Definability and full abstraction

Pierre-Louis Curien[1]

PPS, CNRS and University Paris 7

Abstract

Game semantics has renewed denotational semantics. It offers among other things an attractive classification of programming features, and has brought a bunch of new definability results. In parallel, in the denotational semantics of proof theory, several full completeness results have been shown since the early nineties. In this note, we review the relation between definability and full abstraction, and we put a few old and recent results of this kind in perspective.

Keywords: operational semantics, denotational semantics, sequentiality.

1 Introduction

In the semantics of programming languages, full abstraction studies started with the two papers on PCF by Robin Milner [44] and Gordon Plotkin [49], respectively (see also [50]). Milner showed the uniqueness (up to isomorphism) of the fully abstract (cpo) model of PCF, and constructed it as a suitable quotient of syntax. Plotkin showed that the continuous model of PCF (the only one available at the time) is not fully abstract, but he showed that it becomes fully abstract for an extension of PCF with "parallel or" (actually, originally, "parallel if", see [18] for the more natural variant with parallel or). From there on, two choices were open for investigations on full abstraction:

- vary the language to fit an intended model,
- vary the model to fit an intended language.

Plotkin's result was a result of the first kind. It was followed by a result of Berry and Curien [10], who showed that their model of sequential algorithms [9] (developed with the motivation of solving the full abstraction problem for the original language PCF) is fully abstract for the (kernel CDS01 of the) language CDS – a

[1] Email: curien@pps.jussieu.fr

functional language based on Kahn and Plotkin's concrete data structures (originally called "information matrices") [33]. Later, in [13] (see also [12,19]), the same model was shown to be fully abstract for a more "standard" language called SPCF (an extension of PCF with a control operator called `catch`). At the end of my "Thèse d'Etat", I showed a few counterexamples that witnessed the gap separating a model based on sequential algorithms from full abstraction with respect to the original language PCF [17].

The whole subject restarted with the advent of game semantics, and in the light of linear logic, which taught us in the meantime about the decomposition $A \to B = !A \multimap B$: Such a decomposition for sequential algorithms (and an associated reading of the model in terms of games) was found [37,20]. Shortly after, the two teams now known as HO (Martin Hyland and Luke Ong, to whom one should add the name of Hanno Nickau [45] who worked out the same model independently) and AJM (Samson Abramsky, Radha Jagadeesan and Pasquale Malacaria) came up with syntax independent descriptions of the term model of PCF (based on appropriate Böhm trees, which I have proposed to call PCF Böhm trees [7]). They found two models of games and strategies, similar in spirit to the model of sequential algorithms (but with different underlying constructions of the "!" operation) for which the interpretation of PCF Böhm trees was both injective and surjective. Surjectivity was by far the most interesting property, and it is best understood as the question of characterizing the image of the interpretation. Both teams thus showed very important *definability* results.

Both results came after a similar surjectivity property was obtained by the same teams in the realm of the semantics of (multiplicative linear logic) proofs, under the name of *full completeness* [3,31]. Since then, a number of full completeness, or *denotational completeness* (a terminology introduced by Girard [27]) results for the semantics of various fragments of linear logic in various models have appeared. But let us come back to the semantics of PCF and related languages.

The HO characterization, which proved to be more flexible, was the following: every innocent and well-bracketed strategy is the (unique) image of a PCF Böhm tree by the interpretation. It was then natural to ask what happens if one removes either of these conditions. It turned out that innocent (and not necessarily well-bracketed) strategies characterize the image of the interpretation of programs in an extension of PCF with control [34], while well-bracketed (not necessarily innocent) strategies characterize the image of the interpretation of a version of Idealized Algol [5]. (Roughly, this version of Idealized Algol is "PCF + first-order references".) Moreover, the latter result yields a full abstraction result (almost) without quotient.

In this short note, we recast the relation between definability and full abstraction, and examine the status of older and more recent results with respect to these two related notions. Morally, once definability holds, a fully abstract model can be constructed via a quotient that mimics the operational equivalence at the semantic level. Of course, such full abstraction results are somewhat "cheap" (while the definability results that guide the construction of the model are in general all but easy), but in some cases, fortunately, one does not need a quotient, when the model

has enough separating capabilities. This happened in the following three instances:

- when the model is extensional (Plotkin's result for the continuous model and PCF+ `por`);
- for the model of sequential algorithms, with respect to CDS01 (Berry-Curien), and to PCF+`catch` (Cartwright-Curien-Felleisen);
- for the model of well-bracketed strategies with respect to (a variant of) Idealized Algol (Abramsky-McCusker).

