2, \ldots, z_n$, for some $n \in N$, such that $x <_0 z_1 <_0 \ldots <_0 z_n <_0 y$. Finally x is a *part* of y, in symbols $x \leq y$, if $x < y$ or $x = y$. An object x is said to be *atomic* or *atom* if it has no proper parts, i.e., $(\forall y, z)(yz \neq x)$. *Atom* denotes the class of atomic objects. We denote also by $P(x)$ and $P_a(x)$ the sets of parts and atomic parts of x, respectively. The letters a, b, c,... range over atoms.

Parthood of artificial objects is well-founded. This is postulated by the principle below:

$$\text{(O3)} \qquad \textit{There is no infinite sequence} \quad \ldots < x_n < \ldots < x_1 < x_0.$$

Proposition 2.2 $x < y \rightarrow x \neq y$.

It follows from (O3) that every descending $<$-chain is finite and ends up with an atomic object. On the other hand, it does not yet imply that the sets $P_a(x)$ and $P(x)$ are finite. We have not excluded e.g. the possibility that for distinct pairs of objects $\{a_i, b_i\}$, $i \in N$, $a_i F b_i$ and $a_i b_i = a_j b_j$. In such a case the atoms of $x = a_1 \cdot b_1$ would comprise all a_i, b_i, $i \in N$.

In [8] and [7] we defined equality of artifacts in a rather restrictive way, namely $xy = x'y'$ iff $\{x, y\} = \{x', y'\}$. Here we shall be more liberal, allowing the same object to be constructed in more than one ways, but always using the same atoms.

Definition 2.3 We say that the objects x, y *overlap, if they have parts in common, i.e.,* $P(x) \cap P(y) \neq \emptyset$. *If* $P(x) \cap P(y) = \emptyset$ *we say that* x, y *are parallel* and write $x \parallel y$.

Overlapping objects share a number of parts. That means they are not independent entities, hence they cannot coexist, since ontological independence is a prerequisite of coexistence. A fortiori overlapping objects cannot fit together in order to produce a new object, since fitting presupposes coexistence. Thus we postulate

(O4) $$xFy \Rightarrow x \parallel y.$$

Let $x < y$. By the definition of $<$, there is a finite sequence z_1,...,z_n of objects (not necessarily unique) such that

$$y = (\ldots((xz_1)z_2)\ldots)z_n.$$

The sequence z_1,...,z_n is called an *analysis of* y *over* x. We might also have for the same x, y, another analysis

$$y = (\ldots((xu_1)u_2)\ldots)u_m.$$

Due to axiom (O4), the objects z_1,...,z_n are not only distinct, but pairwise non-overlapping, therefore every such analysis of y can be represented by a *binary tree.*

If we analyze further the parts x, z_1,...,z_n above we obtain a *full analysis* of y and a *full* binary tree corresponding to that. This tree is finitely branching (namely at most doubly branching at each node) and each branch is finite according to axiom (O3). Therefore it is finite with all terminal nodes labelled by atoms. We shall call such a tree, a *full analysis tree* of y.

Let $T(x)$ be the set of full analysis trees of x, and for every $t \in T(x)$ let $term(t)$ be the set of atoms appearing at the terminal nodes of t.

(O5) $$x = y \iff (\forall t_1 \in T(x))(\forall t_2 \in T(y))(term(t_1) = term(t_2)).$$

As an immediate consequence of (O4) we get:

Proposition 2.4 *(i) For every $t \in T(x)$, $term(t) = P_a(x)$.*
(ii) $P_a(x) = P_a(y) \Rightarrow x = y$.
(iii) The sets $P_a(x)$ and $P(x)$ are finite.

2.1 Copies: Object isomorphism vs replaceability.

There are two basic criteria for deciding whether two objects x, y are copies of one another: Either (a) by looking *inwards*, i.e., the internal structure of x, y, or (b) by looking *outwards*, i.e., their ability to be mutually interchangeable as parts of larger objects. In the first case the criterion is *structural*, that is, the isomorphism of objects as algebraic structures. In the second case the criterion is *operational*, that is their replaceability with respect to the fitness relation. Both of them are incomplete and in a sense supplementary. For instance it is easy to see that the second criterion does not imply the former. Indeed, we can imagine two objects (xy) and z, of which the first is made of the parts x, y, while the second is disposable (hence atomic), and yet interchangeable in all assemblies. Furthermore, the operational criterion is *relative*; it depends on the particular *world* in which x, y are contained and the availability of other objects of which x, y may be parts.

On the other hand the first criterion cannot apply to *atomic* objects since they lack structure. Thus we need a mixture of the two criteria. The decisive step is to determine which *atoms* would be copies of one another. And this can be defined only in principle, i.e., by a primitive notion of similarity, partitioning the class of atoms into *types*. This is the intended meaning of the symbol $Type(x, y)$.

(O6) $Type(x, y)$ *is an equivalence relation on atoms.*

The equivalence class of a under *Type* is written $Type(a)$, so $Type(a,b)$ holds iff $Type(a) = Type(b)$.

Definition 2.5 The relation $x \cong y$, *(x,y are copies of each other),* is defined as follows: (a) If x, y are atoms then $x \cong y$ iff $Type(x, y)$. (b) x, y are non-atoms, $x \cong y$ if there is a bijection $f : P(x) \to P(y)$ such that $f(a) \cong a$ for every atom $a \in P(x)$ and $f(x_1 x_2) = f(x_1)f(x_2)$ for $x_1, x_2 \in P(x)$.

Clearly $\cong$ is an equivalence relation that extends *Type*, i.e., $Type \subseteq \cong$, so we can put also

$$Type(x) = \{y : y \cong x\}$$

for the equivalence class of x under $\cong$. Thus every universe M of objects satisfying the axioms is a *typed* set and we can write TYPE1, TYPE2,..., TYPEn for the basic types of its atoms, i.e., the equivalence classes of *Type* on the atoms of M. We turn now to replaceability. The precise formulation of this notion is a bit more intriguing.

We have seen that if z_1,...,z_n is an analysis of y over x, then no two of the objects x, z_1,...,z_n overlap. In order now for another object x' to be able to replace x inside y, it is clearly necessary to coexist or, at least, not to overlap with z_1,...,z_n. We denote this by $x' \parallel (y - x)$, i.e.,

$$x' \parallel (y - x) := [P(x') \cap (P(y) - P(x)) = \emptyset]\,.$$

Definition 2.6 Let $x < y$ and x' be given. We say that $y[x'/x]$ *exists* if for every analysis of z_1,...,z_n of y over x, the object $(\ldots((x'z_1)z_2)...)z_n$ is defined. So we let

$$y[x'/x] = (\ldots((x'z_1)z_2)...)z_n.$$

We say that *x is replaceable by x' in y,* in symbols $Rep(x, x', y)$, if either $x \not< y$ or $x' \not\parallel (y - x)$ or $y[x'/x]$ exists. That is:

$$Rep(x, x', y) := [x < y \;\&\; x' \parallel (y - x) \Longrightarrow (\exists z)(z = y[x'/x])]\,.$$

Finally let

$$re(x, y) := (\forall z)(Rep(x, y, z) \;\&\; Rep(y, x, z)).$$

The primitive notion of type should obey some rules with respect to fitness and assembly and a natural such rule is the following:

(O7) $$(\forall a, b \in Atom)(Type(a, b) \Rightarrow re(a, b)).$$

The two notions of copy, the internal one based on isomorphism and the external one based on replaceability, are comparable but not identical.

Proposition 2.7 *(i) For all x, y, z, if $x \cong y$, $x < z$ and $y \parallel (z - x)$, then $z[y/x]$ exists.*

(ii) For all x, y, $x \cong y \Rightarrow re(x, y)$.

(iii) If xFy, $x' \cong x$ and $x' \parallel y$, then $x'Fy$ and $xy \cong x'y$.

(iv) If $x \cong y$, $x < z$ and $y \parallel (z - x)$, then $z[y/x] \cong z$.

