EGIDIO ASTESIANO

GERARDO COSTA

Nondeterminism and fully abstract models

RAIRO. Informatique théorique, tome 14, nº 4 (1980), p. 323-347

<http://www.numdam.org/item?id=ITA_1980__14_4_323_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 14, n° 4, 1980, p. 323 à 347)

NONDETERMINISM AND FULLY ABSTRACT MODELS (*)

by Egidio ASTESIANO and Gerardo COSTA (1)

Communicated by M. NIVAT

Abstract. — *We study the semantics of some nondeterministic languages derived from PCF* [8]. *In particular we discuss how Milner's results in* [5] *can (or cannot) be applied to obtain models, for our languages, which are fully abstract with respect to the (standard) operational semantics.*

Résumé. — *Nous étudions ici la sémantique de quelques langages non deterministes, dérivés du PCF* [8]. *En particulier, nous discutons dans quelle mesure les résultats de Milner* [5] *peuvent être utilisés pour obtenir des modèles, pour nos langages, qui soient « pleinement abstraits » par rapport à la sémantique operationnelle usuelle.*

1. INTRODUCTION

In [4], Milner introduced the concept of *full abstraction* as a formalization of the idea of complete agreement between a mathematical (denotational) semantics, expressed by the function $\mathscr{S}[\quad]$, and an operational semantics, given by the function Eval, for a language $\mathscr{L}$. $\mathscr{S}[\quad]$ is *fully abstract w. r. t. Eval* if for every two terms M and N, of the same type: $\mathscr{S}[M]=\mathscr{S}[N]$ iff $\text{Eval}(\mathscr{C}[M])=\text{Eval}(\mathscr{C}[N])$, for any context $\mathscr{C}[\quad]$ such that $\mathscr{C}[M]$ and $\mathscr{C}[N]$ are programs (remark that $\mathscr{S}[\quad]$ is defined on all terms of $\mathscr{L}$, while Eval is defined for programs only). In the paper, we shall use the name *extrinsic full abstraction* to denote this property, when we are not interested in specifying the operational semantics.

More recently, in [5], Milner has considered a different notion of full abstraction, which expresses an intrinsic property of the mathematical semantics (or model), without any reference to the operational behaviour; we shall call it

(*) Receveid March 1979, revised October 1979.
(1) Istituto di Matematica dell'Università di Genova, via L.-B.-Alberti, 4, 16132 Genova, Italy.

intrinsic full abstraction. This property is defined by the condition: for any two terms of the same type, M and N, $\mathscr{S}[M]=\mathscr{S}[N]$ iff $\mathscr{S}[\mathscr{C}[M]]=\mathscr{S}[\mathscr{C}[N]]$, for any context $\mathscr{C}[\ ]$ such that $\mathscr{C}[M]$ and $\mathscr{C}[N]$ are programs.

These two properties are strongly related; indeed it is easy to see that if $\mathscr{S}[\ ]$ is intrinsicly fully abstract, then it is also fully abstract w.r.t. Eval, whenever $\mathrm{Eval}(P)=\mathscr{S}[P]$ for any program P; and this seems to be a minimum requirement for our semantic functions.

After the concept of extrinsicly fully abstract semantics has been introduced, some papers have shown the difficulties of finding such semantics for deterministic [2, 8] and nondeterministic [3] languages. An important step towards the solution of this problem, in the light of the above remark, has been made in [5], where Milner has given a powerful technique for building intrinsicly fully abstract models (hence semantics, *see* [2]) for typed λ-calculi.

We discuss here the application of Milner's results to nondeterministic languages derived from λ-calculus. To this end we introduce a hierarchy of three languages, NDL0, NDL1, NDL2: NDL0 is a nondeterministic version of PCF [8] already investigated in [3], while NDL1 and NDL2 are successive extensions of it (which will be justified in the following).

We consider for each language a "natural" interpretation and show that, while the interpretations for NDL1 and NDL2 are SFP interpretations (even articulate for NDL2), the one for NDL0 is not; hence Milner's results can be applied to obtain intrinsicly fully abstract models for NDL1 and NDL2, but not for NDL0. Then, following Plotkin's style [8] we give an operational semantics for NDL1 and NDL2 and show that their Milner's models are fully abstract with respect to these semantics.

This paper shows, for one respect, that Milner's conditions, not very restrictive for deterministic languages, are perhaps restrictive in nondeterministic cases, since NDL0 is a "natural" extension of PCF; hence the need for other powerful techniques in building fully abstract models. But it seems to us that also another interpretation is possible, namely that NDL0 is too poor, since its primitives are not able to perform some basic operations on sets.

We have tacitly assumed throughout the work a "call by name" rule as a natural one, since our aim was to discuss the connection with Milner's results. But we think that this is a point that deserves discussion and in [1] we consider a different operational semantics, which involves a sharing technique of evaluation, and outline a fully abstract model for it.

We assume here that the reader is familiar with references [8] and [5].

2. THE LANGUAGES NDL0, NDL1, NDL2.

2.1. Types

We consider the set T of functional types generated, in the usual way, from *ground types* o and ι. In what follows, σ and τ range over T and $\varkappa$ ranges over $\{o, \iota\}$. We write $\sigma_1 \times \ldots \times \sigma_n \to \varkappa$ instead of $(\sigma_1 \to (\ldots \to (\sigma_n \to \varkappa)\ldots))$ and call *first order type* any type of the form $\varkappa_1 \times \ldots \times \varkappa_n \to \varkappa$, for $n \geqq 1$.