We explain this in more detail in the sequel.

2 Full abstraction from definable separation

The gap between definability and full abstraction is governed by a very simple sufficient condition, which we give below. It is the very purpose of this paper to stress the elementary fact that these properties are not the same. Somehow, the profusion of various terminologies such as full completeness or denotational completeness (cf. above), or intensional full abstraction [4], contributed to obscure the difference.

I do not need to enter in the specificities of PCF or of its various extensions. It will be enough to fix some typed (sequential) applicative (i.e. admitting function types and function application) language L, together with a collection of observable types and values, and an operational semantics for closed programs of observable type. One writes $M \to^* v$ to denote the fact that the evaluation of M terminates with an observable value v.

The models interpret types and terms as objects and morphisms of a category. An element d of type A of the model (that is, a morphism from 1 to $[\![A]\!]$) is *definable* if there is a closed term M of type A such that $[\![M]\!] = d$.

Most models satisfy the so-called *computational adequacy* property, which states that, for all closed terms M of observable type, $M \to^* v$ if and only if $[\![M]\!] = v$. In particular, observable values receive their "standard" interpretation $[\![v]\!] = v$ – this notation meaning also that every element of the model at an observable type is definable.

Thus, there is perfect match between the syntax and the semantics *at observable types*. Computational adequacy is usually proved by a variant of the computability/realisability method.

The property of *full abstraction* allows us to formulate a correspondence between syntax and semantics for arbitrary terms, not necessarily closed nor of observable type. Its formulation relies on the notion of contextual (operational) equivalence, which is the following:

$$M =_{op} N \quad \Leftrightarrow \quad (\forall C \;\; C[M] \to^* v \;\Leftrightarrow\; C[N] \to^* v) \;,$$

where C ranges over contexts (terms with a hole) such that both $C[M]$ (C in which the whole is filled with M) and $C[N]$ are closed and of observable type. Then the

model is called *fully abstract* with respect to L if for all M, N (of the same type):

$$[\![M]\!] = [\![N]\!] \quad \Leftrightarrow \quad M =_{op} N \ .$$

The left-to-right direction is called *adequacy* (or "abstraction") and is an easy consequence of computational adequacy. The other direction is the hard one, and corresponds to the *full* of full abstraction or the *complète* of "complète adéquation" (the French term used for full abstraction). A nice explanation of why this is the hard side of the equivalence comes from the theory of intersection types [15] and of "domains in logical form" [2], which recasts the denotational side of the correspondence as the side of "proofs", and the operational side as the side of "models". Then, under these glasses, fullness/completeness goes in the right direction: if $\models M = N$ (i.e., if $M =_{op} N$), then $\vdash M = N$ (i.e., $[\![M]\!] = [\![N]\!]$).

Now we can state the criterion around which this note is organized. We assume for simplicity that there is only one observable base type o, but of course the same holds in presence of several observable types (for example, PCF has two observable types: `nat` and `bool`).

Criterion 2.1 *If the model is such that for every distinct (definable) f, g of the same type A there exists a* definable *h of type $A \to o$ such that $hf \neq hg$, then it is fully abstract.*

Proof. If $[\![M]\!] \neq [\![N]\!]$, let h be given by our assumption, and let P be such that $[\![P]\!] = h$, $v_1 = h([\![M]\!])$, $v_2 = h([\![N]\!])$, and $C = P\,[\,]$. Then $C[M] \to^* v_1$ and $C[M] \to^* v_2$, and hence $M \neq_{op} N$. □

I propose to call this sufficient property for full abstraction the *definable separability* property. To be best of my knowledge, the first proofs of full abstraction based on definable separability were:

- (in an untyped setting) the one relating the model D_∞ of the untyped lambda-calculus and the notion of head normal form [30,53] (see [21] for a concise and "interactive" account), and
- in the present typed setting, the proof of full abstraction of the model of sequential algorithms with respect to PCF+`catch` [13].

