*Electronic Notes in Theoretical Computer Science 50 No. 2 (2001) { Proc. BOTH 2001 URL:* [*http://www.elsevier.nl/locate/entcs/volume50.html*](http://www.elsevier.nl/locate/entcs/volume50.html) *34 pages*

*From Bohm's Theorem to Observational Equivalences: an Informal Account 1*

*Mariangiola Dezani-Ciancaglini and Elio Giovannetti Dipartimento di Informatica, Universita degli Studi di Torino*

*corso Svizzera 185, 10149 Torino, Italia, e-mail:* [*fdezani,eliog@di.unito.it*](mailto:fdezani%2Celiog@di.unito.it)

*Abstract*

*There are essentially two ways of looking at the computational behaviours of - terms. One consists in putting the term within a context (possibly of -calculus ex- tensions) and observing some properties (typically termination). The other consists in reducing the term until some meaningful information is obtained: this naturally leads to a tree representation of the information implicitly contained in the original term. The paper is an informal overview of the role played by Bohm's Theorem in these observations of terms.*

*Dedicated to Corrado Bohm*

*on the occasion of the EATCS Distinguished Service Award*

*1 Introduction: from bohming out to observing -terms Bohm's Theorem, in its original formulation [5], states that if M and N are two*

*distinct -normal forms, then there is a context C[ ] such that C[M] ! x*

*and C[N] !*

*y, where x; y are arbitrary distinct variables (the opposite*

*implication being obvious!). If C[ ] is such a context, then a context like* ( xy:C[ ])I , where I x:x and ( x:xx)( x:xx), is reducible to normal form when receiving M, and diverges when receiving N. The theorem can therefore be rephrased as stating that, given two distinct -normal forms, there is a context C[ ] such that C[M] has a normal form (i.e., converges to a value) while C[N] is nonterminating.

*In the same year as Bohm's Theorem (1968), Morris [24] for the rst time* de ned a notion of observational or contextual equivalence, which was going to have such important developments in more recent years, particularly in the domain of interactive concurrent computing: two terms were de ned equiv- alent if, whenever they are put in a same context, either they both make it

*1 Partially supported by MURST Co n '00 AITCFA and MCTAAP II Projects.*

*reducible to a normal form (henceforth occasionally abbreviated to nf ), or* they both make it diverge, i.e.,

*8C[ ]:C[M] has a nf () C[N] has a nf:*

*More generally, two terms (two programs, two processes, two computations,* etc.) may be considered equivalent if, when observed from the outside, they exhibit the same behaviour, i.e., if whenever they are put in a same environ- ment they give rise to the same observations. Of course, for nonterminating computations the equivalence can only be refuted, if at some point they behave di erently from one another, but never veri ed.

*One can de ne di erent contextual equivalences depending on the kind of* context used and the kind of observation performed, and indeed many of them have been introduced over the years.

*Morris' equivalence 2 so corresponds to the natural choice of the pure*

*-calculus itself as an environment, and the context's ordinary conver-* gence/divergence as an observation (of course, in this case the seemingly bi- nary observation actually consists of an in nity of observing acts, and it is itself only semi-decidable). Bohm's Theorem can then be viewed as stating that such an observational equivalence coincides, for normalizable terms, with

*-convertibility.*

*The paramount historical importance of Bohm's Theorem lies however* in the fact, already stressed by the author in the original paper and after- wards pointed out by various researchers, that its proof is constructive, since it consists of an algorithm that, given two distinct -normal forms, builds a discriminating context; an elegant implementation in CAML is given in [14]. As a matter of fact, a more speci c formulation of the theorem is the following: two closed -normal forms M and N are distinct i there exist

*closed terms L1, L2, ..., Ln, with n 0, such that*

*ML1 ::: Lnxy ! x and NL1 ::: Lnxy ! y:*

*The extension to open terms is obvious: two possibly open -normal forms* M and N are distinct i their closures are distinct, therefore i there is a context C[ ] of the form

*( x1 ::: xm:[ ])L1 ::: Lnxy*

*such that C[M] !*

*x and C[N] !*

*y. To construct the separating context,*

*Bohm introduced the so-called Bohm-out technique, based on an analysis of* the term structure that Barendregt [3] later called, extending it to diverging terms, Bohm tree. The starting point is that -normal forms satisfy an induc- tive property, which can be read as an inductive de nition. Recall that a head normal form (henceforth occasionally abbreviated as hnf ) is a term of the

*2 Actually, Morris de ned four di erent equivalences, see [3].*

*form x1 ::: xn:xM1 ::: Mm with n; m 0, where the head variable x is either* free or identical to one of the xi; -normal forms may then be inductively de ned as follows:

*x1x2 ::: xn:x, where n 0, is a (head normal form that also is) a -normal* form;

*a head normal form x1x2 ::: xn:xM1M2 ::: Mm where n 0;m 1, is a* normal form if M1, M2, ..., Mm are -normal forms.

*The Bohm tree of a -normal form merely is the tree representation of this* inductive structure of nested head normal forms.

*De nition 1.1 Bohm trees of -normal forms.*

*(i) BTnf ( x1 ::: xn:x) = x1 ::: xn:x (for n 0);*

*x1 ::: xn:x*

*(ii) BTnf ( x1 ::: xn:xM1 ::: Mm) = (for n 0;m 1)*

*,,,,,,*

*BTnf (M1)*

*¸¸¸¸¸¸*

*BTnf (Mm)*

*The Bohm-out technique consists in building a sequence of suitable terms*

*that, when fed as arguments to two di erent normal forms, brings to the top* (possibly an instance of) a subterm by which these di er. This is achieved by successively binding the head variables to the appropriate selectors so as to go down the path from the root to the desired node. For example, let M and N be the two terms below, represented by the two trees in Fig. 1:

*M = x:xt1( yz:z( u:u)t2)t3 N = x:xt1( yz:z( uv:v)t2)t3:*

*Then the basic discriminating context is [ ]U 3LU 2, which extracts the two*

*1 ,,,, x:x ¸¸¸¸3*

*2 1*

*1 ,,,,, x:x ¸¸¸¸3*

*,,,,,,,, 2*

*¸¸¸¸¸*

*,,,,,,, 2*

*¸¸¸¸¸*

*t1 yz:z*

*..*

*t3 t1 , yz:z t3*

*1 s ss*

*ss*

*u:u*

*¸¸¸2¸¸*

*t2*

*1 .*

*..*

*uv:v*

*¸¸¸2¸¸*

*t2*

*Fig. 1. Two Bohm trees for a simple Bohm-out*

*underlined subterms. The U ns are the selectors 3 of the i-th argument among* n, so that starting from the root the second child among three is selected, and then the rst among two; L is any term, only needed for saturating the abstraction on the non-head variable y. For a discrimination based on the convergence/divergence property, it is suÆcient to append a diverging term

*i*

*, which, when fed as an argument to the two non-matching subterms, makes*

*one of them reduce to normal form and the other diverge; for a discrimina-*

*tion based on two distinct variables it is suÆcient to complete the context as* [ ]U 3LU 2(U 2x)y.

*2 1 1*

*3 by de nition, U n is the term x*

*i*

*1*

*::: xn*

*:xi , see [5].*

*The problem is when along the path from the root to the discriminating* node a same head variable occurs more than once, and di erent children are to be selected at di erent occurrences, like for example in two terms of the form

*x:x( yz:xzt1) and x:x( yz:xzt2), where t1 and t2 are the two non-matching*

*subterms. The solution consists in rst replacing di erent occurrences of the* same variable with di erent bound variables (through the application to suit- able combinators which add abstractions) and then, as in the simple case, replacing each variable with the needed selector.

*Since the observation environment only consists of arguments to be passed* to the observed term (apart from top-level abstractions, in case of open terms) and the Bohm-out technique recursively applies this principle to subterms, the method is obviously unable to discriminate between -convertible forms: the terms M and x:Mx, when applied to an argument, behave the same; this is the reason why the theorem only regards -normal forms.

*Bohm's theorem, through the use of the Bohm-out technique, thus also* established the primigenious and simplest form of the context lemma, which allows quanti cation over all contexts to be replaced, in the de nition of ob- servational equivalences, by quanti cation restricted to head contexts, which are contexts with only one hole occurrence, situated in head position; i.e., contexts of the form ( x1 ::: xn:[ ])M1 ::: Mm, with n; m 0.

*Lemma 1.2 Context Lemma for Normalizable Terms.*

*If M and N are two normalizable terms, 8C[ ]:C[M] has nf () C[N] has nf* i 8CH[ ]:CH[M] has nf () CH[N] has nf, where the CH[ ]'s are head con- texts.

*The reason here is totally obvious: taking the lemma's contrapositive, if there* is a generic context discriminating two normal forms, these cannot be identical, so there must also be a head context that performs the separation (in the other direction, a discriminating head context just is a context!).

*If M and N are closed terms, the quanti cation over contexts may be fur-* ther restricted { as previously observed { to applicative contexts, i.e., contexts of the form [ ]M1 ::: Mm.

*This, in turn, allows observational equivalences to be de ned in a coin-* ductive style (see, for example, [16]) which, though not much meaningful in this case where the equivalence is between normal forms and the calculus is sequential, is however the one used in the study of concurrent and interac- tive systems. Let the notation M =I N indicate the fact that M and N \in isolation behave the same", i.e., that either both M and N reduce to values (that is to nfs, or to hnfs, etc.) or both do not, and let the corresponding equivalence w.r.t. applicative contexts be denoted by the generic symbol ':

*M ' N i , by de nition, 8m 0:8L1 ::: Lm:(ML1 ::: Lm =I NL1 ::: Lm):* Then the following holds:

*M ' N () M =I N and 8L:M L ' NL:*

*In fact, if M ' N, then if we take the empty context we have M =I N, and by* trivially considering the application associativity we have ML ' NL for all L; also the opposite direction obviously holds, hence the property stated above; it may be assumed as an alternative de nition of observational equivalence, which is then usually called an applicative bisimulation or bisimilarity, since it was derived from adapting to -calculus [1] and functional programming the notions of bisimulation and bisimilarity originally introduced for concurrent processes [23,25].