2.2. Terms

The set of terms in NDLJ, $J=0, 1, 2$, is the set generated by:

– CJ, the set of ground constant symbols (typical element c);

– FJ, the set of first order constant symbols;

– $\mathrm{VAR}=\{X_i^\sigma, \sigma \in T, i \geqq 1\}$; $\mathrm{FIX}=\{Y_\sigma, \sigma \in T\}$;

using typed λ-abstraction and application, where:

– $\mathrm{C0}=\{\underline{tt}, \underline{ff}$ [of type o], $\underline{0}, \underline{1}, \ldots, \underline{n}, \ldots$ [of type ι]$\}$;

– $\mathrm{C1}=\mathrm{C2}=\mathrm{C0} \cup \{\emptyset_o$ [of type o], $\emptyset_\iota, \underline{e}$ [of type ι] $\}$;

– $\mathrm{F0}=\{(+1), (-1)$ [of type $\iota \to \iota$]; Z [of type $\iota \to o$], $IF_\varkappa$ [of type $o \times \varkappa \times \varkappa \to \varkappa$], $\underline{or}_\varkappa$ [of type $\varkappa \times \varkappa \to \varkappa$]$\}$;

– $\mathrm{F1}=\mathrm{F0} \cup \{E$ [of type $\iota \to o$]$\}$;

– $\mathrm{F2}=\mathrm{F1} \cup \{\exists\, T), (T!)$ [of type $o \to o$]$\}$.

There are two differences between NDL0 and the language used by Henessy and Ashcroft. In [3], the set of primitives includes IF simbols of higher type and does not include $\underline{or}$'s; $\underline{or}$ appears only in combinations like $M\ \underline{or}\ N$, where M and N are of the same type, not necessarily ground. We have chosen to introduce $\underline{or}$'s as first order functions for technical reasons (it allows us to place ourselves in the framework of Milner's paper [5]); the semantic meaning is not changed.

For higher type $\underline{or}$'s and IF's, the fact is that w. r. t. the interpretation we have in mind for them (which is the same as in [3], and seems to us the natural one) those symbols are not needed, as we can define the corresponding functionals by closed terms, using ground $\underline{or}$'s and IF's.

Hence we can say that NDL0 is "the same as" (or at least equivalent to) the language in [3].

For what concerns the primitives $\emptyset_\varkappa$, e, E, $(\exists\ T)$, $(T!)$, their meaning, and the reasons why we have introduced them, will be seen in section 3.

2.3. Notations

We adopt the usual conventions for suppressing redundant parenthesis in writing terms and we shall often omit subscripts in primitive symbols and variables. We use letters M, N to denote arbitrary terms, adding sometimes a subscript to indicate the type. Closed terms of ground type will be called *programs*.

Following Plotkin [8] we define Ω_σ and $Y_\sigma^{(n)}$ by: $\Omega_\varkappa = Y_\varkappa(\lambda X_1^\varkappa X_1^\varkappa)$, $\Omega_{\sigma\to\tau} = \lambda X_1^\sigma \Omega_\tau$, $Y_\sigma^{(0)} = \Omega_{(\sigma\to\sigma)\to\sigma}$ and $Y_\sigma^{(n+1)} = \lambda X(X(Y_\sigma^{(n)} X))$, where X stands for $X_1^{\sigma\to\sigma}$. Then $M^{(n)}$, $n \geqq 0$ is the term obtained from M by replacing all occurrences of Y_σ by $Y_\sigma^{(n)}$, for all σ's.

Finally, we call *finite* any *term* which contains Y's only in the combination for $\Omega_\varkappa$ (e.g. $M^{(n)}$ is finite, for any M, n).

3. MODELS AND DENOTATIONAL SEMANTICS

3.1. Domains

It has been pointed out by Hennessy and Ashcroft in [3] that, considering a non deterministic language, and looking at ground *or*'s as denoting set union, if we want to model a call-by-name with copy-rule evaluation mechanism, we should consider that first order primitives denote functions mapping subsets into subsets, rather then elements into subsets (*see* [1] for the inverse approach: assuming denotations of first order primitives to be functions from elements into subsets we discuss an adequate operational semantics). So we shall take, here, our ground domains to be powerdomains. We need to extend the usual definition [7, 10] to accomodate the empty set. In our case, we regard it as a maximal element; but we are aware of the fact that this might be unsatisfactory in other situations.

We assume the reader is familiar with the notions of (flat) cpo, finite (isolated) element, ($\omega -$) algebraic cpo, consistently complete cpo, continuous function (*see* e.g. [8, 2]). We use the symbols $\sqsubseteq$, $\sqcap$, $\sqcup$ and $\subseteqq$, $\cap$, $\cup$ to denote: the order relation, the glb, the lub (in domains) and inclusion, intersection, union (in sets), respectively. Moreover, if A is a set and A_1, $A_2 \subseteqq A$, then $A_1 \setminus A_2$ stands for $\{a \mid (a \in A_1) \wedge (a \notin A_2)\}$.

Given a *flat cpo D, the* (Egli-Milner) *ordering between non-empty subsets* of D is given by

$$A \sqsubseteq B \quad \text{iff} \quad \left[\begin{array}{l} \forall a \in A,\ \exists b \in B \text{ s.t. } a \sqsubseteq b \\ \forall b \in B,\ \exists a \in A \text{ s.t. } a \sqsubseteq b \end{array}\right] \quad \text{iff} \quad \left[\begin{array}{l} \text{either } A = B \\ \text{or } (\bot \in A) \wedge (A \setminus \{\bot\} \subseteqq B). \end{array}\right.$$

Then the *power-domain of D*. $\mathbb{P}[D]$, is defined to be the set $\{A \mid \emptyset \neq A \subseteqq D, |A| = \infty \Rightarrow \bot \in A\}$, together with the above order.