Criterion 2.2 *Under the assumptions that the type hierarchy is the simple type hierarchy and that the model is enriched over algebraic complete partial orders, a sufficient condition for definable separability (and hence for full abstraction) is the conjunction of the following two properties of the model:*

(i) compact definability*: all compact elements at all types are definable, and*

(ii) extensionality*: the elements of the model at function types are functions (in category-theoretical terms, the model has enough points).*

Proof. We may write an arbitrary type A as $A_1 \to \ldots \to A_n \to o$. Let $f \neq g$. By extensionality and algebraicity, there exist compact $d_1, \ldots, d_n$ such that $f d_1 \ldots d_n \neq$

$gd_1 \dots d_n$. Let P_i be such that $d_i = [\![P_i]\!]$ $(1 \leq i \leq n)$. Let $h = [\![\lambda x.xP_1 \dots P_n]\!]$. Then $hf \neq hg$. □

Of course, one can prove directly (ie. without going through definable separability) that an extensional model enjoying the definability property is fully abstract (in the proof above, instead of defining h, one directly defines the context $[\,]\,P_1 \dots P_n$). This is how Plotkin proved that the continuous model is fully abstract for PCF+`por`.

Examples of models matching the first criterion and not the second are, again, D_∞ (which does not satisfy compact definability) and the model of sequential algorithms (which is not extensional).

Let us make here a short digression. In [49], a further extension of PCF is also presented, such that every *computable* element of the model at all types becomes definable – a property which Plotkin called *universality*.

Coming back to compact definability, Milner proved a converse to the second criterion above: the fully abstract cpo model of PCF *has to be* (and hence is characterized by the property of being) order-extensional and to enjoy the compact definability property. (A model is called order-extensional when not only the elements of function types are functions, but also their ordering is the pointwise ordering.) The long quest for a fully abstract model started from there. First, Berry defined the stable model [8], which is extensional but not order-extensional. Then Berry and myself constructed the model of sequential algorithms, which is not extensional (although it can be made extensional by enlarging the set of observable values with one error element). More importantly, in these models, compact definability with respect to PCF fails:

- A famous counterexample witnessing the failure of compact stable functions to be all definable is the Gustave, or Berry-Kleene function (see, e.g. [7]).
- The counter-examples to compact definability for the model of sequential algorithms in [18] turned out to be influential in the genesis of the AJM game model.

The works on game semantics departed quite radically from this line of work, by renouncing extensionality more decisively. The HO and AJM game semantics for PCF (restricted to the type hierarchy of PCF) amount to syntax-free descriptions of PCF Böhm trees. The game models thus satisfy a fortiori the compact definability property (because the bijection between strategies and PCF Böhm trees restricts to a bijection between finite (= compact) strategies and finite PCF Böhm trees, and because the latter are essentially PCF terms in some canonical form). However, the HO and AJM models fail to be extensional, and they also fail to satisfy the definable separability property.

But there is an "abstract non sense" machinery that forces the definable separability property. One equates any f and g of type A (for all A interpreting a type of the syntax) which cannot be separated by an h, i.e. we set $f =_{op} g$ whenever $hf = hg$ for all h of type $A \to o$. The only delicate point is in verifying that the quotient is still a model (and can in particular interpret the fixpoint construction of PCF).

We stress here that this abstract construction does not exploit any feature of the game model, and as a matter of fact can be applied as well to the model of PCF Böhm trees (which is isomorphic to the HO and AJM models on the finite type hierarchy), yielding a version of Milner's original construction of the fully abstract model of PCF (which was also a quotient of syntax, defined in a slightly different way).

In other words, despite their title, the key contribution of each of the seminal HO and AJM papers is a *definability* result. In some sense, it is the best one could hope for, since it was shown by Loader [39] that the operational equivalence for PCF is not effective (and undecidable for finitary PCF, the version of PCF where the only basic type is the finite type of booleans). This negative result closed the search for a "direct", unquotiented construction of the fully abstract model of PCF, since if such a construction existed, it would surely lead to an effective presentation.

Laird's model of control of innocent (and not necessarily well-bracketed) strategies also enjoys definability, but not definable separability, for the same reason as the original HO model: the candidate h for separating f and g is a strategy which is not innocent in general. So, his model needs quotienting, and as a matter of fact Laird has shown how to obtain the model of sequential algorithms as a quotient of his model [36].