*More precisely, any equivalence ' for which the left-to-right implication* holds is called an applicative bisimulation; the greatest bisimulation, i.e., the one for which the reverse implication also holds, is called (applicative) bisim- ilarity [29].

*The rest of paper is organized as follows. In Sect. 2 we examine the behaviours* of terms within pure -calculus contexts, w.r.t. three di erent choices of what is to be assumed as the set of values. On the other hand, Sect. 3 discusses - calculus extensions allowing to discriminate terms exactly in the same manner as well-known tree representations of terms. We draw some conclusions in Sect. 4.

## *2 Observing pure -terms in pure -calculus*

*If we take two normalizable terms not in normal form, which could represent* two programs still to be run, we might imagine to observe their behaviours by interactively creating, possibly with backtracking, a context that bohms out the subterms being computed. The proof of Bohm's theorem can thus be considered as the rst prototypical example of a refutation procedure for observational equivalence.

*The natural next step consisted in applying the same kind of technique to* obtain a characterization of the observational equivalence for the class of all terms (i.e., also including those without normal forms), as in Morris' de nition. The crucial choice is that of the set of values: we will consider in the following three natural choices, i.e., the sets of normal forms, of head normal

*forms, and of weak head normal forms.*

*2.1 Normal forms as values*

*Since for normalizable terms the observational equivalence amounts to the* coincidence of -normal forms, for generic terms one may expect that it should amount to the coincidence of some kind of generalized, possibly in nite, normal forms. Observe that the above reported inductive de nition of - normal form by means of the head normal form, if read coinductively, becomes the de nition of a notion of possibly in nite -normal form, with a possibly in nite Bohm tree representation.

*The inductive de nition exactly corresponds to the way the normal form*

*is computed by the leftmost-outermost strategy: the term is rst reduced to* its head normal form, then the normal forms of its subterms are recursively computed. The coinductive de nition corresponds to the way the possibly in nite normal form is gradually built by the same strategy in a possibly in nite approximating computation, like an irrational number is built by its successive rational approximations. It is therefore natural to de ne the notions of approximate or partial term [32], and correspondingly of approximate or partial Bohm tree, using the symbol ? for the subterms not yet in head normal form, i.e., the subterms yet to be computed. Like Bohm trees proper, the approximate trees were introduced by [3], with the name of Bohm-like trees.

*De nition 2.1 Approximate Bohm trees.*

*(i) if M = x1 ::: xn:xM1 ::: Mm (with n; m 0),*

*x1 ::: xn:x*

*then ABT(M) =*

*,,,,,,*

*ABT(M1)*

*¸¸¸¸¸¸*

*ABT(Mm)*

*(ii) otherwise (i.e., if M is not in head normal form) ABT(M) = ?.*

*The obvious approximation partial order may be de ned, and the in nite* normal forms obtained by the coinductive reading of 1.1 are the limits of monotone (increasing) sequences of partial terms (approximate trees), which will then be called approximants of the limit.

*The converse does not hold: coinductive in nite normal forms do not con-* tain any occurrences of ?, since this does not occur in the de nition; on the contrary, owing to the phenomenon of unsolvability, i.e., the existence of terms without head normal forms, a computation may generate a sequence of partial terms (partial trees) where some ?-labelled nodes do not expand any further, and cannot therefore be eliminated in the limit. A term that reduces to a term with unsolvable subterms, i.e., to a tree with ?-stuck nodes, does not possess a normal form in the above sense, neither nite nor in nite; if we want to give it a meaning, we are naturally led to complete the space by considering as limits, i.e., as generalized normal forms, also ( nite and in nite) terms (or trees) containing the constant ? as a representation of the unsolvable.

*We will see in the following that ner notions of an in nite normal form* may be introduced; remark, however, that one cannot take the extreme of assuming as de nition of an in nite term (and thus of an in nite normal form) the mere coinductive reading of the ordinary de nition of term, as this would lead { with any reasonable de nition of limit { to the loss of the con uence property for -reduction [18]. Take for example [4] the term HH, with H x:I(xx), which may be thought as resulting { through one step of -reduction { from the application of the xed point combina- tor Y f :( x:f (xx))( x:f (xx)) to the identity: having the property that HH ! I(HH), but also that (in two steps) HH ! , it generates the two

*in nite reduction sequences*

*HH ! ! !*

*HH ! I(HH) ! I(I(HH)) ! I(I(I(HH))) !*

*which, though joinable at every nite step, converge to two di erent limits,* namely and (I(I(I :: :))). With Def. 2.1, on the contrary, all the approx- imants of HH are ?, so the one limit is trivially ?; more generally, the uniqueness of limits is preserved, and with it the uniqueness of meaning (if we adhere to a \syntactic" view of semantics).

*It is therefore natural to assume as de nition of a kind of generalized*

*-normal form, existing for every term, the one emerging from the above* considerations. To keep distinct the newly de ned class of syntactic-semantic objects { possibly in nite and possibly containing the pseudo- nite ? { from the ordinary nite terms, the de nition is formally given in terms of trees: the standard de nition of Bohm trees. Also observe the subtly di erent role played by the symbol ? in Bohm trees, where it denotes the unsolvable, from the one played in the approximants where it denotes the unsolved, i.e., a redex still to be computed; unsolved that keep unsolved forever are unsolvable.

*De nition 2.2 Bohm trees.*

*(i) if M ! x1 ::: xn:xM1 ::: Mm (with n; m 0),*

*x1 ::: xn:x*

*then BT(M) =*

*,,,,,*

*BT(M1)*

*¸¸¸¸¸*

*BT(Mm)*

*(ii) otherwise (i.e., if M does not have a head normal form) BT(M) = ?.*

*The Bohm-out technique may be extended to Bohm trees proper, and Bohm's* theorem, characterizing the observational equivalence restricted to normal- izable terms, may thus be extended, as we anticipated, to characterize the equivalence in the class of all terms.

*We observed that, since the technique is unable to discriminate with re-* spect to applications of the -rule, Bohm's equivalence equates terms possess- ing identical ( nite) -normal forms. Analogously, the unrestricted observa- tional equivalence is unable to discriminate \ -convertible" in nite -normal forms, or Bohm trees; therefore, while two terms having the same Bohm tree are equivalent, the converse does not hold. Clearly, the equivalence corre- sponds to some notion of in nite -normal form, or Bohm -tree.

*To begin with, the nodes of a Bohm -tree must be -head normal forms,* i.e., head normal forms that do not contain -redexes in their top abstractions:

*a term of the form x1 ::: xn 1xn:xM1M2 ::: Mmxn, which may be more ver-*

*bosely but more perspicuously written as x1 ::: xn 1 xn:(xM1M2 ::: Mm)xn,*

*-reduces to x1 ::: xn 1:xM1M2 ::: Mm. Such reduction may in turn recur-* sively give rise to another -redex, if it reduces a rightmost child to the right-

*most variable of its parent node's abstraction, as in the sequence:*

*x1 ::: xn 1xn:xM1M2 ::: Mm( y:xny) ! x1 ::: xn 1xn:xM1M2 ::: Mmxn*

*! x1 ::: xn 1:xM1M2 ::: Mm:*

*When dealing with in nite forms, however, taking -head normal forms is not*

*? y:x*

*y:x*

*y:x*

*y:x*

*...*

*|||*

*¸¸¸¸*

*zzzz*

*¸¸¸¸*

*zzz*

*¸¸¸¸*

*zzz*

*¸¸¸¸*

r*x* $*| y ?*

*y x z y*

*x z y*

*graph of the*

*equation ? x*

*Py = x(Py)*

*?*

*y:x x*

*|||*

*¸¸¸¸*

*x | y x*

*!*

*x x*

*. .*

*Fig. 2. The term y:(Px y)y: approximants, Bohm tree, Bohm -tree.*

*suÆcient [3]; thus, the observational equivalence in the class of all terms does* not simply amount, as in the nite case, to -convertibility. There may be, in fact, in nite -reduction sequences where at every nite step the term is not an -redex, but it becomes an -redex in the limit; the simplest example, following [3,2], is the term y:(Px y)y, where Px is a term (containing the free variable x) such that Px y ! x(Px y) 4 ; it generates the in nite reduction sequence

*y:(Px y)y ! y:x(Px y)y ! y:x(x(Px y))y ! y:x(x(x(Px y)))y ! :::*

*where at every nite step the term of the form y:M y is not an -redex since* M contains an occurrence of y, but it becomes an -redex in the limit, where the y disappears behind an in nite number of x's. Correspondingly, its Bohm tree has an in nite branch generated by the graph of the recursive de nition of Px y, whose all nite approximants do not contain y, as shown in Fig. 2. The -reduced tree is the representation of the in nite term xxx:: :, which coincides with the Bohm tree of any term Qx satisfying the recursive relation

*4 is easily obtained by translating its recursive de nition with the use of the Y combi-*

*nator, i.e., Y( py:x(py)); from which one nally arrives through some reduction steps at Px Ax Ax where Ax zy:x(zzy).*