We shall consider an *extended powerdomain*, $\mathbb{P}_{\emptyset}[D]$, obtained by adding the empty set to $\mathbb{P}[D]$ and extending $\sqsubseteq$ by

$$\{\perp\}\sqsubseteq\emptyset;\ A \text{ and } \emptyset \text{ are incomparable if } A\neq\{\perp\}.$$

LEMMA 3.1.1: *For any denumerable flat cpo D, $\mathbb{P}_{\emptyset}[D]$ ($\mathbb{P}[D]$) is an ω-algebraic, consistently complete cpo, whose finite elements are the finite (non-empty) sets; moreover in $\mathbb{P}_{\emptyset}[D]$ and $\mathbb{P}[D]$ the binary union function is continuous.* □

Consider flat cpo's $D_0, \ldots, D_n$ and a monotonic function g: $D_1\times\ldots\times D_n\rightarrow D_0$ and denote by $\hat{g}$ the function from $2^{D_1}\times\ldots\times 2^{D_n}$ into 2^{D_0}, given by

$$\hat{g}(A_1, \ldots, A_n)=\{g(a_1, \ldots, a_n)\mid a_i\in A_i\}.$$

It is easy to verify that the restrictions of $\hat{g}$ to the powerdomains and to the extended powerdomains are continuous functions. We shall use the same symbol $\hat{g}$ for both (notice that $\hat{g}(\ldots, \emptyset, \ldots)=\emptyset$). The same applies to any function

$$h: D_1\times\ldots\times D_j\times 2^{D_{j+1}}\times\ldots\times 2^{D_n}\rightarrow 2^{D_0}\ (\hat{h}: 2^{D_1}\times\ldots\times 2^{D_n}\rightarrow 2^{D_0}).$$

3.2. Interpretations

Following Milner [5] we specify a domain (cpo) for each ground type and the interpretation for constant and first order symbols, leaving the Y's uninterpreted (as we have not higher type domains; the meaning of the Y's will be given in the model); we refer the reader to Milner's paper for motivations.

We shall consider just one interpretation for each language and keep it fixed for the rest of the paper. Let J be 0, 1, 2; then the interpretation $\mathscr{I}_J$ for NDLJ is given by the cpo's D_o^J, D_ι^J and the constant/continuous function $\mathscr{I}_J(\alpha)$, for any constant/function symbol α; and precisely:

a) $D_o^0=\mathbb{P}[\mathbb{T}]$; $D_o^1=D_o^2=\mathbb{P}_{\emptyset}[\mathbb{T}]$; $D_\iota^0=\mathbb{P}[\mathbb{N}_\perp]$; $D_\iota^1=D_\iota^2=\mathbb{P}_{\emptyset}[\mathbb{N}_\perp^e]$,
where $\mathbb{T}$ is the usual cpo of truth values, $\mathbb{N}=\{0, 1, 2, \ldots\}$, $\mathbb{N}^e=\mathbb{N}\cup\{e\}$, e for "error", and $\mathbb{N}_\perp$, $\mathbb{N}_\perp^e$ are the respective flat cpo's;

b) $\mathscr{I}_J(\underline{c})=\{c\}$, any ground constant symbol c;

(*c*) $\mathscr{I}_J(\underline{or}_\chi)=$ union function;

d) $\mathscr{I}_J(IF_\chi)=\widehat{\text{cond}}_\chi$, where cond_χ is the usual conditional function on D_χ^J;

e) $\mathscr{I}_0((+1))=\widehat{\text{succ}}$; $\mathscr{I}_0((-1))=\widehat{\text{pred}}$; $\mathscr{I}_0(Z)=\widehat{\text{zero}}$, where succ, pred, zero are the usual successor, predecessor, zero functions on $\mathbb{N}_\perp$ (remark that $\text{pred}(0)=\perp$);

f) $\mathscr{I}_I((+1))=\widehat{\text{succ}_e}$; $\mathscr{I}_I((-1))=\text{pred}_e$; $\mathscr{I}_I(Z)=\text{zero}_e$; $\mathscr{I}_I(E)=\widehat{\text{error}}$, where $I=1, 2$ and, if x ranges over $\mathbb{N}^e_\perp$, we have:

$\text{succ}_e=\lambda x.\text{if } x=e \text{ then } e \text{ else } x+1$,

$\text{pred}_e=\lambda x.\text{if } (x=0)\vee(x=e) \text{ then } e \text{ else } x-1$,

$\text{zero}_e=\lambda x.\text{if } x=0 \text{ then } tt \text{ else } ff$,

$\text{error}=\lambda x.\text{if } x=e \text{ then } tt \text{ else } ff$;

g) $\mathscr{I}_2(\exists T)=(\exists tt)$; $\mathscr{I}_2(T!)=(tt!)$, where, if $B\subseteq\mathbb{T}$,

$$(\exists tt)(B)=\begin{cases}\{tt\}, & \text{if } tt\in B;\\ \{ff\}, & \text{if } B=\{ff\},\emptyset;\\ \{\perp\}, & \text{otherwise.}\end{cases}$$

$$(tt!)(B)=\begin{cases}\{tt\}, & \text{if } B=\{tt\};\\ \{ff\}, & \text{if } (ff\in B)\cup(B=\emptyset);\\ \{\perp\}, & \text{otherwise.}\end{cases}$$

3.3. Models and semantics derived from models

The definitions given below are not specific to the languages we consider, but apply in general to typed λ-calculi with primitives, both deterministic [8, 5, 2] and nondeterministic (at least as far as nondeterminism is of the kind we have considered).