Once innocence is removed, then (a variant of) definable separability holds. A variant is needed, as the candidate h is in general not well-bracketed either, but when f and g differ at least in one of their well-bracketed plays, then a well-bracketed h can be found. In the well-bracketed model of Idealized Algol, it holds that $M =_{op} N$ if and only if $[\![M]\!]$ and $[\![N]\!]$ have the same well-bracketed (complete, in the terminology of [5]) plays.

The following table summarizes the definability and full abstraction results discussed in this section.

Language	Model	Definability	Full abstraction
PCF + `por`	*Cont*	*Yes*	*Yes*
PCF + `catch`	$SA \approx (\mathbf{G}_{inn}/=_{op})$	*Yes*	*Yes*
PCF	$PCFBT/=_{op}$	*Yes*	*Yes*
PCF	$\mathbf{G}_{AJM}, \mathbf{G}_{HO}, PCFBT$	*Yes*	*No*
PCF + *control*	$\mathbf{G}_{inn}$	*Yes*	*No*
Idealized Algol	$\mathbf{G}_{wb}$	*Yes*	*Yes*

where *Cont*, *SA*, *PCFBT*, $\mathbf{G}_{AJM}$, $\mathbf{G}_{HO}$, $\mathbf{G}_{inn}$, and $\mathbf{G}_{wb}$ stand for the continuous model, the model of sequential algorithms, the PCF Böhm tree model, the AJM model, the HO model, the model of innocent strategies, and the model of well-bracketed strategies, respectively. (Recall that the three models *PCFBT*, $\mathbf{G}_{AJM}$, $\mathbf{G}_{HO}$ are isomorphic when restricted to the finite type hierarchy.)

3 On the notion of observables

In the table concluding the last section, all languages considered share the same notion of observable. One could also (not exhaustively) add two other full abstraction results to the list:

- Longley's model of sequentially realizable functionals, or, equivalently, Bucciarelli and Ehrhard's model of strongly stable functions [11] – both these models coincide with the extensional collapse of the model of sequential algorithms – is fully abstract for an extension of PCF with a certain operator H [40].
- Paolini has shown that the stable model is fully abstract for an extension of PCF with two operators, one of which is some Gustave like operator [48].

More generally, when varying the language, one could also think of varying the observable types and values. We mention here only one example, close enough to PCF, and however sharply differing in its properties. Unary PCF is the variant of PCF where the only basic type has only one (non bottom) element, and in which the conditional construct of PCF "degenerates" to a conjunction `if` M `then` N. Loader has shown that the operational equivalence for this language is decidable [38], in sharp constract to the (even finitary) PCF case. As a matter of fact, Laird's model of bistable functions is fully abstract for unary PCF [35].

4 Concluding remarks

The full abstraction problem for PCF has triggered a lot of work, and is to be celebrated for its "side effects".

- The stable model (reinvented by Girard) gave rise to linear logic [26], whose enormous influence in our community should be recounted elsewhere.
- The model of sequential algorithms led me to take cartesian closed categories seriously as a syntax, and from there to the categorical abstract machine [16], and then to explicit substitutions [1].
- The full abstraction problem boosted also the study of logical relations [51,52,46], and motivated extensional accounts of sequentiality (cf. section 3).
- Game semantics proved remarkably flexible. Beyond control and first-order references (discussed above), game semantical accounts of a variety of programming features have been given, including subtyping [14], non-determinism [28], probabilistic choice [22], higher-order references [6], call-by-value [29], or concurrency [25].

- Malacaria and Hankin applied game semantics to program analysis [41,42,43]. In a similar vein, applications of game semantics to effective proofs of program equivalences and inequivalences, and more generally to abstract interpretation and model-checking are being developed [24,47,23] under the name of *algorithmic game semantics*.

Let us mention finally that full abstraction is also of pervasive importance in the theory of process calculi. There, the notion of operational equivalence is bisimulation under many variants, and the notion of model is given by a target calculus: one thus studies fully abstract *translations* of a language into another, in a sense faithful to the original definition of full abstraction.

References

[1] Abadi, M., L. Cardelli, P.-L. Curien, and J.-J. Lévy, *Explicit substitutions*, Journal of Functional Programming **1**(4) (1992), 375-416.

[2] Abramsky, S., *Domain theory in logical form*, Annals of Pure and Applied Logic **51** (1991), 1-77.