The converse of proposition 2.7 (ii) need not be true. Two objects may be mutually replaceable without being copies of one another. In many practical situations the supply of copies for each particular atomic part is unlimited. If we add this as a principle, we can show that $re(x, y)$ is transitive, hence an equivalence relation.

(O8) $$(\forall a \in Atom)(Type(a) \text{ is infinite}).$$

Proposition 2.8 ((O8)) *(i) $x \cong y$ & $re(y, z) \Rightarrow re(x, z)$.*

(ii) $re(x, y)$ is an equivalence relation.

2.2 Generalizing the assembly operation.

In the preceding treatment of objects the restriction imposed was that every non-atomic object has *exactly two* immediate parts, that is, every object is produced by combining only two pre-constructed objects at a time. One might consider this limitation as unnecessary and propose instead that an object could be produced by simultaneous fitting together finitely many parts. This has an impact only on the notion of *immediate part* and not on that of part in general. In this case we would have to replace the partial binary assembly operation $\cdot$ by a partial operation $[x_1 x_2 \ldots x_n]$ with a finite but unfixed number of arguments with the obvious intended meaning: Whenever $[x_1 \ldots x_n]$ is defined, the object $y = [x_1 \ldots x_n]$ is the outcome of plugging together (at one step) $x_1, \ldots, x_n$. Fitness is also extended to a relation $F \subseteq \bigcup_{n=1}^{\infty} O^n$, where O is the class of objects. Intuitively $F(x_1, \ldots, x_n)$

holds if $\{x_1, \dots, x_n\}$ is a subset of a set $\{x_1, \dots, x_n, x_{n+1}, \dots, x_{n+m}\}$ such that the object $[x_1 \dots x_{n+m}]$ exists.

For a set X of objects put $P_a(X) = \bigcup\{P_a(x) : x \in X\}$. Write also $\parallel (X)$ if $(\forall x, y \in X)(x \parallel y)$. I.e., $\parallel (X)$ iff the objects of X are pairwise disjoint.

If $x = [x_1 \dots x_n]$, the x_i's are said to be *immediate parts* of x, notation $x_i <_0 x$, and the transitive closure of $<_0$ is the parthood relation $<$. In most cases the object-notation mentioning the objects $\{x_1, \dots, x_n\}$ can be replaced by set-notation employing the symbol X denoting the preceding set. So if $X = \{x_1, \dots, x_n\}$ we can write $[X]$ instead of $[x_1 \dots x_n]$. Also the notation $[xX]$ has the obvious meaning. Given $x < y$, the *analysis of y over x* has now the form

$$y = [[\dots[[xz_{11} \dots z_{1k_1}]z_{21} \dots z_{2k_2}] \dots]z_{n1} \dots z_{nk_n}]$$

or, putting $Z_i = \{z_{i1}, \dots, z_{ik_i}\}$, $i \leq n$, and using the set-notation, the preceding equation is written

$$y = [[\dots[[xZ_1]Z_2] \dots]Z_n].$$

Also we write F(X) for the fitness relation, etc.

The *analysis trees* of x are defined again in the obvious way. The only difference is that these trees are not binary, but general finite. The axioms (O1)-(O8) cited above now take the following form.

(GO1) $$F(X) \Leftrightarrow (\exists y)(y = [X]).$$

(GO2) $$[x_1 \dots x_n] = [x_{f(1)} \dots x_{f(n)}],$$

for every permutation f of $\{1, \dots, n\}$.

(GO3) $$< \quad \textit{is wellfounded.}$$

(GO4) $$F(X) \Rightarrow \parallel (X)$$

(GO5) $$x = y \Longleftrightarrow (\forall t_1 \in T(x))(\forall t_2 \in T(y))(term(t_1) = term(t_2)).$$

$T(x)$ is again the set of full analysis trees of x. The relations $x \cong y$, $Rep(x, y, z)$ and $re(x, y)$ are also defined as before with the obvious adjustments. For example $x \cong y$ if there is a bijection $f : P(x) \to P(y)$ such that $f(a) \cong a$ for every atom $a \in P(x)$, and $f([x_1 \dots x_n]) = [f(x_1) \dots f(x_n)]$.

(GO6) $$Type(x, y) \quad \textit{is an equivalence relation on atoms.}$$

(GO7) $$(\forall a, b \in Atom)(Type(a, b) \Rightarrow re(a, b)).$$

(GO8) $$(\forall a \in Atom)(Type(a) \quad \textit{is infinite}).$$

3 Calculus of objects. The formal systems $\lambda^{\mathbf{o}}$ and $\lambda^{\mathbf{o}}\theta$.

In this section we assume familiarity of the reader with the fundamentals of classical λ-calculus, of type theory and typed λ-calculus. Excellent references for these subjects are [2], [**?**] and [3] respectively. The main advantage of dealing with objects instead of arbitrary computations or events lies in the use of copies. For each object x the class of copies of x behaves like a *type*. As we have seen in section 2.1, this is the equivalence class of x with respect to $\cong$, i.e., $Type(x) = \{y : y \cong x\}$. We shall assume that each type contains also *empty places* denoted by variables $v, u, ...$ and we write $v \in Type(x)$ or $x \cong v$ or $Type(x) = Type(v)$ for the fact that v is of $Type(x)$. Empty places fit to each other and to objects and take part in composite constructs just like objects. Objects, empty places as well as entities resulted by the combination of the latter under [...] will be referred to as *concrete terms* or just "objects" and are denoted also by the letters x, y, z. We have also *non-concrete* terms. These will be first

$$\lambda v.x \quad \text{and} \quad \bar{\lambda} x.y,$$

for any concrete terms x, y and any variable v. Next for any terms t, s such that $t \parallel s$, i.e. $P_a(t) \cap P_a(s) = \emptyset$, $t \odot s$ is a term. And for any terms t, s, $t|s$ is a term.

For concrete terms x, y, the parthood relation $x < y$ is defined in the obvious way. Now the intended meaning of the preceding terms is as follows:

• $\lambda v.x$: In x the empty place v (if contained) is *activated* and is ready to receive an object of the same type.

• $\bar{\lambda} x.y$: In y the part x (if contained) is *activated* and is ready to be thrown away leaving an empty place of the same type.

• $t \odot s$: Juxtaposition of the coexistent entities t and s (conjunction).

• $t|s$: A branching situation: Exactly one of the t, s can be present (exclusive disjunction).

λ and $\bar{\lambda}$ notation are adopted from Milner's calculus ([6]), used also by Boudol in his concurrent λ-calculus [5]. We shall refer to them as *binders* or *activators* and their role is to activate reception and leaving respectively.

The operations $\odot, |$ are also taken from [5] but their meanings here and there cannot be compared since Boudol deals with dynamic processes whereas we deal with static situations of existent objects. Worse, a comparison of the two approaches might confuse the reader because the meaning assigned to certain notions in the two contexts are rather contradictory. For example, in [5], p.151, Boudol says: "... $p|q$ consists in juxtaposing of p, q without any communication wire between them. This operator represents *concurrency*. The second construct, denoted $p \odot q$ and called *cooperation*, consists in plugging together p and q - up to termination of one of them". In contrast, as explained above, we denote juxtaposition by $\odot$ rather than $|$ and we use the word "concurrency" as synonymous to "parallely", a notion attributed to objects connected by $\odot$. $t|s$ here indeed implies non-communication but this is a result of their *incompatibility*, i.e., their non-coexistence.

Having fixed the above meanings to $\odot$ and $|$, (as "and" and "exclusive or" respectively) let us come to the precise formalization.

Though the binary case is more intuitive and its notation is much closer to the familiar lambda formalism, we shall prefer for reasons of economy to treat the generalized (finitary) case from which the binary calculus follows as a subcase.