Let J be 0, 1 or 2 in what follows. Then an **order extensional model (model** for short) $\mathscr{M}$ for the couple (NDLJ, $\mathscr{I}_J$) is given by [2]:

(i) a family $\{D^{\mathscr{M}}_\sigma \mid \sigma\in T\}$ of cpo's, s. t. $D^{\mathscr{M}}_\kappa=D^J_\kappa$;

(ii) a two place continuous mapping $-\,.\,-: D^{\mathscr{M}}_{\sigma\to\tau}\times D^{\mathscr{M}}_\sigma\to D^{\mathscr{M}}_\tau$, for any σ, τ in T, such that:

– if $d\in D^{\mathscr{M}}_\sigma$ then $\perp.d=\perp$,

–$(\forall d\in D^{\mathscr{M}}_\sigma,\ g.d\sqsubseteq g'.d)\Rightarrow g\sqsubseteq g'$, for any g, g' in $D^{\mathscr{M}}_{\sigma\to\tau}$.

Notice that the converse of this implication is given by the monotonicity of .. So we can identify $D^{\mathscr{M}}_{\sigma\to\tau}$ to a subset of $[D^{\mathscr{M}}_\sigma\to D^{\mathscr{M}}_\tau]$.

(iii) for any term M_σ, a continuous mapping $\mathscr{M}[M_\sigma]$: $\text{Env}^{\mathscr{M}}\to D^{\mathscr{M}}_\sigma$ (where $\text{Env}^{\mathscr{M}}$ denotes the set of environments, i. e. type-preserving maps from VAR into $\cup_{\sigma\in T}D^{\mathscr{M}}_\sigma$; which is a cpo w.r.t. the pointwise order) such that, $\forall\rho\in\text{Env}^{\mathscr{M}}$:

$$\mathscr{M}[\Omega_\kappa]\,\rho=\perp_\kappa;$$

$$\mathscr{M}[X^\sigma_i]\,\rho=\rho(X^\sigma_i);$$

$$\mathscr{M}[\alpha]\,\rho=\mathscr{I}_J(\alpha),$$

for any constant/first order function symbol α;

$$\mathcal{M}[MN]\rho=\mathcal{M}[M]\rho.\mathcal{M}[N]\rho;$$
$$\mathcal{M}[\lambda X_i^\sigma M]\rho.d=\mathcal{M}[M]\rho[d/X_i^\sigma],$$

where $\rho[d/X_i^\sigma](x)=$ if $x=X_i^\sigma$ then d else $\rho(x)$, for any d in $D_\sigma^{\mathcal{M}}$;

$$\mathcal{M}[Y]\rho=\bigsqcup_{n\geq 0}\mathcal{M}[Y^{(n)}]\rho=\text{fix},$$

where fix denotes the least fixed point operator.

The **(denotational) semantic mapping, denotational semantics** for short, **associated to** $\mathcal{M}$ is the (type-preserving) mapping

$$\mathcal{M}[\quad]: \text{NDL}J\to\cup_{\sigma\in T}[\text{Env}^{\mathcal{M}}\to D_\sigma^{\mathcal{M}}].$$

The **denotational pre-order** $\sqsubseteq_{\mathcal{M}}$ is defined by: $M\sqsubseteq_{\mathcal{M}}N$ iff $\mathcal{M}[M]\sqsubseteq\mathcal{M}[N]$; we denote by $\equiv_{\mathcal{M}}$ the induced equivalence. We say that $\mathcal{M}$ ($\mathcal{M}[\quad]$) is an **intrinsicly fully abstract model (semantics),** whenever, for any two terms M, N of type σ, $M\sqsubseteq_{\mathcal{M}}N$ iff $\mathcal{C}[M]\sqsubseteq_{\mathcal{M}}\mathcal{C}[N]$ for all contexts $\mathcal{C}[\quad]$ s.t. $\mathcal{C}[M]$ and $\mathcal{C}[N]$ are programs (a context is simply a term with one or more holes).

We shall need later on the following lemma.

LEMMA 3.3.1 [5]: *In any model* $\mathcal{M}$, *if* M, M' *are closed terms of type* $\sigma_1\times\ldots\times\sigma_n\to\varkappa$, *then:* $\mathcal{C}[M]\sqsubseteq_{\mathcal{M}}\mathcal{C}[M']$ *for all contexts* $\mathcal{C}[\quad]$ *s.t.* $\mathcal{C}[M]$ *and* $\mathcal{C}[M']$ *are programs* iff *for all closed* N_i *of type* σ_i, $1\leq i\leq n$, $MN_1\ldots N_n\sqsubseteq_{\mathcal{M}}M'N_1\ldots N_n$. $\square$

We say that **an element** d in $D_\sigma^{\mathcal{M}}$ **is (finitely) definable in the language** NDLJ if there is a (finite) closed term M in NDLJ such that $d=\mathcal{M}[M_\sigma](\perp)$, where $\perp$ denotes the totally undefined environment.

3.4. Continuous functions models

Let J be 0, 1 or 2, as before, and I be 0 or 1.