[3] Abramsky, S., and R. Jagadeesan, *Games and full completeness for multiplicative linear logic*, Journal of Symbolic Logic, **59**(2) (1994), 543–574 (conference version in Proc. FOSSACS'92).

[4] Abramsky, S., R. Jagadeesan, and P. Malacaria, *Games and full abstraction for PCF*, Information and Computation **163**(2) (2000), 409-470 (preliminary conference version in Proc. Theoretical Aspects of Computer Science 1994, Springer LNCS 789, 1-16).

[5] Abramsky, S., and G. McCusker, *Linearity, sharing and state: a fully abstract game semantics for Idealized Algol with active expressions*, in O'Hearn and Tennent (eds.), Algol-like languages, Birkhauser (1997).

[6] Abramsky, S., K. Honda, and G. McCusker, *A fully abstract game semantics for general references*, in Proc. Thirteenth Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press, 334-344 (1998).

[7] Amadio, R., and P.-L. Curien, "Domains and Lambda-Calculi", Cambridge University Press (1998).

[8] Berry, G., *Stable models of typed lambda-calculi*, Proc. ICALP, Springer Lect. Notes in Comp. Sci. 62 (1978).

[9] Berry, G., and P.-L. Curien, *Sequential algorithms on concrete data structures*, Theoretical Computer Science **20** (1982), 265-321.

[10] Berry, G., and P.-L. Curien, *The kernel of the applicative language CDS: theory and practice*, Proc. French-US Seminar on the Applications of Algebra to Language Definition and Compilation, Cambridge University Press (1985), 35-87.

[11] Bucciarelli, A., and T. Ehrhard, *Sequentiality in an extensional framework*, Information and Computation **110**(2) (1994), 265-296.

[12] Cartwright R., and M. Felleisen, *Observable sequentiality and full abstraction*, Proc. ACM Principles of Programming Languages 1992.

[13] Cartwright, R. , P.-L. Curien, and M. Felleisen, *Fully abstract semantics for observably sequential languages*, Information and Computation **111**(2) (1994), 297-401.

[14] Chroboczek, J. , *Game semantics and subtyping*, in Proc. of the 15th annual IEEE symposium on Logic in Computer Science (LICS'00).

[15] Coppo, M., and M. Dezani, *A new type assignment for lambda-terms*, Archiv. Math. Logik **19** (1978), 139-156.

[16] Cousineau, G. , P.-L. Curien, and M. Mauny, *The categorical abstract machine*, Science of Computer Programming **8** (1987), 173-202.

[17] Curien, P.-L., "Combinateurs catégoriques, algorithmes séquentiels et programmation fonctionnelle", Thèse d'Etat, Univ. Paris 7 (1983).

[18] Curien, P.-L., "Categorical Combinators, Sequential Algorithms and Functional Programming", Pitman (1986), revised edition, Birkhäuser (1993).

[19] P.-L. Curien, Observable sequential algorithms on concrete data structures, Proc. 7th Symposium on Logic in Computer Science (LICS '92), Santa Cruz, published by ACM (1992).

[20] Curien, P.-L., *On the symmetry of sequentiality*, Proc. Mathematical Foundations of Programming Semantics '93, Springer Lect. Notes in Comp. Sci. 802, 29-71 (1994).

[21] Curien, P.-L., *Sur l'η-expansion infinie*, Comptes-Rendus de l'Académie des Sciences **334** Sec. I (2002), 77-82.

[22] Danos, V. , and R. Harmer, *Probabilistic game semantics*, ACM Transactions on Computational Logic **3**(3) (2002).

[23] Dimovski, A., D. R. Ghica, and R. Lazic. *Data-abstraction refinement: a game semantic approach*, in Proceedings of the 12th International Static Analysis Symposium (SAS'05), London (2005).

[24] Ghica, D.R., and Guy McCusker, *The regular-language semantics of second-order idealized ALGOL*, Theoretical Computer Science **309**(1-3) (2003), 469-502.

[25] Ghica, D.R., and A. Murawski, *Angelic semantics of fine-grained concurrency*, in Proc. of Foundations of Software Science and Computation Structures (FOSSACS'04), Lecture note in Computer Science 2987 (2004).

[26] Girard, J.-Y., *Linear logic*, Theoretical Computer Science **50**, 1-102 (1987).

[27] Girard, J.-Y., *On denotational completeness*, Theoretical Computer Science **227** (1999), 249-273.