I. Language. The language L^o of the intended calculus practically extends the language L of objects (see section 2), although L^o is not a logical language but an operational one. It consists of:

a) A set of *types* $\mathbf{T}^o = \{\tau, \sigma, \ldots\}$, usually finite. A subset $\mathbf{T}^o_a$ of $\mathbf{T}^o$ containing *atomic* types denoted by the letters $\alpha, \beta, \gamma, \ldots$.

b) A relation symbol F for *type fitness* and a function symbol $[\cdots]$ for *type composition*. The same symbols F and $[\cdots]$ will be used also for fitness and composition of terms.

c) A countable collection $V^\tau = \{v^\tau_0, v^\tau_1, \ldots\}$ of *variables* for each type τ. v^τ_i range over concrete terms of type τ. As a rule, however, neither the subscripts i nor the superscripts τ appear in practice. Instead we use the simplified notation $v, u, w, \ldots$ assuming that each such variable has a prescribed type.

d) A countable collection of *constants* $a^\alpha_i, i \geq 1$ for each type α, denoting *atoms of type* α.

e) The operators $\lambda, \bar{\lambda}$.

f) The binary operations $\odot$ and $|$.

II. Types and their axioms. The set of types $\mathbf{T}^o$ is given together with a *structure* on it, namely $[\cdots]$ is a mapping from certain subsets of $\mathbf{T}^o$ into $\mathbf{T}^o$

and we write $[\tau_1 \cdots \tau_n]$ for the composite type if $[\cdots]$ is defined at $\{\tau_1, \ldots, \tau_n\}$. The atomic types of $\alpha, \beta, \ldots$ of $\mathbf{T}^o_a$ are just the non-composite ones. The fact that $[\tau_1 \cdots \tau_n]$ exists is expressed also via F by writing $F(\tau_1, \ldots, \tau_n)$. That is, F is the domain of $[\cdots]$.

The following axioms concerning type composition and fitness are accepted:

$$F(\tau_1, \cdots, \tau_n) \Leftrightarrow (\exists \sigma)(\sigma = [\tau_1 \cdots \tau_n]). \tag{T1}$$

$$[\tau_{f(1)} \cdots \tau_{f(n)}] = [\tau_1 \cdots \tau_n] \tag{T2}$$

for every permutation f of $\{1, \ldots, n\}$.

$$\tau = \sigma \Leftrightarrow MAtom(\tau) = MAtom(\sigma). \tag{T3}$$

(T1) and (T2) are obvious. (T3) warrants that two objects assembled by the same atoms in different ways, and hence being identical according to axiom (GO5), have also identical types.

III. Terms. Next we come to terms. The concrete terms defined below are the syntactic analogs of objects.

Definition 3.1 The set $\mathbf{\Lambda}^o_c$ of *concrete terms* and their *types* are defined inductively as follows:

(a) Every variable v^τ and every constant a^α are in $\mathbf{\Lambda}^o_c$. Moreover $Type(v^\tau) = \tau$ and $Type(a^\alpha) = \alpha$.

(b) Suppose $x_1, \ldots, x_n \in \mathbf{\Lambda}^o_c$ and $Type(x_i) = \tau_i$. Then $[x_1 \cdots x_n] \in \mathbf{\Lambda}^o_c$ iff $F(\tau_1, \ldots, \tau_n)$ and $x_i \parallel x_j$ for all $i \neq j$. Moreover

$$Type([x_1 \cdots x_n]) = [Type(x_1) \cdots Type(x_n)].$$

We say that the concrete terms $x_1, \ldots, x_n$ *fit* and write $F(x_1, \ldots, x_n)$, if $[x_1 \cdots x_n]$ is a term.

The letters $x, y, z \ldots$ will range over concrete terms. Clearly, concrete terms intend to represent objects as well as entities resulting from them if we replace any number of their parts by empty places (variables). So fitness makes sense only for this kind of terms and not for the entire $\mathbf{\Lambda}^o$ defined next. The letters t, s, r etc. range over arbitrary terms.

Definition 3.2 The set $\mathbf{\Lambda}^o$ of *terms* and the set $Subtrm(t)$ of *subterms of* t for each t, are defined inductively by the following clauses:

a) $\mathbf{\Lambda}_c^o \subseteq \mathbf{\Lambda}^o$. For each $x \in \mathbf{\Lambda}_c^o$, if $x = [x_1 \cdots x_n]$, then

$$Subtrm(x) = Subtrm(x_1) \cup \cdots \cup Subtrm(x_n) \cup \{x\}.$$

b) For any $t, s \in \mathbf{\Lambda}^o$ such that $Subtrm(t) \cap Subtrm(s) = \emptyset$, $t \odot s \in \mathbf{\Lambda}^o$, and $Subtrm(t \odot s) = Subtrm(t) \cup Subtrm(s) \cup \{t \odot s\}$.

c) For any $t, s \in \mathbf{\Lambda}^o$, $t|s$ is a term and

$$Subtrm(t|s) = Subtrm(t) \cup Subtrm(s) \cup \{t|s\}.$$

d) For every concrete x and any variable v, $\lambda v.x$ is a term, and

$$Subtrm(\lambda v.x) = Subtrm(x) \cup \{\lambda v.x\}.$$

e) For any concrete terms x, y $\bar{\lambda} x.y$ is a term, and

$$Subtrm(\bar{\lambda} x.y) = Subtrm(y) \cup \{\bar{\lambda} x.y\}.$$

Definition 3.3 A variable v occurs *free* in a term t if v is not in the scope of an operator λv. Otherwise occurs *bound.* We denote $FV(t)$ the set of free variables of t.

Non-concrete terms will be called also *ideal.* Note that since, by definition 3.1, every variable v occurs at most *once* in a concrete term, the graphs of its parts are trees again, called *analysis trees.* A minor difference between concrete terms and objects is that every variable occurring in x, *no matter* what its type is, is an *atomic* part. We keep denoting $y < x$, $y \leq x$ and $y <_0 x$ for the facts that y is a proper part, a part and an atomic part of x, respectively.

Two terms are said to be *parallel* and we denote $t \parallel s$ if $Subtrm(t) \cap Subtrm(s) = \emptyset$. Otherwise they are *overlapping.* The set X of terms is *parallel*, notation $\parallel(X)$, if the terms of X are pairwise parallel. As follows from definition 3.2, $t \odot s$ makes sense only if $t \parallel s$. Sometimes we express this fact by saying that $t \odot s$ is “legal”.

IV. Substitution. Substituting a *concrete* term x for the free variable v in t will be denoted

$$t[x/v].$$

This has the meaning of filling the empty place v in t by the entity x. The reverse operation of evacuating a place in t occupied by x is also meaningful and will be denoted

$$t[v/x],$$

where v is a new variable not occurring in t. These operations are subject to the conditions imposed by the construction of concrete terms (see definition 3.1) that concern fitness.

Definition 3.4 (Substitution) 1. For a concrete term y and a variable v, $y[x/v]$ is defined as follows: Let $Type(v) \in \mathbf{T}^o$, $v \leq y$ and

$$y = [[\cdots[[vX_1]X_2]\cdots]X_n]$$

be the analysis of y over v, where X_i are sets of concrete terms. If $Type(x) = Type(v)$ and $[[\cdots[[xX_1]X_2]\cdots]X_n]$ is a concrete term, then

$$y[x/v] = [[\cdots[[xX_1]X_2]\cdots]X_n].$$

Otherwise

$$y[x/v] = y.$$

2. For non-concrete t we have the following clauses:

a) $(t_1 \odot t_2)[x/v] = t_1[x/v] \odot t_2[x/v]$, if $t_1[x/v] \parallel t_2[x/v]$. Otherwise $(t_1 \odot t_2)[x/v] = t_1 \odot t_2$.

b) $(t_1|\cdots|t_n)[x/v] = t_1[x/v]|\cdots|t_n[x/v]$.

c) $(\lambda u.y)[x/v] = \lambda u.y[x/v]$, provided $u \notin FV(x)$. (We express this by saying that x *is free for* v *in* $\lambda u.y$.)

d) $(\bar{\lambda} y.z)[x/v] = \bar{\lambda} y.z[x/v]$.