A first example of model for (NDLJ, $\mathcal{I}_J$) is $\mathcal{S}_J$, the usual Scott-Milner continuous functions model built starting from ground domains D_o^J, D_ι^J. However it is easy to notice that $\mathcal{S}_I$ is "too large" in the sense that too many finite elements in $D_\sigma^{\mathcal{S}_I}$ are not definable, for σ non-ground.

A "smaller" model, $\mathcal{H}_I$, (for NDLI, $\mathcal{I}_I$) can be given following Hennessy and Ashcroft (*see* [3] for motivations and details). Indeed, if $\subseteq_\varkappa$ is the usual set inclusion relation on $D_\varkappa^I$, one can take $D_{\varkappa\to\varkappa'}^{\mathcal{H}_I}$, to be $\{f\mid f\colon D_\varkappa^I\to D_{\varkappa'}^I, f$ *is* $\sqsubseteq$-

continuous and $\sqsubseteq$-monotonic}; on those domains, $\sqsubseteq_{\varkappa\to\varkappa'}$ can be defined pointwise and the construction iterated (notice that this kind of model does not fit (NDL2, $\mathscr{I}_2$) because ($\exists\, tt$) and ($tt!$) are typically non $\sqsubseteq$-monotonic). The following lemma shows that $\sqsubseteq$-monotonicity is not sufficient, as a restriction, to ensure full abstraction. We conjecture that the lemma is true for $\mathscr{S}_2$ also, but we have not yet a proof of this.

LEMMA 3.4.1: *The models $\mathscr{H}_I$ and $\mathscr{S}_I$ are not intrinsicly fully abstract for* (NDLI, $\mathscr{I}_I$), $I=0, 1$.

Proof: One can show (*see* Appendix) that no function φ, s.t.

$$\varphi(\{tt\}, \{\perp\}) = \varphi(\{\perp\}, \{tt\}) = \{tt\}, \qquad \varphi(\{ff\}, \{ff\}) = \{ff\}$$

can be defined in NDLI. Now consider M_1 and M_2 defined as in [8], where X is $X_1^{o\times o\to o}$ and $n=1, 2$:

$$M_n = \lambda X\,(IF_\iota\,(X\,\underline{tt}\,\Omega_o)\ (IF_\iota\,(X\,\Omega_o\,\underline{tt})\ (IF_\iota\,(X\,\underline{ff}\,\underline{ff})\ \Omega_\iota\,\underline{n})\ \Omega_\iota)\ \Omega_\iota).$$

Clearly, if $[\![\ \ ]\!]$ is either $\mathscr{H}_I[\![\ \ ]\!]$ or $\mathscr{S}_I[\![\ \ ]\!]$ and φ is as above, $([\![M_n]\!]\,8\,(\perp))(\varphi)=n$; hence, omitting subscripts $\mathscr{S}_I$, $\mathscr{H}_I$ in $\sqsubseteq$, $M_1 \not\sqsubseteq M_2$ *and* $M_1 \not\sqsupseteq M_2$. However, using lemma 3.3.1 and non-definability of φ it is immediate to show that $\mathscr{C}[M_1]\equiv\mathscr{C}[M_2]$, for any context $\mathscr{C}[\ \ ]$ s.t. $\mathscr{C}[M_1]$ and $\mathscr{C}[M_2]$ are programs. Indeed: M_n is of type $(o\times o\to o)\to\iota$; if N is any term of type $(o\times o\to o)$ and we denote $[\![M_n N]\!](\perp)$ by α_n, then $\alpha_n=([\![M_n]\!](\perp))([\![N]\!](\perp))$; to get $\alpha_1\neq\alpha_2$ we should have that $[\![N]\!](\perp)$ satisfies the conditions for φ. $\square$

3.5. SFP interpretations and Milner's models

An SFP object [7] is any cpo D for which an ascending chain $\psi_1, \ldots, \psi_n, \ldots$ of finite projections exists such that its lub is the identity on D, id_D (by finite projection we mean a continuous function $\psi: D\to D$ s.t. $\psi\circ\psi=\psi$, $\psi\sqsubseteq id_D$ and $\psi(D)$ is a finite set).

An interpretation is SFP [2] when, for each ground type $\varkappa$, the interpretation domain $D_\varkappa$ is an SFP-object and all finite elements in $D_\varkappa$ together with a chain of finite projections whose lub is $id_{D_\varkappa}$ are finitely definable.

An SFP-interpretation is **articulate** [5] if, for each ground type $\varkappa$: $D_\varkappa$ is ω-algebraic and consistently complete; the binary glb-function for $D_\varkappa$ (which exists and is continuous because of the given conditions on $D_\varkappa$) is definable; for some fixed $\bar{\varkappa}$ and $\bar{a}\in D_{\bar{\varkappa}}$, $\bar{a}$ finite and $\neq\perp$, and each finite a in $D_\varkappa$, the $\sqsupseteq$ **function** $(\sqsupseteq a) \underset{\text{def}}{=} \lambda(d\in D_\varkappa)$ if $d\sqsupseteq a$ then $\bar{a}$ else $\perp$ is finitely definable.