*P*

*x*

*Qx ! xQx : the simplest is Qx Dx Dx where Dx z:x(zz), represented* in Fig. 3. As a consequence, the two terms y:(Px y)y and Qx , though not -

*?*

|  |  |  |  |
| --- | --- | --- | --- |
| *x* | *x* | *x x* | |
| *?* | *x*  *?* | *x x*  *?* | *x x*  *.* |

*Fig. 3. The term Qx ( z:x(zz))( z:x(zz)): approximants and Bohm tree.*

*(nor -) convertible, cannot be discriminated by observing their e ects on the* context: whatever the sequence of arguments given to their respective closures

*xy:(Px y)y and x:Qx , i.e., whatever the context C[ ] [ ]L1L2 ::: Ln, one* has:

*C[ xy:(Px y)y] ! PL1 L2L2 ::: Ln ! L1(PL1 L2)L2 ::: Ln*

*C[ x:Qx ] ! QL1 L2 ::: Ln ! L1QL1 L2 ::: Ln*

*where PL1 Px [L1=x]; QL1 Qx [L1=x] and moreover*

*PL1 L2 ! L1(PL1 L2); QL1 ! L1QL1 :*

*The two resulting expressions have the same head L1 which initially deter-* mines their behaviours, and the same tail L2 ::: Ln of the argument sequence; they only di er in their rst argument, which in both cases reproduces L1 in head position and is therefore unable to di erentiate the behaviours. For example, for L1 I both contexts diverge, for L1 U n, with 1 < i n, both

*i*

*contexts behave like Li.*

*y:x x*

*zzzz*

*?*

*¸¸¸¸ !*

*y ?*

*Fig. 4. The term y:x( y)y: its Bohm tree and Bohm -tree.*

*Observe that the -reduction cannot be performed on the approximants,* since one does not know whether ? \contains" y or not. However, a term that is not -reducible may happen to have a Bohm tree that is nite but

*-reducible, if the subterm where the \forbidden" variable (i.e., the variable*

*concerned by the -rule) occurs is unsolvable: e.g., the term y:x( y)y of* Fig. 4 obviously does not -reduce to x( y) since this subterm contains y, but its Bohm tree, which in a term-like notation is y:x?y, -reduces to x?, since

*the constant ? has \swallowed" the y in a single node (of course, the fact is* that ? itself is the semantic counterpart of an in nite computation).

*As the above considerations suggest, the notion of Bohm -tree, represent-* ing a kind of in nite -normal form, may then be de ned { again following

*[2] { as the -normal form (T ) of an ordinary Bohm tree T .*

*De nition 2.3 -normal form of a Bohm tree.*

*The -normal form (T ) of a Bohm tree T is de ned by cases as follows:*

*(i) (?) = ?*

*(ii) ( x1 ::: xn:x) = x1 ::: xn:x*

# *>8 0*

*x1 ::: xn 1:x*

*1 if Tm is nite,*

*0 x1 ::: xn:x 1*

*>*

*> @*

*<>*

*T1*

*}}}*

*}*

*}*

*¸¸¸*

*¸*

*¸*

*::: Tm 1*

*(Tm) = xn 6= x*

*and xn 62 FV (Ti)*

*A*

*(iii) @*

*ccccc*

*,,,,, A =*

*for 1 i m 1.*

*T1 ::: Tm*

*>*

*x1 ::: xn:x*

*}}}*

*¸¸*

*:> (T1)*

*}*

*}*

*¸¸*

*¸ otherwise*

*::: (Tm)*

*De nition 2.4 Bohm -trees.*

*The Bohm -tree BT (M) of a term M is de ned as:*

*BT (M) = (BT(M)).*

*The observational equivalence of two terms in Bohm's and Morris' sense then* exactly coincides with the equality of their Bohm -trees, as was proved by Hyland [15] in 1975. We may express the result more formally by introducing an explicit notation for the equivalence, as follows.

*De nition 2.5 Normal observational equivalence.*

*We say that two terms M and N are observationally equivalent w.r.t. to normal* convergence, i.e., when the set of values is the set of normal forms, and we write M 'n N, i

*8C[ ] : C[M] has nf () C[N] has nf.*

*As we will see, this is the nest (i.e., the most discriminating) observational* equivalence that can be obtained with a pure -calculus context and respects Bohm tree equality.

*Theorem 2.6 (Hyland [15]). For any two terms M and N:*

*M 'n N () BT (M) = BT (N)*

*i.e., expressed with the contrapositive:*

*BT (M) 6= BT (N) ()*

*9C[ ] : C[M] has nf ^ C[N] has no nf, or the converse.*

*2.2 Head normal forms as values*

*In the original Morris' paper [24], cited by [3], four di erent observational* equivalences were de ned, and three of them were proved to coincide. In particular, as can be expected, observing whether two closed terms, when put in a context, make it reduce to the same -normal form, does not add any discriminating power w.r.t. the normal equivalence de ned in the previous section.

*Another natural equivalence, introduced by Wadsworth [32] in 1976 along* with the properties described below, consists in limiting oneself to observing whether the context reduces to a head normal form, thus possibly stopping its computation without waiting for a normal form to nally appear.

*De nition 2.7 Head observational equivalence.*

*We say that two terms M and N are observationally equivalent w.r.t. to head* convergence, i.e., when the set of values is the set of head normal forms, and we write M 'h N, i

*8C[ ] : C[M] has hnf () C[N] has hnf.*

*Observe that, owing to the double implication contained in both the equiva-* lence de nitions, the fact that one of them implies the other does not trivially follows by boolean logics. It is however easy to prove that the head equivalence cannot discriminate more than the normal equivalence.

*This relies upon two basic and well-known facts of -calculus. On the* one hand, head divergence is stronger than ordinary divergence: a term (and thus a lled context) not possessing a hnf, a fortiori does not pos- sess a nf; moreover, it also diverges when applied to any sequence of argu- ments, for at every successive reduction step it reduces to a term of the form

*x1 ::: xn:( z:M )NN1 ::: Nm, with n; m 0, where the head position is al-*

*ways occupied by a -redex and never becomes a head variable. But binding a* head variable to an argument is the only means, as already recalled, by which the argument may literally take the lead and a ect the term behaviour.

*On the other hand, a term reducible to a hnf, even if it does not have a nf,* may take an argument and put it in head position, wherefrom it can modify the term itself, e.g., eliminate the divergent subterms.

*Now suppose M and N are two terms separated by the head equivalence,* i.e., there exists a context C[ ] such that, say, C[M] reduces to a hnf while C[N] does not; then it is always possible, owing to the two above recalled properties, to build a context C0[ ] such that C0[M] reduces to a nf while C0[N] does not, thus discriminating w.r.t. the normal equivalence.

*In fact, let C[M] be reducible to a hnf x1 ::: xn:xkM1 ::: Mm, where the* head variable xk is one of the x1 ::: xn (if it is not, one can always add to the context an outermost abstraction to bind it), and where some or all of the Mi may be diverging. Then it is suÆcient to append to the context C[ ] a suitable eraser that, when bound to the head variable, cancels all

*the Mi (if not only the o ending ones), or { which is the same thing { a* normalizing term (say, a variable) preceded by a suitable selector to pick it up: C0[ ] C[ ]L1 ::: Lk 1(U m+1z), where the Li are arbitrary terms needed to saturate the abstractions, and U m+1z is the selector with its argument (at their place the eraser x1 ::: xm:z could have directly been written, saving a

*1*

*1*

*-reduction step).*

*Thus C0[M] ! x1 ::: xn:z, while C0[N] diverges { for the reasons men-*

*tioned { since C[N] has no hnf.*

*The next question is whether the head observational equivalence is strictly* less discriminating than the normal equivalence and, as the answer is positive, whether the equivalence relation it therefore induces on Bohm -trees can be independently characterized in a natural way, for example by the property of having the same normal form, for some new kind of tree normal form.

*Two terms M and N may be observationally distinct w.r.t. to the normal* equivalence (M 6'n N) but undistinguishable w.r.t. to the head equivalence (M 'h N) only if there is context C[ ] such that, say, C[M] has no normal form while C[N] has, and at the same time both C[M] and C[N] have head normal forms (for if they were both without hnf, of course they would both also be non-normalizing).

*We may for simplicity and { as we saw { without loss of generality restrict* ourselves to closed terms, and therefore to applicative contexts. Then M must be a term such that, for any sequence of arguments L1 ::: Lk, either both expressions ML1 ::: Lk and NL1 ::: Lk have hnf or both haven't, but with at least one such sequence having no nf on M, while normalizing on N. This kind of common behaviour of M and N on any arguments requires a sort of - convertibility between them; but for M to be nonterminating, the conversion must consist of an in nite number of steps, namely an in nite -expansion of N. An in nite expansion of the form ::: yhyh+1yh+2 ::: :N : :: yh+2yh+1yh :: :, either obtained from the interior or from the exterior, will not do, since in either case it could only be produced by a recursion in head position, and so by a term without hnf. The expansion can therefore only be performed, at each step, on the newly introduced variable:

*N y0:N y0 y0:N ( y1:y0y1) y0:N ( y1:y0( y2y1:y2)) :::*

*The expanding transformation Q is thus de ned by the recursive relation* Qz ! y:z(Qy), or Q ! zy:z(Qy); its translation by means of the xed point combinator gives Y( qzy:z(qy)) which with the usual steps of -reduction

*nally yields Q RR, where R xzy:z(xxy).*

*Using the notation T x to indicate that the tree T is a nite or in nite* (in the above sense) -expansion of the variable x, and following [3,2] (where the relation is also formally de ned), one may introduce the notion of

*1-normal form of a Bohm tree, or Bohm 1-tree, and state for the head*

*observational equivalence a characterization theorem analogous to 2.6.*

*De nition 2.8 1-normal form of a Bohm tree.*

*The 1-normal form 1(T ) of a Bohm tree T is de ned by cases as follows:*

*(i) 1(?) = ?*

*(ii) 1( x1 ::: xn:x) = x1 ::: xn:x*

# *8> 0*

*x1 ::: xn 1:x*

*1 if Tm xn,*

*0 x1 ::: xn:x 1*

*>*

*@*

*1*

*>>< T1*

*}}}*

*}*

*}*

*¸¸¸*

*¸*

*¸*

*::: Tm 1*

*xn 6= x and*

*xn 62 FV (Ti) for*

*A*

*(iii) 1@*

*¸¸*

*ccccc*

*,,,,, A =*

*1 i m 1.*

*T1 ::: Tm*

*>*

*x1 ::: xn:x*

*>>:*

*}}}*

*}*

*}*

*1(T1)*

*¸¸*

*¸ otherwise*

*::: 1(Tm)*

*Curien [8] shows that the 1-normal form of a Bohm tree can be obtained* by means of a nite number of nite eta-reductions and of a nite number of in nite eta-reductions of variables.