The *evacuation* $t[v/x]$ of x in t is defined similarly, taking care only that v does not occur in t and $Type(v) = Type(x)$.

In the sequel it is going without saying that in every substitution $t[x/v]$, x is free for v in t.

Equality of terms follows either from syntactic conventions called *syntactic equivalences* or from axioms expressing semantic equivalence. We denote the former relation by "$\equiv$" and the latter by "$=$".

V. Syntactic equivalence. First we adopt the ordinary syntactic conventions aiming to simplify notation. For example the operators λ and $\bar{\lambda}$ are *associated to the left*, that is

$$\lambda v_1 \cdots v_n.x \equiv \lambda \vec{v}.x \equiv \lambda v_1.(\lambda v_2.(\cdots(\lambda v_n.x)\cdots)). \tag{1}$$

and

$$\bar{\lambda} x_1 \cdots x_n.y \equiv \bar{\lambda} \vec{x}.y \equiv \bar{\lambda} x_1.(\bar{\lambda} x_2.(\cdots(\bar{\lambda} x_n.y)\cdots)). \tag{2}$$

Another convention is

$$\bar{\lambda} x.x \equiv \bar{\lambda} x. \tag{3}$$

Further, as an extension of axiom (GO5) we have the following convention: Two concrete terms containing the same subterms are identical. In symbols

$$x \equiv y \Leftrightarrow P_0(x) = P_0(y), \tag{identity}$$

for any concrete x, y.

VI. Axioms for term equality. A basic axiom is β^o-conversion. To state it with sufficient precision let us give the following definition:

Definition 3.5 A term of the form $\lambda v.x \odot \bar{\lambda} y.z$ is said to be a *machine.* The machine $\lambda v.x \odot \bar{\lambda} y.z$ is said to be *active* if (i) $v \leq x$, (ii) $y \leq z$ and (iii) $Type(v) = Type(y)$. Otherwise it is called *inactive.*

Equality Axioms:

$$\lambda v.x \odot \bar{\lambda} y.z = x[y/v] \odot z[v/y], \tag{β^o-conversion}$$

(provided the machine $\lambda v.x \odot \bar{\lambda} y.z$ is active).

$$t \odot s = s \odot t, \tag{$\odot$-commut.}$$

$$t|s = s|t, \tag{$|$-commut.}$$

$$\lambda v.x = x \qquad (\text{if } \ v \notin FV(x)), \tag{void receiver}$$

$$\bar{\lambda} x.y = y, \qquad (\text{if } \ x \not\leq y). \tag{void sender}$$

$$t_1|(t_2|t_3) = (t_1|t_2)|t_3. \tag{$|$-assoc.}$$

$$t|t = t. \qquad (|\text{-idempot.})$$

$$t \odot (s_1|s_2) = (t \odot s_1)|(t \odot s_2). \qquad (\odot\text{-distrib})$$

$$t|v = t \odot v = t \qquad (\text{for every } \ v \notin FV(t)). \qquad (\theta\text{-conversion})$$

Remarks. 1) In order for $\lambda v.x \odot \bar{\lambda} y.z$ to be active, it must, first, be legal, that is, $\lambda v.x \parallel \bar{\lambda} y.z$, hence also $x \parallel z$. From this it follows immediately that $x[y/v] \parallel z[v/y]$, hence $x[y/v] \odot z[v/y]$ is legal too. Therefore β^o-conversion is a transformation that preserves parallely.

2) $\odot$ is not associative. The reason is that parentheses is the only means to denote interaction, so they cannot be dropped as associativity requires. E.g. the terms $(\lambda v.x \odot \lambda u.y) \odot \bar{\lambda} z_1.z$ and $\lambda v.x \odot (\bar{\lambda} z_1.z \odot \lambda u.y)$ are clearly distinct.

Let

$$\lambda^{\mathbf{o}} = \{\beta^o \text{ -conver., void rec., void send., } |\text{-assoc., } |\text{-idempot., } \odot\text{-distr.}\},$$

and

$$\lambda^{\mathbf{o}}\theta = \lambda^{\mathbf{o}} \cup \{\theta\text{−conversion}\}.$$

If $t = s$ is provable in $\lambda^{\mathbf{o}}$ we write $\lambda^{\mathbf{o}} \vdash t = s$. If $t = s$ is provable in $\lambda^{\mathbf{o}}\theta$ we write $\lambda^{\mathbf{o}}\theta \vdash t = s$.

VII. Reduction. The definitions below follow the terminology of [2].

Definition 3.6 A *notion of reduction on* $\mathbf{\Lambda}^o$ is a binary relation $R \subseteq \mathbf{\Lambda}^o \times \mathbf{\Lambda}^o$ such that for any concrete terms t, t',

$$t \equiv t' \Rightarrow (t, t') \in R.$$

Every such R induces the binary relations $\longrightarrow_R$ *(one step R-reduction),* $\leadsto_R$ *(R-reduction)* and $=_R$ *(R-equality)* as follows: $\longrightarrow_R$ is the *compatible closure* of R, i.e.:

1) $(t, t') \in R \Rightarrow t \longrightarrow_R t'$.

2) $t \longrightarrow_R t' \Rightarrow (t \odot s) \longrightarrow_R (t' \odot s)$.

3) $t \longrightarrow_R t' \Rightarrow (t|s) \longrightarrow_R (t'|s)$.

The relation $\leadsto_R$ is the *transitive and reflexive closure of* $\longrightarrow_R$, while $=_R$ is the *equivalence relation* generated by $\leadsto_R$.

Definition 3.7 A relation R is *substitutive* if for any terms t, s, any concrete x and any variable v,

$$(t, s) \in R \Rightarrow (t[x/v], s[x/v]) \in R.$$

Lemma 3.8 *If R is substitutive so are $\longrightarrow_R$, $\leadsto_R$ and $=_R$.*

Proof. By easy induction on the steps of definitions of the relations in question. $\dashv$

Given the notion of reduction R, *R-redexes, R-contracta, R-normal terms* and *R-normal forms* are defined as usual (see [2]).

The notions of reduction we shall be mainly interested here are $\beta^{\mathbf{o}}$ and $\beta^{\mathbf{o}}\theta$. The crucial rule in $\beta^{\mathbf{o}}$-reduction is the transformation of the machine

$$(\lambda v.x \odot \bar{\lambda} y.z),$$

whenever it is active, to

$$x[y/v] \odot z[v/y].$$

Definition 3.9 The relation $\beta^{\mathbf{o}} \subseteq \mathbf{\Lambda}^o \times \mathbf{\Lambda}^o$ consists of the following pairs (for readability we write $t \longrightarrow_\beta s$ instead of $(t, s) \in \beta^{\mathbf{o}}$):

1) If x is concrete, then for every t,

$$x \longrightarrow_\beta t \Leftrightarrow x \equiv t.$$

2) If $\lambda v.x \odot \bar{\lambda} y.z$ is active, then $(\lambda v.x \odot \bar{\lambda} y.z) \longrightarrow_\beta (x[y/v] \odot z[v/y])$.
3) $t_1|(t_2|t_3) \longrightarrow_\beta t_1|t_2|t_3$.
4) $t|t|s \longrightarrow_\beta t|s$.
5) $t \odot (s_1|s_2) \longrightarrow_\beta (t \odot s_1)|(t \odot s_2)$.
6) $(\lambda v.x) \longrightarrow_\beta x$, if $v \not\leq x$.
7) $(\bar{\lambda} x.y) \longrightarrow y$, if $x \not\leq y$.

The relation $\beta^o\theta$ extends β^o containing in addition the pairs
8) $t \odot v \longrightarrow_{\beta\theta} t$ and
9) $t|v \longrightarrow_{\beta\theta} t$,
for any term t and any variable v (provided of course that $t \odot v$ is legal).