LEMMA 3.5.1: *a)* $\mathscr{I}_0$ *is not an SFP-interpretation;*

b) $\mathscr{I}_1$ *is an SFP-interpretation which is not articulate;*

c) $\mathscr{I}_2$ *is an articulate SFP-interpretation.*

Proof: a) It is the condition about definability of finite projections for $\mathbb{P}[\mathbb{N}_\perp]$ which is not fulfilled. Indeed, using Plotkin's techniques [6, 9] one can show that no function φ s. t. $\varphi(\{0, 1\}) = \{0, 1\}$ and $\varphi(\bar{n}) = \{\perp\}$, some $\bar{n} \geqq 2$, is definable in NDL0 (*see* Appendix);

b) $\mathscr{I}_1$ is not articulate because some $\sqsupseteq$-functions cannot be defined as they are not monotonic w.r.t. inclusion (e. g. $(\sqsupseteq \{tt\})$).

To show that $\mathscr{I}_1$ is SFP, the non trivial part (given lemma 3.1.1) is the definability of finite projections for D_ι^1; ψ_n is given by the term $ID^{(n)}$, where

$$ID = \lambda X^\iota (IF(EX^\iota)\,\underline{e}(MX^\iota)),$$

$$M = Y(\lambda X^{\iota\to\iota}(\lambda X^\iota(IF(ZX^\iota)\underline{0}(IF(EX^\iota)\emptyset_\iota((+1)(X^{\iota\to\iota}((-1)X^\iota))))))).$$

REMARK: Analising the above definition (which is a recursive definition of the identity on $\mathbb{P}_0[\mathbb{N}^c]$) we see the technical reasons for introducing $\underline{e}$, E, $\underline{\emptyset}_\chi$ (with the given interpretation). We point out, however, that for $\underline{e}$ (and e) there is also a deeper motivation: we think that $\perp$ should stand for lack of information and/or for non termination only and not for *precise* illegal situations like $0-1$. Technically, we need $\underline{e}$, and the fact that $0-1$ is e (and not $\perp$ as in [8] and [3]), to insure termination of the process, when starting from a finite maximal element (i.e. a finite set not containing $\perp$). The symbol E and its interpretation, the function error, are needed (in M) to test this termination of the process, i.e. the fact that by successive decrements by 1 we have explored the whole set. Finally, $\emptyset_\chi$ provides us with the equivalent of a null statement;

c) $\mathscr{I}_2$ is SFP as $\mathscr{I}_1$ is such. To prove that it is articulate, we give the definitions of the $\sqsupseteq$-functions and glb-functions.

In what follows ::=, NEG, AND stand for "is defined by", $\lambda X(IFX\,\underline{ff}\,\underline{tt})$, $\lambda X_1 X_2(IFX_1(IFX_2\,\underline{tt}\,\underline{ff})\,\underline{ff})$, respectively.

$\sqsupseteq$-*functions*:

We choose $\bar{\varkappa} = o$ and $\bar{a} = \{tt\}$. Moreover, if a is finite, let $(> a)$ be the function

$$\lambda d.\text{if } d \sqsupseteq a \text{ then } \{tt\} \text{ else } \quad \text{if } \; d \# a \text{ then } \{ff\} \text{ else } \perp$$

(where $d \# a$ means that d and a have no upper bound).

Clearly if M defines $(> a)$, the term $\lambda X(IF(MX)\,\underline{tt}\,\Omega)$ defines $(\sqsupseteq a)$.

$>$*-functions for* $\mathbb{P}_\varnothing[\mathbb{T}]$:

1) $(>\bot) ::= \lambda X\, \underline{tt}$;

2) $(>\varnothing) ::= EMPTY_o = \lambda X\ (NEG\ ((\exists T)\ (IF\ X\ \underline{tt}\ \underline{tt})))$;

3) $(>\{\bot, tt\}) ::= (\exists T)$; $(>\{tt\}) ::= (T!)$;

4) $\begin{cases} (>\{\bot, ff\}) ::= (\exists F) = \lambda X\, (IF\, (EMPTY_o\, X)\, \underline{ff}\, (NEG\, ((T!)\, X))), \\ (>\{ff\}) ::= (F!) \text{ same as } (\exists F) \text{ but replacing } (T!) \text{ with } (\exists T); \end{cases}$

5) $\begin{cases} (>\{\bot, ff, tt\}) ::= \tilde{M} = \lambda X\, (AND\, ((\exists T)\, X)\, ((\exists F)\, X)), \\ (>\{ff, tt\}) ::= \lambda X\, (IF\, (\tilde{M} X)\, ((T!)\, (IFX\, \underline{tt}\, \underline{tt}))\, \underline{ff}). \end{cases}$

$>$*-functions for* $\mathbb{P}_\varnothing[\mathbb{N}_\bot^e]$:

1) $(>\{\bot\}) ::= \lambda X\, \underline{tt}$;

2) $(>\varnothing ::= EMPTY_\iota = \lambda X\ (NEG\ ((\exists T)\ (IF\ (ZX)\, \underline{tt}\ \underline{tt})))$.

In what follows let *if* M' *then* M'' *else* M''' stand for $IFM'M''M'''$ and $M-\underline{n}$ stand for n times (-1) applied to M.