[28] Harmer, R., and G. McCusker, *A fully abstract game semantics for finite nondeterminism*, in Proc. of the 14th annual IEEE symposium on Logic in Computer Science (LICS'99).

[29] Honda, K., and N.Yoshida, *Game-theoretic analysis of call-by-value computation*, Theoretical Computer Science **221**(1-2) (1999), 393-456.

[30] Hyland, J.M.E.., *A syntactic characterization of the equality in some models of lambda calculus*, J. London Math. Soc. **2** (1976), 361-370.

[31] Hyland, J.M.E.., and C.-H. L. Ong, Fair Games and Full Completeness for Multiplicative Linear Logic without the Mix-Rule. Preprint(1993).

[32] Hyland, J.M.E., and L. Ong, *On full abstraction for PCF I, II, and III*, Information and Computation **163**(1) (2000), 285-408 (preliminary draft circulated since 1994).

[33] Kahn, G., and G. Plotkin, *Concrete domains*, Theoretical Computer Science **121** (1993), 187-277 (appeared in French as TR IRIA-Laboria 336 in 1978).

[34] Laird, J., *Full abstraction for functional languages with control*, Proc. 12th Annual Symposium on Logic in Computer Science (LICS '97) (1997).

[35] Laird, J., *A Fully Abstract Bidomain Model of Unary FPC*. in Proc. of the Int. Conf. on Typed Lambda-Calculi and Applications (TLCA '03), Lecture Notes in Computer Science 2701 (2003).

[36] Laird, J., *Games and sequential algorithms*. To appear.

[37] Lamarche, F., *Sequentiality, games and linear logic* (manuscript) (1992).

[38] Loader, R., *Unary PCF is decidable*, Theoretical Computer Science **206**(1-2) (1998), 317-329.

[39] Loader, R., *Finitary PCF is undecidable*, Theoretical Computer Science **266**(1-2) (2001), 341-364.

[40] Longley, J., *The sequentially realizable functionals*, Annals of Pure and Applied Logic **117**(1-3) (2002), 1-93.

[41] Malacaria, P., and C. Hankin, *A new approach to control flow analysis*, in Proc. of Compiler Construction 1998, Lecture Notes in Computer Science 1383 (1998).

[42] Malacaria, P., and C. Hankin, *Generalised flowcharts and games*, in Proc. ICALP'98, Lecture Notes in Computer Science 1443 (1998).

[43] Malacaria, P., and C. Hankin, *Non-deterministic games and program analysis: an application to security*, in Proc. of the 14th annual IEEE symposium on Logic in Computer Science (LICS'99).

[44] Milner, R., *Fully abstract models of typed lambda-calculi*, Theoretical Computer Science **4** (1977), 1-23.

[45] Nickau, H., *Hereditarily sequential functionals*, Proc. Logical Foundations of Computer Science: Logic at St. Petersburg, Eds. A. Nerode and Yu. V. Matiyasevich, Lecture Notes in Computer Science 813, 253-264 (1994).

[46] O'Hearn, P., and J. Riecke, *Kripke logical relations and PCF*, Information and Computation **120**(1) (1995), 107-116.

[47] Ong, C.-H. L., *Observational equivalence of third-order Idealized Algol is decidable*, in Proceedings of IEEE Symposium on Logic in Computer Science 2002, Copenhagen Denmark, pages 245-256, Computer Society Press (2002).

[48] Paolini, L., "Lambda-theories: some investigations", PhD thesis, Università di Genova and Université de la Méditerranée (2003).

[49] Plotkin, G., *LCF as a programming language*, Theoretical Computer Science **5** (1977), 223-257.

[50] Scott, D., *A type-theoretical alternative to ISWIM, CUCH, OWHY*, Theoretical Computer Science **121**, 411-440 (1993) (manuscript circulated since 1969).

[51] Sieber, K., *Reasoning about sequential functions via logical relations*, Proc. Applications of Categories in Computer Science, Cambridge University Press (1992).

[52] Stoughton, A., *Mechanizing logical relations*, Proc. Mathematical Foundations of Progamming Semantics, Springer Lect. Notes in Comp. Sci. 802 (1994).

[53] Wadsworth, C., *The relation between computational and denotational properties for Scott's D-infinity models of the lambda-calculus*, SIAM J.ournal of Computing **5** (1976), 488-521.