Theorem 3.10 *For any two terms $t, s \in \mathbf{\Lambda}^o$,*

$$t =_\beta s \Leftrightarrow \lambda^{\mathbf{o}} \vdash t = s$$

and

$$t =_{\beta\theta} s \Leftrightarrow \lambda\theta^o \vdash t = s.$$

Proof. The proof is easy but tedious. For the $\Rightarrow$-directions we use induction on the definitions of $\rightsquigarrow_\beta$ and $\rightsquigarrow_{\beta\theta}$, while for the $\Leftarrow$-directions we use induction on the length of the proof of $t = s$. $\dashv$

Lemma 3.11 *The relation $\beta^{\mathbf{o}}$, and hence $\longrightarrow_\beta$, $\rightsquigarrow_\beta$, $=_\beta$, are substitutive.*

Proof. We have to check the 7 kinds of pairs contained in $\beta^{\mathbf{o}}$. We just check the β^o-rule the other being trivial. Recall that according to definition 3.4(2.c), $(\lambda u.x)[y/v] = \lambda u.[y/v]$ only if $u \not\leq y$. Assume

$$(\lambda v.x \odot \bar{\lambda} y.z) \longrightarrow_\beta (x[y/v] \odot z[v/y]), \tag{4}$$

(the machine being active), and let us verify that for any variable w and any concrete term p, free for w in the above terms:

$$(\lambda v.x \odot \bar{\lambda} y.z)[p/w] \longrightarrow_\beta (x[y/v] \odot z[v/y])[p/w]. \tag{5}$$

Subcase i. w occurs neither to x nor to z. Then, clearly, the redexes and the contracta in (4) and (5) are identical.

Subcase ii. $w < x$ and $w \not< z$. Since p is free for w in x it follows

$$(\lambda v.x \odot \bar{\lambda} y.z)[p/w] = (\lambda v.x[p/w] \odot \bar{\lambda} y.z).$$

Then, clearly, the last machine is active, therefore

$$(\lambda v.x[p/w] \odot \bar{\lambda} y.z) \longrightarrow_\beta x[p/w, y/v] \odot z[v/y].$$

The other subcases are similar. $\dashv$

We come now to define $\beta^{\mathbf{o}}$ and $\beta^{\mathbf{o}}\theta$- normal forms. For simplicity we say just *normal* instead of $\beta^{\mathbf{o}}$- normal, and θ*-normal* instead of $\beta^{\mathbf{o}}\theta$-normal.

Definition 3.12 A term t is said to be *simple* if it is $|$-free. A simple term is *normal* if it does not contain:
(a) any active machine $(\lambda v.x \odot \bar{\lambda} y.z)$,
(b) any subterm of form $\lambda v.x$ with $v \not\leq x$,
(c) any subterm $\bar{\lambda} x.y$ with $x \not\leq y$.
t is *θ-normal* if in addition it does not contain
(d) any subterm of the form $(s \odot v)$.

A term t is *disjunctive* if $t = (t_1|\cdots|t_n)$ for $n \geq 2$. The disjunctive term $(t_1|\cdots|t_n)$ is *expandable* if at least one of the t_i's is also disjunctive. $(t_1|\cdots|t_n)$ is *contractible* if for some $i, j \leq n$, $t_i \equiv t_j$.

The term t is *normal* (resp. *θ-normal*) if either t is normal simple (resp. θ-normal simple), or $t = (t_1|\cdots|t_n)$, where t_i are normal simple (resp. θ-normal simple) terms and $(t_1|\cdots|t_n)$ is neither expandable nor contractible.

We say that the term t' is a *normal form (nf)* (resp. *θ-normal form (θ-nf)*) of t if t' is a normal term (resp. θ-normal term) and $t \rightsquigarrow_\beta t'$ (resp. $t \rightsquigarrow_{\beta\theta} t'$).

Theorem 3.13 (Existence of nfs) *Every term t has a nf and a θ-nf.*

Proof. It suffices to describe an algorithm for reducing a term t to a normal one t'. The steps of such an algorithm are as follows:

(A) Expand t if t is disjunctive, as well as every disjunctive subterm of t, to a non-expandable disjunctive term (i.e. a maximal disjunctive) using step 3 of definition 3.9 as many times as necessary, and let t_1 be the resulting term.

(B) Contract t_1, if it is disjunctive, as well as every disjunctive subterm of t_1, to a non-contractible term (i.e. a minimal disjunctive) using step 4 of the same definition repeatedly, and let t_2 be the resulting term.

(C) Replace in t_2 every subterm of the form $s \odot (r_1|\cdots|r_n)$ by $(s \odot r_1)|\cdots|(s \odot r_n)$, by the help of step 5 of the aforementioned definition, and repeat until all such subterms are eliminated.

(D) Let $t_3 = (s_1|\cdots|s_m)$ be the term resulting from step (C). It is easy to see that all s_i are simple and t_3 is neither expandable nor contractible. Thus it suffices to normalize each s_i by reducing every active machine they contain and replacing every $\lambda v.x$ such that $v \not\leq x$ by x, and every $\bar{\lambda} y.z$ such that $y \not\leq z$ by z.

If t_4 is the resulting term, clearly, t_4 is normal. In order to get a θ-normal term, it suffices to make one more step:

(E) If $t_4 = (r_1|\cdots|r_m)$, first eliminate every r_i such that $r_i = v$ for some variable v, and second, inside the remaining r_j's replace every subterm $(p\odot v)$ by p.

The resulted term t_5 is θ-normal. $\dashv$

Theorem 3.14 (Uniqueness) *Every term has a unique normal and a unique θ-normal form.*

Proof. We have to show that all normalization algorithms lead to the same normal form. But from the definition of normal forms it is clear that every such algorithm must consist of the steps (A)-(D) or (A)-(E) above. These steps are independent, so two algorithms can differ only in the *order* in which they execute the above steps. Thus one has to verify that the algorithms e.g. ABCD and BCDA when applied to a term t give the same normal output t'. This verification is trivial and tedious and is left to the patient reader. $\dashv$

In classical λ-calculus uniqueness of R-nfs is shown through the Church-Rosser (CR or diamond) property for R: If $t \leadsto_R t_1$ and $t \leadsto_R t_2$, then there exists a term t_3 such that $t_1 \leadsto_R t_3$ and $t_2 \leadsto_R t_3$. The converse is trivially true: Uniqueness of R-nfs implies that R has the CR-property. It implies also the consistency of the calculus.

Corollary 3.15 *(i) For any two terms t, s $\lambda^{\mathbf{o}} \vdash t = s$ (resp. $\lambda^{\mathbf{o}}\theta \vdash t = s$) iff t, s have a common normal (resp. θ-normal) form.*

(ii) The notions of reduction $\beta^{\mathbf{o}}$ and $\beta^{\mathbf{o}}\theta$ are CR.

(iii) The theories $\lambda^{\mathbf{o}}$ and $\lambda^{\mathbf{o}}\theta$ are consistent.

4 Graph-theoretic semantics.

In this section we provide an interpretation of the terms of $\mathbf{\Lambda}^o$ in terms of graphs that renders true the axioms of $\lambda^{\mathbf{o}}$. The graphs in question are defined over *object structures*. An object structure (o.s.) is a quadruple $M = (|M|, [...]^M, F^M, Type^M)$, where $|M|$ is a set whose elements are called "objects", $[...]^M$ is a partial operation from the set $|M|^{<\omega}$ of finite subsets of $|M|$ into $|M|$, F^M is the domain of $[\cdots]$ and $Type^M$ is an equivalence relation on $|M|$, such that M satisfies axioms (GO1)-(GO8). Overlined letters $\bar{x}, \bar{y}, \bar{z}$

range over elements of $|M|$. Parthood, $P(\bar{x})$, $P_a(\bar{x})$ (the sets of parts and atomic parts of $\bar{x}$ respectively), etc. are defined as usual. In particular the letters $\bar{a}, \bar{b}, \bar{c}$, often with subscripts, denote atoms of M. The notations $Type(\bar{x}, \bar{y})$, $Type(\bar{x}) = Type(\bar{y})$ and $\bar{x} \cong \bar{y}$ are equivalent. The letters $\bar{\tau}, \bar{\sigma}$ etc. range over equivalence classes of $Type$. In particular we denote by $\bar{\alpha}, \bar{\beta}, \bar{\gamma}$ equivalence classes of atoms. By (GO8), every class $\bar{\alpha}$ is (countably) infinite, so we can fix enumerations $\bar{\alpha} = \{\bar{a_1}, \bar{a_2}, \ldots\}$, $\bar{\beta} = \{\bar{b_1}, \bar{b_2}, \ldots\}$ for all these types.