*De nition 2.9 Bohm 1-trees.*

*The Bohm 1-tree BT 1 (M) of a term M is de ned as:*

*BT 1 (M) = 1(BT(M)).*

*!1 z*

|  |  |  |  |
| --- | --- | --- | --- |
| *y:z* | *y0:z* | *y0:z y0:z* | |
| *?* | *y1:y0* | *y1:y0* | *y1:y0* |
|  | *?* | *y2:y1* | *y2:y1* |
|  |  | *?* | *.* |

*Fig. 5. The term Qz: approximants, Bohm tree, Bohm 1-tree.*

*In Fig. 5 are shown the approximants and the resulting in nite Bohm tree of* the term Qz, whose 1-normal form is the variable z. The respective closures

*z:Qz and z:z, are obviously non-equivalent w.r.t. the normal observational* equivalence, for they are trivially discriminated by the empty context (one of them diverges and the other converges). On the other hand, no context can tell them apart if values are head normal forms, as stated in the general form by Wadsworth's result.

*Theorem 2.10 (Wadsworth [32]). For any two terms M and N:*

*M 'h N () BT 1 (M) = BT 1 (N)*

*i.e., expressed with the contrapositive:*

*BT 1 (M) 6= BT 1 (N) ()*

*9C[ ] : C[M] has hnf ^ C[N] has no hnf, or the converse.*

*A short and self-contained proof of the above theorem can be found in [8].*

*2.3 Weak head normal forms as values*

*In order to model the implementation of the actual programming languages,* where program execution does not include program transformation, in par- ticular to model the implementation of functional programming languages, where the evaluation does not evaluate function bodies, a third kind of nor- mal form was introduced in pure -calculus too [26]: the weak head normal form, henceforth often abbreviated as whnf, which is either a variable applied to (possibly zero!) arguments xM1 ::: Mm, or an abstraction x:M. The eval- uation to weak head normal form of a closed term stops as soon as it reaches an abstraction, that is, a functional value.

*Correspondingly, a third kind of observational equivalence may be consid-* ered, where the discriminating context behaviour is whether it reaches a weak head normal form or not.

*De nition 2.11 Weak head observational equivalence.*

*We say that two terms M and N are observationally equivalent w.r.t. to weak* head convergence, i.e., when the set of values is the set of weak head normal forms, and we write M 'w N, i

*8C[ ]:C[M] has whnf () C[N] has whnf.*

*Interestingly, such equivalence turns out to be strictly ner than the normal* equivalence, and thus the nest of the three so far examined. To begin with, one can show that if two terms are distinct w.r.t. to the normal equivalence, so they are w.r.t. the weak head equivalence.

*Let M, N be two such terms, and C[ ] a discriminating context for them;* let therefore M0 C[M] and N0 C[N] be two closed terms such that, say, M0 has a nf, which of course also is a whnf, while N0 has no nf. If N0 has no whnf either, the two terms are distinct w.r.t. to the weak head equivalence too, and no further argument 5 is required.

*The only interesting case is when both M0 and N0 have whnfs respectively*

*x:M0 and x:N0 , with M0 in nf, and N0 diverging. The term N0*

*may be an*

*0 0 0 0 0*

*application not reducing to a whnf, or it may in turn reduce to an abstraction*

*whose body is diverging; depending on whether this recursive structure is nite* or in nite, there are two possibilities. Either a term not having a whnf may be extracted by application to a suÆciently long sequence of free variables z1 ::: zn; these, on the other hand, can be fed to M0 without altering its normalizing behaviour: thus M0z1 ::: zn has whnf and N0z1 ::: zn has not. Or N0z1 ::: zn always reduces to an abstraction for any nite sequence z1 ::: zn,

*5 if the pun is allowed ...*

*and in such case it does so even if some or all of the zi are replaced by diverging* terms Li; on the other hand M0, being in (closed) nf, may be transformed, by application to suitable arguments, into any desired term, in particular into one without whnf; thus a context ( x1 ::: xk:[ ])L1 ::: Ln may be built such that ( x1 ::: xk:M0)L1 ::: Ln has no whnf and ( x1 ::: xk:N0)L1 ::: Ln has. In either case there is thus a context separating M0 from N0 w.r.t. the weak head equivalence.

*Terms always reducing to an abstraction whatever the number of argu-* ments passed to them, like the one hypothesized above, indeed exist. The

*simplest of them is a term such that !*

*z: , known as the ogre,*

*since it eats every successive argument fed to it; its explicit expression is* obtained, as usual, by applying the Y combinator to its recursive de nition:

*Y( yz:y) !*

*( xz:xx)( xz:xx). We may informally write the equality*

*= z1: z2: z3 :: :, whose intuitive meaning will be given a formal description* in Subsect. 3.2.

*The weak head equivalence, which { as we just argued { does not dis-* criminate less than normal equivalence, can immediately be seen to actually discriminate more. It distinguishes { trivially by the empty context { the term

*from x: , which on the contrary are equated by the two previous equiv-* alences because they are both unsolvable (and thus represented by the same Bohm tree ?). In a sense, it makes the di erence between a piece of running code that loops forever without doing anything, and a de ned procedure that does the same but it's not invoked by any other part of the program.

*Moreover, it considers as di erent any two abstractions with a di erent* number of abstracted variables (i.e., any two functions with a di erent number of parameters) even if they are -convertible (it is suÆcient to take a context consisting of a number of arguments equal to the lesser of the two arities, so as to saturate one of two abstractions, but not the other).

*The example of x: and shows that there are pairs of terms with iden-* tical Bohm trees that are separated by the weak head equivalence; however, there also exist equivalent terms having distinct Bohm trees, like M x:xx and N x:x( y:xy). They are equivalent since no applicative context [ ]L1 ::: Ln reduces to whnf on one of them and does not on the other. It is suÆcient to consider the form of L1: if it reduces to an abstraction, then one immediately sees that ML1 and NL1 become -convertible, so they behave the same for any further common sequence of arguments; if on the contrary L1 never reduces to an abstraction, then the reducts of ML1 and NL1, which are respectively L1L1 and L1( y:L1y), either both have whnfs (if L1 has whnf) or they can never become redexes, and thus neither of them can reduce to a whnf.

*Hence the weak head observational equivalence, though ner than the nor-*

*mal equivalence, is orthogonal w.r.t. Bohm tree equality:*

*M 'w N ) M 'n N ) M 'h N* BT(M) = BT(N) ) M 'n N ) M 'h N

*but BT(M)= BT(N) 6) M 'w N and M 'w N 6) BT(M)= BT(N).*

*The issue to be explored, at this point, was whether and how it is possible* to nd some kind of observational equivalence that exactly corresponds to Bohm tree equality.

## *3 Observing pure -terms in extended -calculi*

*In this section we review the various notions of trees which were successively* introduced, beside Bohm trees, to represent the evaluation of terms. A uni ed view is obtained by considering the di erent kinds of trees as corresponding to di erent possible formalizations of the intuitive notion of stable relevant minimal information coming out of a computation (dually, they also naturally induce di erent notions of meaningless term [19]).

*When in a reduction sequence a term reduces to one of the following forms,* the underlined parts will remain stable during the rest (if any) of the com- putation: xM1 ::: Mm, x:M, P @ Q (where @ is the explicit representation, normally omitted, of the operation of application, and P is a term which will never reduce to an abstraction). Having a stable part in a computation, how- ever, does not necessarily mean that we consider it relevant. For instance, we may decide that an abstraction x:M is only relevant when M is of the form

*y1 ::: yn:zN1 ::: Nm (n; m 0): this leads us to the notion of hnf; but other*

*choices of what is to be taken as relevant are possible.*

*As we will see, depending on whether we assume as stable relevant mini-* mal information the head normal form, the weak head normal form, or the top normal form, we respectively obtain Bohm trees, L evy-Longo trees, or Berar- ducci trees. For each of these we will examine -calculus extensions that allow the corresponding version of Bohm's theorem to be established, i.e., the de - nitions of observational equivalences that discriminate terms exactly as trees do.