If A *denotes a finite element* in $\mathbb{P}_\varnothing[\mathbb{N}_\bot^e]$, i. e. a finite set, different from $\varnothing$ and $\{\bot\}$, and B is in $\mathbb{P}_\varnothing[\mathbb{N}_\bot^e]$, then:

$\bot \in A$ gives:

$$B \sqsupseteq A \quad \text{iff } A \setminus \{\bot\} \subseteq B;$$

$$B \# A \quad \text{iff } (\bot \notin B,\ B \not\sqsupseteq A);$$

$\bot \notin A$ gives:

$$B \sqsupseteq A \quad \text{iff } B = A;$$

$$B \# A \quad \text{iff } (\exists c : c \neq \bot,\ c \in B,\ c \notin A) \vee (\bot \in B,\ B \neq A).$$

Hence:

3) $\begin{cases} (>\{\bot, 0\}) ::= (\exists Z) = \lambda X\, ((\exists T)(ZX)); \\ (>\{0\}) ::= (Z!) = \lambda X\, ((T!)(ZX)); \end{cases}$

4) $\begin{cases} (>\{\bot, e\}) ::= (\exists E) = \lambda X\, ((\exists T)(EX)); \\ (>\{e\}) ::= (E!) = \lambda X\, ((T!)(EX)); \end{cases}$

5) let A be $\{\bot, n_1, \ldots, n_m\}$, $n_j \in \mathbb{N}$, $A \neq \{\bot, 0\}$, then

$$(>A) ::= N_1 = \lambda X\, (\textit{if}\, (\exists Z)(X - \underline{n_1})\ \textit{then}\ (\ldots(\textit{if}\, (\exists Z)\, X - \underline{n_m})\ \textit{then}\ \underline{tt}\ \textit{else}\ \underline{ff})\ldots)\ \textit{else}\ \underline{ff});$$

$$(>A \cup \{e\}) ::= \lambda X\, (\textit{if}\, (\exists E)\, X\ \textit{then}\ N_1 X\ \textit{else}\ \underline{ff});$$

6) for the case $\bot \notin A$, $A \neq \varphi$, $\{0\}$, $\{e\}$, the general formula is rather long, we prefer to give an example (written following the pattern):

$(>\{1\}) ::= \lambda X$ (*if* $(\exists E) X$ *then* $\underline{ff}$ *else if* $(\exists Z) X$ *then* $\underline{ff}$ *else*

if $(\exists Z)(X-\underline{1})$ *then* (*if* $(E!)(X-\underline{2})$ *then* $\underline{tt}$ *else* $\underline{ff}$) *else* $\underline{ff}$).

Glb functions:

First notice that, for flat D, if A and B are non-empty subsets of D, then $A \sqcap B = (A \cap B) \cup$ (if ($\exists d$, $d \in A$, $d \notin B$ or conversely) then $\{\bot\}$ else $\emptyset$. Then letting x, y and w stand for $X_1^{\varkappa}$, $X_2^{\varkappa}$ and $X_1^{\iota \times \iota \to \iota}$ and denoting the two glb functions by $\sqcap_o$ and $\sqcap_\iota$ respectively we have:

$\sqcap_\varkappa ::= \lambda x (\lambda y (\mathit{if}\ (EMPTY_\varkappa\ x)\ \mathit{then}\ (\mathit{if}\ (EMPTY_\varkappa\ y)\ \mathit{then}\ \underline{\emptyset}_\varkappa\ \text{else}\ \Omega_\varkappa)\ \mathit{else}\ P_\varkappa))$,

where:

$$P_0 = \mathit{if}\, x\ \mathit{then}\ Q_t\ \mathit{else}\ Q_f;$$

$Q_t = \mathit{if} (\exists T)$ y *then* $\underline{or}_o\ \underline{tt}$

(*if* $(T!)\,y$ *then* (*if* $(T!)\,x$ *then* $\emptyset_o$ *else* Ω_0)

else (*if* $(T!)\,x$ *then* Ω_o *else* Φ_0))

else Ω_o;

Q_f is like Q_t but for the replacement of $(\exists T)$, $(T!)$ and $\underline{tt}$ by $(\exists F)$, $(F!)$-see above- and $\underline{ff}$, respectively.

$$P_\iota = \mathit{if}\ \text{Ex}\ \mathit{then}\ Q_e\ \mathit{else}\ \text{Mxy};$$

where: Q_e is obtained from Q_t by changing type and replacing $(\exists T)$, $(T!)$, $\underline{tt}$ with $(\exists E)$, $(E!)$, $\underline{e}$.

$M = Y(\lambda w (\lambda x (\lambda y$ (*if* $Z x$ *then* Q_0 *else*

if $(E!)\,x$ *then* (*if* $(E!)\,y$ *then* $\emptyset_\iota$ *else* Ω_ι)

else $(+1)(w((-1)\,x)((-1)\,y))))))$,

where Q_0 is obtained from Q_t in the (now) usual way. □

REMARK: It is not hard to *see* that part *c*) can be proved also for a language obtained by adding to NDL1 symbols for ($\sqsubseteq \emptyset$, ($\sqsubseteq \{tt\}$), ($\sqsubseteq \{\bot, tt\}$), and probably this is a minimal extension of NDL1 to this purpose.

However the above functions are rather strange from a computational point of view; e.g. $(\sqsupseteq \{tt\})(\{ff\}) = \{\bot\}$ implies that having computed $\{ff\}$ we force our computation to loop (as we have already pointed out, we think that a stuck computation should not correspond to $\perp$). For this reason we have preferred to use ($\exists T$) and ($T!$) as primitives.

Another point is that ($\exists T$) and (T !) require a certain amount of *parallelism* in the evaluation mechanism (*see* next section). However this is mandatory if we want an articulate interpretation, as we need some non $\subseteqq$-monotonic primitives, which always require parallel evaluation of their arguments.

The above lemma implies that Milner's results [5] apply to (NDL1, $\mathscr{I}_1$) and (NDL2, $\mathscr{I}_2$) so we have as a corollary.