Further we require $|M|$ to contain, beside the usual objects, *empty places*. These will be denoted by overlined variables $\bar{v}, \bar{u}, \bar{w}$ and will take part in the formation of other objects. Hence we allow objects to contain empty places among their parts. At syntactic level empty places can be introduced by a new unary predicate V added to the language L of objects, and at semantic level by a set $\bar{V} \subseteq |M|$ added to the structure of M, containing the places $\bar{v}, \bar{u}, \ldots$. Also two additional axioms (V1), (V2) will be added to (GO1)-(GO8). From the point of view of parthood empty places behave like atoms, but their types may be non-atomic. Not only this but we shall assume that for every object $\bar{x}$ there is an abundance of empty places of the type of $\bar{x}$. Thus we add to (GO1)-(GO8) the following principles in the language $L(V)$:

(V1) $$(\forall v \in V)(\forall x)(x \leq v \Rightarrow x = v),$$

and

(V2) $$(\forall x)(\{v \in V : v \cong x\} \quad \text{is infinite}).$$

Henceforth by an *object structure* (o.s.) we shall mean a quintuple

$$M = (|M|, [\cdots]^M, F^M, Type^M, \bar{V})$$

satisfying axioms (GO1)-(GO8), (V1) and (V2).

The fact that $\bar{v} \in \bar{\tau}$ is denoted $\bar{v}^{\bar{\tau}}$, and let $\bar{V}^{\bar{\tau}} = \bar{V} \cap \bar{\tau}$. By axiom (V2) above, each $\bar{V}^{\bar{\tau}}$ is infinite and we can fix enumerations $\bar{V}^{\bar{\tau}} = \{\bar{v}_1^{\bar{\tau}}, \bar{v}_2^{\bar{\tau}}, \ldots\}$ for every class $\bar{\tau}$.

Moreover let

$$P_p(\bar{x}) = P(\bar{x}) \backslash \bar{V} \textit{ and } P_{pa}(\bar{x}) = P_a(\bar{x}) \backslash \bar{V}$$

for the sets of *proper* parts and *proper* atomic parts of $\bar{x}$ respectively. We write $\bar{x} \parallel \bar{y}$ if $P(\bar{x}) \cap P(\bar{y}) = \emptyset$.

Given an o.s. M it is easy to extend F^M and $[\cdots]^M$ over the set of types of M in a natural way:

Definition 4.1 We say that the types $\bar{\tau_1}, \ldots, \bar{\tau_n}$ of M *fit* and write $F^M(\bar{\tau_1}, \ldots, \bar{\tau_n})$ if for some (hence for all) $\bar{x_1} \in \bar{\tau_1}$, ..., for some (hence for all) $\bar{x_n} \in \bar{\tau_n}$ such that $\parallel \{\bar{x_1}, \ldots, \bar{x_n}\}$, $F^M(\bar{x_1}, \ldots, \bar{x_n})$. In such a case we write $[\bar{\tau_1} \cdots \bar{\tau_n}]^M = Type([\bar{x_1} \cdots \bar{x_n}])$.

Recall that the language L^o contains types $\tau, \sigma, \ldots$ structured with respect to F and $[\cdots]$, constants a_i^α,$i \geq 1$, for each atomic type α, and variables v_i^τ, $i \geq 1$ for each type τ. Therefore in order for an object structure M to be an L^o*-structure* it is necessary and sufficient that the following hold:

i) There is an injection

$$\mathbf{T}_a^o \ni \alpha \mapsto \bar{\alpha} \subset |M|$$

from the atomic types of L^o to equivalence classes of $Atom^M$ with respect to $Type^M$. This entails also an injection

$$\mathbf{T}^o \ni \tau \mapsto \bar{\tau} \subset |M|$$

from the set of all types of L^o into classes of $Type^M$.

(ii) For every $\tau_1, \ldots, \tau_n$, $F(\tau_1, \ldots, \tau_n)$ holds inside the language iff $F^M(\bar{\tau_1}, \ldots, \bar{\tau_n})$ holds in M.

(iii) For each particular α, there is an 1-1 correspondence

$$\alpha \ni a_i \mapsto \bar{a}_i \in \bar{\alpha}.$$

(iv) For each type τ there is an 1-1 correspondence

$$V^\tau \ni v_i^\tau \mapsto \bar{v}_i^{\bar{\tau}} \in \bar{V}^{\bar{\tau}}$$

from the variables of type τ to places of type $\bar{\tau}$. Henceforth M denotes an L^o-structure.

Lemma 4.2 *Every L^o-structure M provides a unique interpretation x^M of every concrete term x of L^o, such that:*

(i) $a^M = \bar{a}$, for every constant a, and $v^M = \bar{v}$ for every variable v.

(ii) $([x_1 \cdots x_n])^M = [x_1^M \cdots x_n^M]^M$.

(iii) $Type(x) = \tau$ iff $Type(x^M) = \bar{\tau}$.

In order to interpret also ideal terms we shall extend M to a directed *graph* M^* which contains M as a subset of its nodes. The graph M^* interprets the operations λ, $\bar{\lambda}$, $\odot$ and $|$. For simplicity we denote the corresponding operations in M^* by the same symbols.

M^* will be defined as $M^* = \bigcup_{n\geq 0} M_n$, where M_n will be inductively defined below. To each node $\bar{t}$ of M_n will be assigned a *pointed* finite subgraph $G(\bar{t})$, with point the node labelled by $\bar{t}$. (A directed graph G is *pointed* if there is a unique node a such that for any other node b of G there is a path leading from a to b.)

Let $M_0 = |M|$. For every $\bar{x} \in M$ the pointed graph $G(\bar{x})$ of $\bar{x}$ is just the node $\cdot$ with label $\bar{x}$. We have already seen what $\bar{x} \parallel \bar{y}$ means for $\bar{x}, \bar{y} \in M_0$. M_1 is defined as follows:

For any objects $\bar{x}$, $\bar{y}$, $\bar{z}$ of $M = M_0$ and for each place $\bar{v}$, we introduce new nodes labelled by $\lambda\bar{v}.\bar{x}$ and $\bar{\lambda}\bar{y}.\bar{z}$ and add to M_0 the following new edges:

$$\begin{array}{ccc} \lambda\bar{v}.\bar{x} & \qquad & \bar{\lambda}\bar{y}.\bar{z} \\ \downarrow & & \downarrow \\ G(\bar{x}) & & G(\bar{z}) \end{array}$$

That is, we set

$$M_1 = M_0 \cup \left(\bigcup\{G(\lambda\bar{v}.\bar{x}) : \bar{x} \in M, \bar{v} \in \bar{V}\}\right) \cup \left(\bigcup\{G(\bar{\lambda}\bar{y}.\bar{z}) : \bar{y}, \bar{z} \in M\}\right).$$

Concerning parallely in M_1, let

$$P_a(\lambda\bar{v}.\bar{x}) = P_a(\bar{\lambda}\bar{y}.\bar{x}) = P_a(\bar{x}),$$

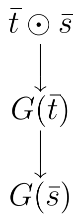
for all $\bar{x}, \bar{y}, \bar{v}$ and let $\bar{t} \parallel \bar{s}$ iff $P_a(\bar{t}) \cap P_a(\bar{s}) = \emptyset$.