*3.1 Bohm's theorem for Bohm trees*

*As we saw in the previous sections, it is impossible within the pure -calculus* to build a context that discriminates two -convertible Bohm trees in every case. The reason basically is that, as is well known, in the pure calculus every term can be considered a function and thus applied to an argument; the ability to discriminate between x and its -expansion y:xy may then be obtained by enriching the context calculus with primitive elements that are not functions, such as a numeric constant 0 (obviously not considered as an

*abbreviation of corresponding Church numeral [3]), and consequently a notion* of an error situation which results from the application of a non-function to an argument. Termination with error has therefore to be considered di erent from normal termination and assimilated to divergence, so that x:x, where x may be anything, is discriminated from xy:xy, where x must be a function, by a simple context [ ]0 that passes as argument the non-functional object. These two terms, however, being abstractions of di erent arities, are already separated by the weak head equivalence.

*A variable may have multiple occurrences in a term, some of them in* functional positions, and some not. It is therefore essential, in this approach, to allow di erent occurrences of the same variable to become ultimately bound to di erent terms; to this end, a natural solution is the introduction of an operator of nondeterministic choice, which of course immediately makes the calculus to become non-con uent, where in general only some of the many possible reduction paths will lead to a correct termination. As a consequence, the observed convergence has to be the may-convergence, which means that a term is convergent if it is the starting point of at least one converging reduction sequence, i.e., if at least one of the computations it may generate is terminating with a value.

*Actually, the nondeterministic choice operator gives too much freedom* for replacing variables. We need to control that every time one occurrence of a variable is replaced by a combinator di erent from the expected one, the whole term cannot converge. This control can be realized by adding a standard numerical system [3], i.e., besides the constant 0, three numeric unary functional constants: a constructor s, a destructor p, and a test zero?.

*The above sketched approach is the one adopted by [10], where the nonde-* terministic operator is denoted by the symbol + (not to be confused with the arithmetic operator!). The rules added to the pure calculus (Æ-rules, following the established terminology) are:

|  |  |  |
| --- | --- | --- |
| *p0 ! 0* | *zero? 0 ! T* | *M + N ! M* |
| *p(sn) ! n* | *zero?(sn) ! F* | *M + N ! N* |

*where n stands for s(s ::: (s 0) :: :), and T xy:x; F xy:y.*

*| n t{imzes }*

*The set of values is the set of numerals, the other terms being a sort of Not a*

*Number expressions, and convergence is correspondingly intended as conver-* gence to a numeral; the resulting context equivalence exactly coincides with Bohm tree equality.

*We denote the extended calculus by N+ , and the usual many-step re-*

*duction relation by !*

*N*

*; to simplify the statement of the main property, we*

*introduce a couple of straightforward de nitions.*

*De nition 3.1 May-convergence to a numeral, or N+ -convergence.*

*We say that a term M in the extended calculus may converge to a numeral,* or N+ -converges, and we write M #N+ , if there exists a numeral n such that

*n; we say that M N+ -diverges, and we write M "N+ , if it does not*

*M !*

*N*

*N+-converge.*

*De nition 3.2 N+-observational equivalence. We say that two terms M and N are N+ -observationally equivalent, and we write M 'N N, i*

*8C[ ] 2 N+:C[M] #N+ () C[N] #N+ .*

*Theorem 3.3 (Dezani et al. [10]).* For any two pure terms M and N:

*M 'N N () BT(M) = BT(N)*

*i.e., expressed with the contrapositive:*

*BT(M) 6= BT(N) () 9C[ ] 2 N+:C[M] #N+ ^ C[N] "N+ or the converse.*

*The above equivalence may therefore be considered a kind of \external" op-* erational semantics that is nally sound and complete w.r.t. the semantics consisting of Bohm trees; or, put in the opposite way, an operational seman- tics for which the Bohm tree semantics is fully abstract.

*( x:xx)(s + 0)*

,

*(s + 0)(s + 0)*

¸*,,,,,,*

r

*s(s + 0) ¸*

*¸¸¸¸¸¸*s)

*0(s + 0)*

*uuuu*

*¸¸¸¸¸*s) ¸*,,,,,,*

*¸¸¸¸¸*

*ss u*˛r

r

*s0 0s*

%*00*%

*( x:x( y:xy))(s + 0)*

,

*(s + 0)( y:(s + 0)y)*

*,,,,,,,,*

¸'

*s( y:(s + 0)y)*

*¸¸¸¸¸¸¸¸*z+

*0( y:(s + 0)y)*

¸*,,,,,,*

r

¸'

*¸¸¸¸¸¸¸¸*z+

*,,,,,,,,*

*¸¸¸¸¸¸*s)

*s( y:sy)*

*s( y:0y) 0( y:sy)*

*0( y:0y)*

*Fig. 6. Reduction trees of x:xx and x:x( y:xy) in the context [ ](s + 0)*

*As an example, the Fig. 6 reports the reduction trees of the two terms* obtained by lling the context [ ](s + 0) respectively with the term x:xx and with its -expansion x:x( y:xy). In the rst case there is a path leading to the numeral s 0, while in the second case every computation ends in Not a Number; so the context is able to discriminate, as remarked above, between x, which may be anything, and y:xy which, being a function, cannot be a numeral.

*The two terms have di erent Bohm trees, but being -convertible they can-* not be distinguished by any observational equivalence based on pure -calculus contexts and respecting Bohm tree equality; actually, they are equated by all three equivalences based on pure -calculus contexts: '( = n; h; w).

*An analogous result of an observational equivalence exactly matching the* Bohm tree semantics is obtained in [28,29] through an encoding of the - calculus into the -calculus, followed by the use of an appropriate (e.g., taking divergence into account) bisimulation between processes.

*3.2 L evy-Longo trees vs. Bohm trees*

*As recalled respectively in Subsect. 2.1 and in Subsect. 2.3, besides the ordi-* nary -reduction two other kinds of reduction { and correspondingly of normal form { have been considered in the pure -calculus: the head reduction, with the associate notion of head normal form, and the weak head reduction, or lazy reduction, with the associate notion of weak head normal form.

*Referring to a usual formal presentation of -calculus, with contextual* rules written explicitly:

*( ) ( x:M)N ! M[N=x]*

*( ) x:Mx ! M if x 62 FV (M)* ( ) M ! N ) LM ! LN

*( ) M ! N ) ML ! NL*

*( ) M ! N ) x:M ! x:N*

*the ordinary reduction, also generically called -reduction, is the one induced* by the rules ; ; ; ; by excluding the rule , one obtains the head reduction, induced by the rules ; ; ; nally, the weak head reduction is obtained by further excluding the rule , i.e., it is the one induced by ; . For each reduction relation there is the corresponding notion of normal form, which is a term where none of the rules of the respective set applies. By adding the -rule to the ordinary and head reduction one respectively obtains the

*-reduction and -head reduction.*

*It is well known that in the case of ordinary and head reduction more* than one rule may in general apply to a given term, and therefore di erent reduction strategies can be de ned, not all of them guaranteed to lead to the respective normal forms whenever they exist (actually, head reduction is usually intended to be associated with the normalizing outermost strategy); the weak head reduction, on the other hand, is completely deterministic.

*The normalizing leftmost outermost strategy for ordinary reduction, sim-* ply called normal reduction in the following, can be recursively de ned, as reminded in Subsect. 2.1, by means of the head reduction. It also admits a recursive de nition using the weak head reduction: to normally reduce a term,

*reduce it by weak head reduction; if this terminates, then either the resulting* whnf is a variable, in which case the normal reduction also terminates, or it has the form x:M or xM1 ::: Mm, in which cases recursively weak head reduce the subterms, respectively M or M1,... ,Mm.

*This corresponds, of course, to another possible way of inductively de ning* the notion of -normal form:

*a variable x is a (weak head normal form that also is) a normal form;*

*a weak head normal form x:M is a normal form if M is a normal form;*

*a weak head normal form xM1 ::: Mm is a normal form if M1, ..., Mm are* normal forms.

*The L evy-Longo tree [22,21,20] of a -normal form is the tree represen-* tation of this inductive structure of nested weak head normal forms, like the Bohm tree was for nested head normal forms. Here too the above de nition, if read coinductively, de nes the notion of a generalized, possibly in nite, - normal form, existing for every term. Approximate L evy-Longo trees are the analogous of the approximate Bohm trees, and so are the notions { w.r.t. their Bohm-tree homologous { of partial order and limit.

*De nition 3.4 Approximate L evy-Longo trees.*

*(i) if M = x, then ALT(M) = x*

*(ii) if M = x:M, then ALT(M) =*

*x ALT(M)*

*ppppx ¸¸¸¸¸¸*

*(iii) if M = xM1 ::: Mm (m > 0), ALT(M) =*

*pp*

*ALT(M1)*

*¸*

*ALT(Mm)*

*(iv) otherwise (i.e., if M is not in whnf) ALT(M) = ?. De nition 3.5 L evy-Longo trees.*

*(i) if M ! x, then LT(M) = x*

*(ii) if M ! x:M, then LT(M) =*

*x LT(M)*

*.... x ¸¸¸¸¸*

*(iii) if M ! xM1 ::: Mm (m > 0), LT(M) =*

*..*

*LT(M1)*

*¸*

*LT(Mm)*

*(iv) otherwise (i.e., if M does not have a whnf) LT(M) = ?.*

*Observe that, owing to the recursive de nition of the leftmost outermost strat-* egy through head or weak head reduction, when the computation reaches a hnf or respectively a whnf, the successive reduction, only acting on subterms, does not change the term's top-level form; or, in the language of trees, it does not change the root node.

*A term reduction is therefore, as already pointed out in Subsect. 2.1, the* step-by-step construction of a possibly in nite structure, through the accu- mulation of stable atoms { i.e., not subject to future change { of relevant information. Depending on which syntactical structure is assumed as the atomic (or minimal) relevant information, descriptions of di erent granulari- ties are obtained for the computed result, as anticipated at the beginning of the section.

*In the representation based on head normal forms, the atom of rele-* vant information is the whole underlined part in x1 ::: xn:xM1 ::: Mm (with m; n 0). In the representation based on weak head normal forms, this atom is further split into smaller separate components x1,..., xn, x, since the minimal relevant information is the underlined part in x:M or xM1 ::: Mm.

*x*

*}}x ,,,*

*}}}}*

*v*

*,,*

*y w*

*x:x*

*z1*

*ssss*

*¸¸¸¸¸*

*z2*

*z v ss*

*yz:z w ?*

*}}z ,,,*

*u:u*

*sssss*

*¸¸¸*

*¸*

*¸*

*x*

*z3*

*.*

*}}} ,, .*

*u x*

*u*

*Fig. 7. L evy-Longo and Bohm trees of the term x:xv( yz:z( u:u)x)w and of the term ( xz:xx)( xz:xx)*

*L evy-Longo trees are thus ner than Bohm trees, in the sense that there* is a homomorphic node mapping from the L evy-Longo tree to the Bohm tree of the same term. For example, in the left part of Fig. 7 are shown the L evy-Longo tree and the Bohm tree of the normal form x:xv( yz:z( u:u)x)w (instance of one considered in Sect. 1). However, the fact that the relevant information labelling one Bohm tree node generally happens to be distributed over several nodes in the corresponding L evy-Longo tree is a mere super cial syntactic appearance; the actual di erence between the two structures lies in the unsolvable terms, i.e., in that a ?-labelled node in a Bohm tree may correspond to a non-?-subtree in its L evy-Longo correspondent. For example, in Fig. 7 is also shown an in nite L evy-Longo tree corresponding to a nite Bohm tree: the term introduced in Subsect. 2.3, which has no hnf, whose Bohm tree is therefore simply ?.

*As a consequence, two di erent L evy-Longo trees corresponding to identi-* cal Bohm trees may only di er in an unsolvable node, represented by a Bohm tree ?-node.

*3.3 Bohm's theorem for L evy-Longo trees*

*As we saw in the preceding subsection, L evy-Longo tree equality strictly im-* plies Bohm tree equality, and therefore it strictly implies head and normal observational equivalences. It is also strictly ner than weak head observa- tional equivalence: in the rst place, it cannot be less discriminating, roughly since if M 6'w N, that is, say, C[M] has whnf while C[N] has not, then M and N must di er in some homologous subterms, which cannot be both without whnf; therefore LT(M) 6= LT(N) (for a rigorous proof see [1]). Secondly,

*x:xx and x:x( y:xy) are an example of two terms having dif-*

*ferent L evy-Longo trees which are equated by the weak head equivalence.* Summarizing, we have:

*LT(M)= LT(N) ) BT(M)= BT(N)*

*LT(M)= LT(N) ) M 'w N ) M 'n N ) M 'h N:*

*Remark however that the two terms and , which have di erent L evy-Longo* trees but identical Bohm trees, are already discriminated by the weak head equivalence, while and, which have di erent L evy-Longo trees but are observationally equivalent, have di erent Bohm trees.

*As a matter of fact, if two terms have di erent L evy-Longo trees but* identical Bohm trees, they must di er { as remarked above { in an unsolvable node (like in the trees of and x: or , where the di erence is directly in the root), and are therefore separated by the weak head equivalence; hence, one has in general:

*LT(M) = LT(N) () BT(M) = BT(N) and M 'w N:*

*The exact discrimination of L evy-Longo trees cannot thus be achieved by a* purely observational equivalence in the pure -calculus; put in other words, an operational semantics for pure -calculus that is sound and complete w.r.t. the L evy-Longo tree semantics cannot be de ned observationally within the calculus itself.

*Such an equivalence is obtained by Sangiorgi in [27,29] by rst adopting* the Milner encoding of the lazy -calculus into the -calculus, let it be denoted by ( ), and then considering within the -calculus a standard observational equivalence between processes, as the weak bisimilarity, or a barbed congru- ence, let it be generically denoted by the symbol ': the result is proved that

*(M) ' (N) i LT(M) = LT(N).*

*Observational equivalences sound and complete w.r.t. the L evy-Longo tree* equality have been de ned outside the -calculus by resorting to various stan- dard (or less standard) extensions of -calculus. In [27] the operator + of nondeterministic choice, already presented in Subsect. 3.1, is added to the calculus, obtaining the set of terms + and the reduction relation ! +: the usual applicative bisimilarity (between closed terms) is then considered.

*( x:xx)( K)*

,

*K( K)*

r

*( x:x( y:xy))( K)*

,

*K( y: Ky)*

*K( K)*

,

¸*,,,,,,*

*¸¸¸¸¸¸*s)

*( K)*

,

*,,,,,,,,,*

*K( y: Ky)*

¸*,* '

,

*¸¸¸¸¸¸¸¸*

*( y:* s*K*\**y)*

,

*v: K*

*. vy: Ky .*

*Fig. 8. Reduction trees of and applied to the argument K*

*De nition 3.6 +-observational equivalence*

*We say that two closed term M and N are +-observationally equivalent, or* applicatively bisimilar in the +-calculus, and we write M '+ N, i :

*(i) M has whnf () N has whnf ;*

*(ii) for all closed L 2 + one has ML '+ NL;*

*(iii) if M !*

*+*

*if N !*

*+*

*M0, there exists an N0 such that N !*

*N0, there exists an M0 such that M !*

*+*

*+*

*N0 and M0 '+ N0,* M0 and M0 '+ N0.

*Observe that the third clause, not present in the observational equivalence def-* initions we previously introduced, is necessary because of the non-con uence of the extended calculus; it is a typical feature of bisimilarity proper, as de ned in concurrent (and thus non-con uent) calculi, in contrast with the ordinary contextual equivalences in -calculi.

*The operator + is only needed in expressions of the form M + , to produce* a nondeterministic branching between a possibly converging term and the diverging ; hence a restricted unary form suÆces, denoted , with the rules:

*M ! M M ! :*

*More formally, if we denote by ' the applicative bisimilarity in, de ned* analogously to Def. 3.6, we have

*M '+ N () M ' N:*

*The main result of [27] is then:*

*Theorem 3.7 (Sangiorgi [27]).* For any two pure terms M and N:

*M ' N () LT(M) = LT(N).*

*For example, we can discriminate and by applying them to K, with* K uv:u. Fig. 8 shows the reduction trees: now, if obeying to the third clause of Def. 3.6 we apply the two corresponding (underlined) terms to an

*arbitrary argument L we get:*

*( v: K)L*

,

*pp K ¸¸¸*

*( vy: Ky)L*

,

*y: Ky*

*pp*

*pp*

*K* ¸r

*¸¸¸*&&

,

*.*

*where obviously and the value y: Ky are not bisimilar, hence neither* are and in the rst place.

*Another way of obtaining a Bohm's theorem for L evy-Longo trees is the one* of Boudol and Laneve [6], who introduce a \resource-conscious" re nement of -calculus in which every argument comes with a multiplicity. More pre- cisely, the argument of a -redex may happen to be available only a nite number of times, in contrast with its always in nite availability in the ordi- nary -calculus. In such calculus of multiplicities [6] the -rule is consequently modi ed as follows:

*( ) ( x:M)Nm ! M < Nm=x >*

*where < Nm=x > is the explicit substitution that can replace at most m* occurrences of x in M by N. The ordinary -rule is recovered by putting m = 1.

*A term of the form x < N0=x > is a deadlock, since we are required to* replace an occurrence of x with the term N that is not available. For example, the reduction paths of and when applied to I1 are:

*( x:xx)I1 ! (xx) < I1=x >! (Ix) < I0=x >! z < x1=z >< I0=x >*

*! x < x1=z >< I0=x >*

*( x:x( y:xy))I1 ! (x( y:xy)) < I1=x >! (I( y:xy)) < I0=x >*

*! z < ( y:xy)1=z >< I0=x >! ( y:xy) < ( y:xy)1=z >< I0=x >;*

*where ! is the reduction relation induced by the rule ( ).*

*Let denote the set of terms with multiplicities: a contextual equivalence* is then obtained by choosing as set of values the set of abstractions (abs).

*De nition 3.8-observational equivalence. We say that two terms M and N are-observationally equivalent, and we write M ' N, i*

*8C[ ] 2: C[M] !*

*an abs () C[N] !*

*an abs:*

*Theorem 3.9 (Boudol and Laneve [6]).* For any two pure terms M and N:

*M ' N () LT(M) = LT(N)*

*i.e., expressed with the contrapositive:*

*LT(M) 6= LT(N) ()*

*9C[ ] 2 : C[M] !*

*an abs ^ C[N] 6!*

*an abs or the converse.*

*For example 6', sinceI1 reduces to an abstraction, while this is false* for I1.