Suppose M_n has been defined for $n \geq 1$, suppose also we have defined for each node $\bar{t}$ of M_n its graph $G(\bar{t})$; and suppose we have defined for each $\bar{t} \in M_n$ the set $P_a(\bar{t})$ of *atoms* of $\bar{t}$. Then $\bar{t} \parallel \bar{s}$ means that $P_a(\bar{t}) \cap P_a(\bar{s}) = \emptyset$. Given two graphs $G(\bar{t})$, $G(\bar{s})$ of M_n let the picture

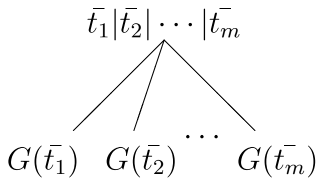
$$\begin{array}{c} G(\bar{t}) \\ \downarrow \\ G(\bar{s}) \end{array}$$

denote the graph produced by driving an arrow from every terminal node of $G(\bar{t})$ to the point $\bar{s}$ of $G(\bar{s})$.

Then for every two nodes $\bar{t}$ and $\bar{s}$ of M_n such that $\bar{t} \parallel \bar{s}$, we introduce a new node labelled by $\bar{t} \odot \bar{s}$ and edges forming the directed pointed graph



with point $\bar{t} \odot \bar{s}$. This graph is just $G(\bar{t} \odot \bar{s})$, i.e. the graph assigned to the new node $\bar{t} \odot \bar{s}$. Similarly for any $m \geq 2$ and any nodes $\bar{t_1}, \cdots, \bar{t_m} \in M_n$, we introduce a new node labelled by $\bar{t_1}|\bar{t_2}| \cdots |\bar{t_m}$ and edges forming the pointed graph



with point $\bar{t_1}| \cdots |\bar{t_m}$. This graph is $G(\bar{t_1}| \cdots |\bar{t_m})$. Note that nodes $\bar{t_1}| \cdots |\bar{t_m}$ are *branching* while $\bar{t} \odot \bar{s}$ are co-linear with $\bar{t}$, $\bar{s}$. Let

$$M_{n+1} = M_n \cup \Big(\bigcup \{ G(\bar{t} \odot \bar{s}) : \bar{t}, \bar{s} \in M_n, \bar{t} \parallel \bar{s} \} \Big) \cup \Big(\bigcup \{ G(\bar{t_1}| \cdots |\bar{t_m}) : \bar{t_i}, \in M_n \} \Big) .$$

Also given the new nodes $(\bar{t} \odot \bar{s})$, $\bar{t_1}| \cdots |\bar{t_m}$ of M_{n+1}, we set

$$P_a(\bar{t} \odot \bar{s}) = P_a(\bar{t}) \cup P_a(\bar{s}), \quad P_a(\bar{t_1}| \cdots |\bar{t_m}) = P_a(\bar{t_1}) \cup \cdots \cup P_a(\bar{t_m}).$$

Two nodes $\bar{t}, \bar{s} \in M_{n+1}$ are said to be *parallel*, notation $\bar{t} \parallel \bar{s}$, if $P_a(\bar{t}) \cap P_a(\bar{s}) = \emptyset$. This finishes the definition of the sequence of M_n. Then set

$$M^* = \bigcup_{n \geq 0} M_n .$$

By some abuse of language we identify each node $\bar{t}$ of M^* with its graph $G(\bar{t})$, so we can refer to the graphs as elements of M^*. Then M^* interprets the terms of L^o in the following way.

Definition 4.3 For any term t, the *M^*-interpretation* t^{M^*} of t is defined inductively as follows:
(a) $t^{M^*} = G(t^M)$ if t is concrete.
(b) $(\lambda v.x)^{M^*} = G(\lambda \bar{v}.x^M)$.
(c) $(\bar{\lambda} x.y)^{M^*} = G(\bar{\lambda} x^M.y^M)$.
(d) $(t \odot s)^{M^*} = G(t^{M^*} \odot s^{M^*})$.
(e) $(t|s)^{M^*} = G(t^{M^*}|s^{M^*})$.

Given a pointed graph G by a *path* of G we shall mean a *maximal* path, i.e., one starting from the point and going down to a terminal node. We let the letters ξ, ζ range over paths. Paths are going to represent *simple terms.* A *normal* path is defined like a normal simple term.

Definition 4.4 Let ξ be a path. ξ is said to be *normal* if it does not contain (a) nodes $\lambda\bar{v}.\bar{x}$ with $\bar{v} \not\leq \bar{x}$, (b) nodes $\bar{\lambda}\bar{y}.\bar{z}$ with $\bar{y} \not\leq \bar{z}$, and (c) notes labelled by active machines $(\lambda\bar{v}.\bar{x} \odot \bar{\lambda}\bar{y}.\bar{z})$. ξ is *θ-normal* if in addition does not contain nodes $(\bar{t} \odot \bar{v})$.

The path ζ is a *normal form of* ξ if it is normal and results from ξ by the obvious normalization procedure, i.e., by (a) identifying nodes $\lambda\bar{v}.\bar{x}$ and $\bar{x}$ if $\bar{v} \not\leq \bar{x}$, (b) identifying nodes $\bar{\lambda}\bar{y}.\bar{z}$ and $\bar{z}$ if $\bar{y} \not\leq \bar{z}$, and (c) replacing the graph having point an active machine $(\lambda\bar{v}.\bar{x} \odot \bar{\lambda}\bar{y}.\bar{z})$ by the graph having point the term $(\bar{x}[\bar{y} \mapsto \bar{v}] \odot \bar{z}[\bar{v} \mapsto \bar{y}])$. Similarly is defined the *θ-normal form of* ξ.

As with terms we easily see that normal forms of paths are unique. So we can define the following equivalences between paths:
$\xi \sim \zeta$, if ξ and ζ have common normal forms,
$\xi \approx \zeta$, if ξ and ζ have common θ-normal forms.
Obviously,

$$\xi \sim \zeta \Rightarrow \xi \approx \zeta$$

but not conversely. Given a graph G, let us write for simplicity $\xi \in G$ for the fact that ξ is a path of G. The equivalences $\sim$ and $\approx$ on paths induce equivalences $\sim^*$ and $\approx^*$ on the graphs of M^*, having the form of "bisimulations", as follows:

Definition 4.5 For $\bar{t}, \bar{s} \in M^*$ let $\bar{t} \sim^* \bar{s}$ iff:

$$[(\forall \xi \in G(\bar{t}))(\exists \zeta \in G(\bar{s}))(\xi \sim \zeta)] \ \& \ [(\forall \xi \in G(\bar{s}))(\exists \zeta \in G(\bar{t}))(\xi \sim \zeta)]\,.$$

Let also $\bar{t} \approx^* \bar{s}$ iff:

$$[(\forall \xi \in G(\bar{t}))(\exists \zeta \in G(\bar{s}))(\xi \approx \zeta)] \ \& \ [(\forall \xi \in G(\bar{s}))(\exists \zeta \in G(\bar{t}))(\xi \approx \zeta)] \, .$$

Clearly

$$\bar{t} \sim^* \bar{s} \Rightarrow \bar{t} \approx^* \bar{s}$$

but not conversely. The structures $(M^*, \sim^*)$ and $(M^*, \approx^*)$ are models of $\lambda^{\mathbf{o}}$ and $\lambda^{\mathbf{o}}\theta$, respectively. To see this let us first establish the following.

Lemma 4.6 *(i) If $t \equiv s$, then $G(t^{M^*}) \sim^* G(s^{M^*})$.*

(ii) For every step X of the algorithm ABCD described in theorem 2 (or a clause X of definition 3.9), if the term s results from t by applying X to t, then $G(t^{M^*}) \sim^* G(s^{M^*})$. Similarly if X is a step of ABCDE, then $G(t^{M^*}) \approx^* G(s^{M^*})$.