*( x:xx)(P + Q) #*

*(P + Q)(P + Q)*

*. &*

*P (P + Q) Q(P + Q)*

*. & #*

*( x:x( y:xy))(P + Q) #*

*(P + Q)( y:(P + Q)y) #*

*P ( y:(P + Q)y) #*

*PP PQ*

*# #*

*P (P O) Q(QO) # #*

*P (O(OO)) x:x #*

*P I #*

*I(IO)*

*# IO #*

*O*

*(a)*

*x:x*

*( y:(P + Q)y)(( y:(P + Q)y)O) #*

*(P + Q)(( y:(P + Q)y)O) #*

*P (( y:(P + Q)y)O) #*

*P ((P + Q)O) #*

*P (QO) #*

*P ( x:x ) #*

*( x:x )(( x:x )O) #*

*( x:x )O #*

*O #*

*#*

*.*

*(b)*

*Fig. 9. (a) The reduction tree of (P + Q). (b) An in nite reduction path out of*

*(P + Q).*

*Finally, Dezani et al. [12] consider the behaviour of pure terms within contexts* of the concurrent -calculus de ned in [9]. This calculus is obtained from the pure -calculus (with call-by-value and call-by-name variables) by adding the nondeterministic choice operator + and a parallel operator k, whose main

*reduction rule is*

*M ! M0 N ! N0*

*MkN ! M0kN0*

*where ! stands for one-step reduction.*

*(jj)*

*Let ! +k be the so obtained reduction relation. We compare terms by* taking as values call-by-value variables, abstractions, and parallel composi- tions of an arbitrary term with a value, the last being quite natural in view

*of the above rule for k. If we denote call-by-value variables by Greek letters* we obtain the following grammar for the set V of values:

*V ::= j x:M j :M j V kM j MkV* where M is any term.

*We consider must-convergence, which means that a term is convergent if*

*all reduction sequences starting from it reach a value.*

*De nition 3.10 +k-convergence.*

*We say that a term M in the extended calculus +k-converges, and we write* M #+k, if there is no in nite reduction path out of M and moreover whenever

*M !*

*+k*

*+k*

*N there exists a value V such that N !*

*V ; we say that M*

*+k-diverges, and we write M "+k, if it does not +k-converge.*

*De nition 3.11 +k-observational equivalence. We say that two terms M and N are +k-observationally equivalent, and we write M '+k N, i*

*8C[ ] 2 +k:C[M] #+k() C[N] #+k.*

*Theorem 3.12 (Dezani et al. [12]).* For any two pure terms M and N:

*M '+k N () LT(M) = LT(N)*

*i.e., expressed with the contrapositive:*

*LT(M) 6= LT(N) () 9C[ ] 2 +k:C[M] #+k ^ C[N] "+k or the converse.*

*Fig. 9 shows that 6'+k by applying them to the term P + Q, where* P : ( O), O zt:t, and Q yx:x .

*3.4 Bohm's theorem for Berarducci Trees*

*Bohm trees and L evy-Longo trees may be viewed as two particular kinds* of syntax trees of possibly in nite normal forms, where some parts of the syntactic structure are hidden as non-relevant: in particular, following the concrete syntax of -calculus, the binary application operator is left implicit. The third kind of trees representing possibly in nite normal forms is ob- tained by directly starting from abstract syntax trees of terms, where we

*explicitly represent application with the symbol @. De nition 3.13 Syntax trees.*

*(i) ST(x) = x;*

*(ii) ST( x:M) =*

*x ST(M)*

*sss @ ¸¸¸¸*

*(iii) ST(MN) =*

*ss*

*ST(M)*

*¸*

*ST(N)*

*For example the syntax trees of the terms x:f (xx), x:xx and x:xxx are* shown in Fig. 10, and Fig. 11 gives the syntax trees of the self-applications of these terms.

*x*

*x*

*cc@ ,,*

*x*

*c@ ,,,*

*ccc ,,*

*c@ ,,*

*ccc ,,*

*f c@ ,*

*cccc ,,,*

*@ ,, x*

*cccc*

*x*

*,,, x*

*x*

*x eeee*

*x*

*,,,*

*x*

*Fig. 10. The syntax trees of x:f (xx), x:xx and x:xxx*

*,,,,,,@ ¸¸¸¸¸*

*,,,,,,,*

*¸¸¸*

*...@ ¸¸¸¸*

*x*

*cc@ ,,*

*x*

*cc@ ,,*

*.....*

*x*

*¸¸¸¸*

*x*

*ccc ,,*

*ccc ,,*

*c@ ,,*

*c@ ,,*

*f c@ ,*

*f c@ ,*

*cccc ,,,*

*cccc ,,,*

*cccc*

*x*

*,,,*

*x*

*cccc*

*x*

*,,, x x x x*

*x*

*....@ ¸¸¸¸¸¸¸*

*....*

*x*

*¸¸¸¸¸¸*

*x*

*c@ ,,, c@ ,,,*

*ccc ,,*

*e@ ,,, x*

*ccc ,,*

*e@ ,,, x*

*eee ,,*

*x x*

*eee ,,*

*x x*

*Fig. 11. The syntax trees of ( x:f (xx))( x:f (xx)), ( x:xx)( x:xx) and ( x:xxx)( x:xxx)*

*Now if we reduce these terms we get:*

*( x:f (xx))( x:f (xx)) ! f (( x:f (xx))( x:f (xx))) ! f (f (:: :)) ! ::: ( x:xx)( x:xx) ! ( x:xx)( x:xx) ! ( x:xx)( x:xx) ! :::*

*( x:xxx)( x:xxx) ! ( x:xxx)( x:xxx)( x:xxx) ! :::*

*so the trees of Fig. 11 are not informative of their behaviours. As a matter* of fact ( x:f (xx))( x:f (xx)) generates an in nite number of applications in which the terms in function position are always f , ( x:xxx)( x:xxx) generates

*an in nite number of applications in which the terms in argument position are* always x:xxx, while ( x:xx)( x:xx) always reduces to itself.

*In analogy with the previously presented kinds of trees, consider for every* term the possibly in nite sequences of syntax trees associated to reduction sequences starting from that term; for a notion of limit to be de ned, the application operator must also be taken into account { beside abstraction and variables { as an atom of relevant information, whose occurrences become part of the limit tree when they are stable.

*,,,, @ ¸¸¸¸¸¸¸*

*e@ ,,*

*,,,,,,*

*e@ ,,,*

*¸¸¸¸¸¸*

*x*

*eee ,,*

*e ,,*

*f @*

*,*

*,*

*ccc ,,*

*. e x*

*?*

*@*

*cc ,,,*

*f c @*

*eee ,,*

*c@ ,,,*

*ccc ,,*

*e@ ,,, x*

*ee ,*

*f .*

*eeee*

*x*

*ccc ,,*

*@ ,, x*

*,*

*,,*

*x*

*eee ,,*

*x x*

*Fig. 12. The Berarducci trees of ( x:f (xx))( x:f (xx)), ( x:xx)( x:xx) and ( x:xxx)( x:xxx)*

*The question is then: when is an application occurrence stable? Given a* term MN ( M@N), we have two cases: if M reduces to a -abstraction, i.e., M ! x:M0, then a step of -reduction of course destroys the occurrence of the application operator, and the tree of MN must be the tree of M0[N=x]; on the contrary, if M does not reduce to an abstraction, the application is

*pppp@ ¸¸¸¸¸*

*stable and the tree of MN is*

*pp*

*tree of (M)*

*¸ : This leads to the*

*tree of (N)*

*key notions of zero term and of top normal form, and through them to the* de nition of Berarducci trees, introduced in [4] and independently in [17] at the same time.

*De nition 3.14 Zero Terms and Top Normal Forms.*

*(i) A term is called a zero term i it cannot -reduce to an abstraction.*

*(ii) A term is called a top normal form (tnf) if it is a variable, or an abstrac-* tion, or an application of the form MN, where M is a zero term.

*It is easy to verify that zero terms are either unsolvable terms of order zero [1],* like and ( x:xxx)( x:xxx), or they are reducible to terms of the form xM1 ::: Mn where n 0, i.e., to applications of a free variable to any number (also zero!) of arguments.

*Examples of top normal forms are ( x:xxx)( x:xxx)( x:xxx) and x: .*

*Alternatively one can de ne the top normal forms as the normal forms w.r.t.* the top reduction, i.e., the reduction induced by the rules ( ) and ( t), where:

*( t) M ! N ) ML ! NL (provided M is not a zero term):*

*If we observe terms by putting them within pure -calculus contexts and* taking as values tnfs, we obtain a corresponding contextual equivalence.

*De nition 3.15 Top observational equivalence.*

*We say that two terms M and N are observationally equivalent w.r.t. to top* convergence, i.e., when the set of values is the set of top normal forms, and we write M 't N, i

*8C[ ]:C[M] has tnf () C[N] has tnf.*

*This last equivalence is not comparable with the weak head equivalence 'w:* on the one hand it distinguishes from ( x:xxx)( x:xxx) which are equated by 'w; on the other hand it equates x: I and I which are separated by 'w (the rst of the two terms has whnf, the second has not; both are tnfs, and moreover, by de nition, they still have tnfs to whatever sequence of arguments they might be applied).

*From the above it is natural to de ne the Berarducci trees of terms as the* trees we can draw as soon as we reach a top normal form.

*De nition 3.16 Berarducci trees.*

*(i) if M ! x, then BeT(M) = x*

*(ii) if M ! x:N, then BeT(M) =*

*x*

*BeT(N)*

*(iii) if M ! M1M2, where M1 is a zero term, then*

*.... @ ¸¸¸¸¸*

*BeT(M) =*

*..*

*BeT(M1)*

*¸*

*BeT(M2)*

*(iv) otherwise (i.e. if M does not have a tnf) BeT(M) = ?.*

*Berarducci trees are more discriminating than L evy-Longo trees, and hence* than Bohm trees, since for example ( x:xx)( x:xx) and ( x:xxx)( x:xxx) have di erent Berarducci trees (shown in Fig. 12), while they have identical L evy-Longo trees and Bohm trees, namely ?.

*Berarducci tree equality is also at least as discriminating as the top ob-* servational equivalence 't, as follows from an argument analogous to the one used for L evy-Longo trees and weak head equivalence in Subsect. 3.3 (for a proof see [11]).

*Observing terms within pure -calculus will therefore equate some terms* having di erent Berarducci trees, whatever kind of normal forms is chosen as the set of values. To separate terms having di erent Berarducci trees we need

*to bohm out also arguments of unsolvable terms, as for example in case we* want to nd a context which discriminates II and I, whose Berarducci trees are represented in Fig. 13. To this purpose the paper [11] considers the

*c@ ¸¸¸*

*ccc ¸*

*c@ ,,*

*cccc*

*?*

*@ ,, x*

*,*

*,*

*x x* x

*cccc*

*?*

*ccc ,,*

*@ ,, x*

*,*

*,*

*? x*

*Fig. 13. Berarducci trees of II and I*

*set of terms OA obtained from the pure -calculus by adding the two constants* O and A. The constants O and A select the operator and the argument of a closed, stable application. These constants have the following reduction rules:

*O(MN) ! M if MN is a closed term and M is a zero term* A(MN) ! N if MN is a closed term and M is a zero term:

*For instance, II and I are discriminated by the context A(O[ ]). In fact,* if we denote by !OA the induced reduction relation, we have:

*A(O( II)) !OA A( I) !OA I*

*A(O( I)) !OA A( ) !OA :*

*With the Bohm-out technique employed in the calculi presented so far, one* had to solve (either by using suitable combinators, or by means of the nonde- terministic choice) the problem of replacing di erent occurrences of the same variable by di erent selectors; such problem disappears in this last -calculus extension, because the selection is performed by the two constants O and A.

*De nition 3.17 OA-observational equivalence. We say that two terms M and N are OA-observationally equivalent, and we write M 'OA N, i*

*8C[ ] 2 OA : C[M] has a tnf () C[N] has a tnf.*

*Theorem 3.18 (Dezani et al. [11]).* For any two pure terms M and N:

*M 'OA N () BeT(M) = BeT(N)*

*i.e., expressed with the contrapositive:*

*BeT(M) 6= BeT(N) ()*

*9C[ ] 2 OA : C[M] has tnf ^ C[N] has no tnf or the converse.*

## *4 Conclusion*

*In Fig. 14 are summarized the observational equivalences and tree represen-* tations discussed in the present paper. An arrow between two points means that the starting point induces a -theory ner than the arriving point: for example the arrow between 'w and 'n says that M 'w N implies M 'n N. Similarly for double arrows. We want to remark that all the proofs of the horizontal arrows use (some variant of) the Bohm-out technique.

*BT 1 () 'h*

*\* \**

*BT () 'n*

*\* \**

*'N () BT 'w*

*\* \**

*'+k () ' () ' () LT 't*

*\* \**

*'OA () BeT*

*Fig. 14. Relations between observational equivalences and equalities of trees*

*It should be clear that the present overview leaves many questions unan-* swered or in wait for more satisfactory answers. In particular, all the various

*-calculus extensions need to be better justi ed, in the sense that it must be* determined how they actually depend on the structure of the kinds of trees respectively addressed.

*To conclude, it is worthwhile to mention that there exist precise correspon-* dences between the tree representations of terms and the local structures (or equivalently the -theories) of certain -models ([3], Chapter 19). In partic- ular, such correspondences amount to the fact that two terms have the same tree representation if and only if they are equal in the -model. For example,

*the Bohm 1-trees represent the local structure of Scott's D1 model as* de ned in [30] (this result was proved in [32]);

*the Bohm -trees represent the local structure of the inverse limit model* de ned in [7];

*the Bohm trees represent the local structure of Scott's P! model as de ned* in [31] (a discussion on this topic can be found in [3], Chapter 19);

*the L evy-Longo trees were introduced by Longo in [22] (following [21]), to* prove that they represent the local structure of Engeler's models as de ned in [13].

## *Acknowledgement*

*We are very grateful to Pierre-Louis Curien, Roger Hindley, Fer-Jan de Vries* and the anonymous referees for their useful remarks and suggestions.

## *References*

*[1] S. Abramsky and C.-H. L. Ong. Full abstraction in the lazy lambda calculus. Inform. and Comput., 105(2):159{267, 1993.*

*[2] F. Barbanera, M. Dezani-Ciancaglini, and F.-J. de Vries. Types for trees. In Proceedings of PROCOMET'98, pages 11{29. Chapman & Hall, London, 1998.*

*[3] H. P. Barendregt. The lambda calculus. Its syntax and semantics. North- Holland Publishing Co., Amsterdam, revised edition, 1984.*

*[4] A. Berarducci. In nite -calculus and non-sensible models. In Logic and algebra (Pontignano, 1994), pages 339{377. Dekker, New York, 1996.*

*[5] C. Bohm. Alcune proprieta delle forme - -normali nel -K-calcolo.*

*Pubblicazioni dell'IAC, 696:1{19, 1968.*

*[6] G. Boudol and C. Laneve. The discriminating power of multiplicities in the*

*-calculus. Inform. and Comput., 126(1):83{102, 1996.*

*[7] M. Coppo, M. Dezani-Ciancaglini, and M. Zacchi. Type theories, normal forms, and D1-lambda-models. Inform. and Comput., 72(2):85{116, 1987.*

*[8] P.L. Curien. Sur l'eta-expansion in nie. Comptes Rendus de l'Acad emie des Sciences, to appear, 2001.*

*[9] M. Dezani-Ciancaglini, U. de'Liguoro, and A. Piperno. A lter model for concurrent -calculus. SIAM J. Comput., 27(5):1376{1419 (electronic), 1998.*

*[10] M. Dezani-Ciancaglini, B. Intrigila, and M. Venturini-Zilli. Bohm's theorem for Bohm trees. In ICTCS'98 (Prato, 1998), pages 1{23. World Scienti c, Oxford, 1998.*

*[11] M. Dezani-Ciancaglini, P. Severi, and F.-J. de Vries. Bohm's theorem for Berarducci trees. In CATS'00 (Canberra, 2000), volume 31(1) of Electronic Notes in Theoretical Computer Science, pages 143{166. Elsevier, 2000.*

*[12] M. Dezani-Ciancaglini, J. Tiuryn, and P. Urzyczyn. Discrimination by parallel observers: the algorithm. Inform. and Comput., 150(2):153{186, 1999.*

*[13] E. Engeler. Algebras and combinators. Algebra Universalis, 13(3):389{392, 1981.*

*[14] G. Huet. An analysis of Bohm's theorem. Theoret. Comput. Sci., 121:145{167, 1993.*

*[15] M. Hyland. A syntactic characterization of the equality in some models for the lambda calculus. J. London Math. Soc. (2), 12(3):361{370, 1975/76.*

*[16] B. Jacobs and J. Rutten. A tutorial on (co)algebras and (co)induction. Bulletin of EATCS, 62:222{259, 1997.*

*[17] R. Kennaway, J. W. Klop, R. Sleep, and F.-J. de Vries. In nite lambda calculus and Bohm models. In Rewriting Techniques and Applications, pages 257{270. Lecture Notes in Comput. Sci. 914. Springer-Verlag, 1995.*

*[18] R. Kennaway, J. W. Klop, R. Sleep, and F.-J. de Vries. In nitary lambda calculus. Theoretical Computer Science, 175(1):93{125, 1997.*

*[19] R. Kennaway, V. van Oostrom, and F.-J. de Vries. Meaningless terms in rewriting. J. Funct. Logic Programming, Article 1:35 pp, 1999. (electronic)* [*http://www.cs.tu-berlin.de/journal/j p/articles/1999/A99-01/A99-01.h*](http://www.cs.tu-berlin.de/journal/j)*tml.*

*[20] J.-J. L evy. An algebraic interpretation of the K-calculus and a labellel - calculus. In -Calculus and Computer Science Theory, pages 147{165. Lecture Notes in Comput. Sci. 37. Springer-Verlag, 1975.*

*[21] J.-J. L evy. An algebraic interpretation of the K-calculus, and an application of a labelled -calculus. Theoret. Comput. Sci., 2(1):97{114, 1976.*

*[22] G. Longo. Set-theoretical models of -calculus: theories, expansions, isomorphisms. Ann. Pure Appl. Logic, 24(2):153{188, 1983.*

*[23] R. Milner. Communication and Concurrency. Prentice Hall, 1989.*

*[24] J.-H. Morris. Lambda calculus models of programming languages. PhD thesis, M.I.T., 1968.*

*[25] D. Park. Concurrency and automata on in nite sequences. In 5th GI Conference on Theoretical Computer Science, pages 167{183. Lecture Notes in Comput. Sci. 184. Springer Verlag, 1981.*

*[26] S. L. Peyton Jones. The Implementation of Functional Programming Languages. Prentice Hall, 1987.*

*[27] D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. Inform. and Comput., 111(1):120{153, 1994.*

*[28] D. Sangiorgi. L evy-Longo Trees and Bohm Trees from encodings of calculus*

*into -calculus. Notes, 1995.*

*[29] D. Sangiorgi. Interpreting functions as pi-calculus processes: a tutorial. Revised version of TR RR-3470, INRIA Sophia Antipolis. Available as ftp://ftp- sop/meije/theorie-par/davides/functionPItutorial.ps.gz, 1999.*

*[30] D. Scott. Continuous lattices. In Toposes, algebraic geometry and logic (Conf., Dalhousie Univ., Halifax, N. S., 1971), pages 97{136. Lecture Notes in Math., Vol. 274. Springer, Berlin, 1972.*

*[31] D. Scott. Data types as lattices. SIAM J. Comput., 5(3):522{587, 1976.*

*Semantics and correctness of programs.*

*[32] C. P. Wadsworth. The relation between computational and denotational properties for Scott's D1-models of the lambda-calculus. SIAM J. Comput., 5(3):488{521, 1976.*