(iii) Therefore if $t \rightsquigarrow_\beta s$, then $G(t^{M^*}) \sim^* G(s^{M^*})$ and if $t \rightsquigarrow_{\beta\theta} s$, then $G(t^{M^*}) \approx^* G(s^{M^*})$.

(iv) Conversely, if $G(t^{M^*}) \sim^* G(s^{M^*})$, then t, s have same normal forms.

Proof. (i) It suffices to consider the syntactic equivalences $t \equiv s$ of section 3 and check that for all such $t \equiv s$, the graphs of t^{M^*} and s^{M^*} are $\sim^*$-equivalent. For example it is trivial to check that the graphs interpreting the terms $t \odot s$ and $s \odot t$ have essentially the same paths (essentially means up to $\sim$-equivalence).

(ii) Let us consider the steps A,B,C,D,E. The claim is proved by simply comparing the graphs before and after each reduction step, from the point of view of $\sim^*$-equivalence. Step A (expansion) produces the transform of figure 1.

Figure 1

It is clear that the two graphs have essentially the same paths, therefore they are $\sim^*$-equivalent.

Step B (contraction) produces the transform of figure 2.

Figure 2

The two graphs are again obviously $\sim^*$-equivalent.

Figure 3 shows step C (distribution of $\odot$ over $|$).

Figure 3

Again the paths are essentially the same.

Step D (normalization of simple terms) cannot be depicted by a figure, since the transforms now take place inside the paths. But it obviously preserves $\sim^*$-equivalence by the very definitions: A simple term t is reduced to the normal simple term s iff the path t^{M^*} is reduced to the $\sim$-equivalent normal path s^{M^*}.

(iii) follows from (ii).

(iv) If t, s have distinct normal forms $t_1|\cdots|t_n$ and $s_1|\cdots|s_m$ respectively, then, clearly, as follows from the normalization procedure, at least one of the t_i is distinct from all s_j or vice versa. Since $t_i^{M^*}$, $s_j^{M^*}$ are just $\sim$-equivalent paths of t^{M^*} and s^{M^*} respectively, this means that at least one path of the former is similar to no path of the latter. $\dashv$

Theorem 4.7 (Soundness and Completeness) *Let M be an L^o-o.s. Then for any terms t, s of L^o the following hold:*

(i) $\lambda^{\mathbf{o}} \vdash t = s$ iff $(M^*, \sim^*) \models t = s$ (i.e., $t^{M^*} \sim^* s^{M^*}$).

(ii) $\lambda^{\mathbf{o}}\theta \vdash t = s$ iff $(M^*, \approx^*) \models t = s$ (i.e., $t^{M^*} \approx^* s^{M^*}$).

Proof. (i) Just note that as follows from corollary 1 of the last subsection, $\lambda^{\mathbf{o}} \vdash t = s$ iff t, s have the same normal form r. Thus if $t = s$ is provable and r is their common normal form, then, by the previous lemma we have

$$G(t^{M^*}) \sim^* G(s^{M^*}) \sim^* G(r^{M^*}).$$

The converse follows from (iv) of the previous lemma.

(ii) is similar. $\dashv$

Another pair of equivalences over M^*, broader and, perhaps, more natural than $\sim^*$ and $\approx^*$, are $\sim_1^*$ and $\approx_1^*$ defined as follows:

Definition 4.8 The *resources* of a path ξ is the set $P_a(\xi)$ of all atoms contained in objects occurring in ξ. The *proper resources of* ξ is the set $P_{pa}(\xi)$ of all proper atoms contained in objects occurring in ξ. (Recall that $P_{pa}(\bar{x}) = P_a(\bar{x})\backslash\bar{V}$.) For two paths ξ, ζ let

$$\xi \sim_1 \zeta \quad \text{iff} \quad P_a(\xi) = P_a(\zeta),$$

and

$$\xi \approx_1 \zeta \quad \text{iff} \quad P_{pa}(\xi) = P_{pa}(\zeta).$$

For $\bar{t}, \bar{s} \in M^*$ let $\bar{t} \sim_1^* \bar{s}$ iff:

$$[(\forall \xi \in G(\bar{t}))(\exists \zeta \in G(\bar{s}))(\xi \sim_1 \zeta)] \ \& \ [(\forall \xi \in G(\bar{s}))(\exists \zeta \in G(\bar{t}))(\xi \sim_1 \zeta)],$$

and let $\bar{t} \approx_1^* \bar{s}$ iff:

$$[(\forall \xi \in G(\bar{t}))(\exists \zeta \in G(\bar{s}))(\xi \approx_1 \zeta)] \ \& \ [(\forall \xi \in G(\bar{s}))(\exists \zeta \in G(\bar{t}))(\xi \approx_1 \zeta)].$$

Then, obviously

$$\sim \subseteq \sim_1, \ \approx \subseteq \approx_1, \ \sim^* \subseteq \sim_1^*, \ \approx^* \subseteq \approx_1^* . \tag{6}$$

These relations are reasonable if we see each path of a graph as a "situation" of coexistent entities. Two such situations are "equivalent" if they are formed of the same primitive resources (i.e., atoms, proper or non-proper). The equivalence $\sim_1$ ignores the order in which the operation $\odot$ acts on simple objects, and the operators λ and $\bar{\lambda}$. For example

$$(\lambda \bar{v}.\bar{x} \odot \bar{y}) \odot (\bar{\lambda}\bar{p}.\bar{z}) \sim_1 (\bar{x} \odot \bar{y}) \odot \bar{z} \sim_1 \bar{x} \odot (\bar{y} \odot \bar{z}).$$

Thus $\bar{t} \sim_1^* \bar{s}$ means that the graphs $G(\bar{t})$ and $G(\bar{s})$ contain the same alternative situations. It follows from the relations (6) and theorem 4.7 that $(M^*, \sim_1^*)$ still interprets the axioms of $\lambda^{\mathbf{o}}$, however completeness now fails. That is we have the following:

Theorem 4.9 *Let M be an L^o-o.s. Then for any terms t, s of L^o the following hold:*
(i) If $\lambda^{\mathbf{o}} \vdash t = s$ then $(M^*, \sim_1^*) \models t = s$ (i.e., $t^{M^*} \sim_1^* s^{M^*}$).
(ii) If $\lambda^{\mathbf{o}}\theta \vdash t = s$ then $(M^*, \approx_1^*) \models t = s$ (i.e., $t^{M^*} \approx_1^* s^{M^*}$).

References

[1] M. Abadi and L. Cardelli, A theory of primitive objects: Untyped and first-order systems, *Infor. and Comp.*, vol 125 (1996), 78-102.

[2] H.P. Barendregt, *The Lambda Calculus. Its Syntax and Semantics* (North Holland PC., Amsterdam 1984).

[3] H.P. Barendregt, Lambda Calculi with Types in: S.Abramsky, D. Gabbay, T. Maibaum eds., *Handbook of Logic in Computer Science, Vol 2* (Clarendon Press, Oxford 1992) 117-309.

[4] G. Berry and G. Boudol, The chemical abstract machine, *Theoret. Comp. Sci.* **96** (1992) 217-248.

[5] G. Boudol, Towards a Lambda-Calculus for concurrent and Communicating Systems, in *TAPSOFT 1989*, Lecture Notes in Computer Science, Vol. 351 (Springer, Berlin 1989) 149-161.

[6] R. Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science, Vol. 92 (Springer, Berlin 1980).

[7] A. Tzouvaras, Worlds of homogeneous artifacts, *Notre Dame J. Formal Logic*, **36**(3) (1995) 454-474.

[8] A. Tzouvaras, Significant parts and identity of artifacts, *Notre Dame J. Formal Logic*, **34**(3) (1993) 445-452.

[9] A. Tzouvaras, The Linear Logic of multisets, *Logic. J. of the IGPL*, vol. 6 (1998), pp. 901-916.