THEOREM 3.5.2: *The couple* (NDL1, $\mathscr{I}_I$) $I=1, 2$ *admits an intrinsicly fully abstract model, which is unique for* $I=2$. $\square$

For the construction of this model, which we denote by $\mathscr{M}_I$, we refer the reader to Milner's paper; here we shall be mainly concerned with its properties.

We simply mention some relations between $\mathscr{M}_I$ and the models in section 3.4. If $\equiv$ and $\underset{\sim}{\subset}$ denote isomorphism (between domains) and inclusion up to isomorphism, respectively, we have:

(i) $D_\varkappa^{\mathscr{M}_1} = D_\varkappa^{\mathscr{M}_2} \equiv D_\varkappa^{\mathscr{H}_1} = D_\varkappa^{\mathscr{S}_1} = D_\varkappa^{\mathscr{S}_2}$;

(ii) $D_{\sigma \to \tau}^{\mathscr{M}_2} \underset{\sim}{\subset} [D_\sigma^{\mathscr{M}_2} \to D_\tau^{\mathscr{M}_2}]$ (*see* the definition of model, section 3.3);

(iii) defining $\subseteqq_\varkappa$ on $D_\varkappa^{\mathscr{M}_1}$ by isomorphism with $D_\varkappa^1$ and $\subseteqq_\sigma$ on $D_\sigma^{\sim 1}$ inductively, we have:

$$D_{\sigma \to \tau}^{\mathscr{M}_1} \underset{\sim}{\subset} \{ f \mid f \in [D_\sigma^{\mathscr{M}_1} \to D_\tau^{\mathscr{M}_1}],\ f \text{ is } \subseteqq\text{-monotonic} \}.$$

The inclusion in (iii) is strict, indeed one can see that if φ is the natural extension to sets of the parallel disjunction [8] then φ is in $\mathscr{H}_1$ while it has no image in $\mathscr{M}_1$ (*see* Appendix). Intuitively, this happens because the nondeterministic <u>or</u> is not sufficient to substantially change the sequential nature of NDL1.

For the inclusion in (ii), we conjecture that it is strict. In other words, we believe that we have not introduced enough parallelism in NDL2.

At this point, one could ask what happen if we added a parallel conditional, pcond, to NDL2 (notice that, differently from [8], we should have: pcond $(\perp, x, y) = x \sqcap y$, as our ground domains are not flat). We leave this question open; we think it would be worthwhile to investigate it from a more general point of view, i. e. the study of languages to express operations on sets.

For what concerns (NDL0, $\mathscr{I}_0$), lemma 3.5.1 *a*) implies that we cannot use Milner's results to give a fully abstract model for it. Hence, for this language, we are not able to improve the results in [3] (Berry's *stable models* [2] do not apply to our case; indeed the interpretation of $\underline{or}$, the union function, is not stable and not even consistently multiplicative).

Hence, *from now on we shall be concerned with* NDL1 *and* NDL2 *only.*

4. OPERATIONAL SEMANTICS

4.1. Direct derivation

Following Plotkin [8] we define a relation $\to$ (direct derivation) between terms, by the rules below (they are for NDL2; for NDL1 consider rules 1 to 8 only).

We have already remarked that $(\exists T)$ and (T !) require a parallel evaluation of their argument. For instance, to compute $(\exists T)$ ($\underline{or}$ PQ), assuming that P and Q are programs, we cannot simplify ($\underline{or}$ PQ) into either P or Q, but we must compute P and Q in parallel until we know that the set of all possible computations starting from ($\underline{or}$ PQ) contains $\underline{tt}$ or it is empty or it is just $\{\underline{ff}\}$ (*see* rules 9, IV, V).

We have used, in connection with $(\exists T)$ and $(T!)$, the auxiliary relation $\rightsquigarrow$; its purpose is to make our rewriting system "almost monogenic", *see* lemma 4.1.1, and to avoid defining $\to$ in terms of $\overset{*}{\to}$.

The relation $\to$ is defined by the following rules:

1) $\underline{or}\ MN \to M$; $\quad \underline{or}\ MN \to N$;

2) $\begin{cases} (+1)\underline{n} \to \underline{n}+1; \quad (-1)\underline{n+1} \to \underline{n}; \quad (-1)\underline{0} \to \underline{e}; \\ (+1)\underline{e} \to \underline{e}; \quad (-1)\underline{e} \to \underline{e}; \quad (+1)\ \underline{\Omega}_\iota \to \underline{\Omega}_\iota; \quad (-1)\ \underline{\Omega}_\iota \to \underline{\Omega}_\iota; \end{cases}$

3) $Z\,\underline{0} \to \underline{tt}$; $\quad Z\,\underline{n+1} \to \underline{ff}$; $\quad Z\,\underline{e} \to \underline{ff}$; $\quad Z\,\underline{\Omega}_\iota \to \underline{\Omega}_0$;

4) $E\,\underline{e} \to \underline{tt}$; $\quad E\,\underline{n} \to \underline{ff}$; $\quad E\,\underline{\Omega}_\iota \to \underline{\Omega}_o$;

5) $If\ \underline{ff}\ MN \to N$; $\quad \text{IF}\ \underline{tt}\ MN \to M$; $\quad IF\,\underline{\Omega}_o M_\varkappa N_\varkappa \to \underline{\Omega}_\varkappa$;

6) $YM \to M(YM)$;

7) $(\lambda XM)N \to M[N/X]$;

8 *a*) $$\frac{M \to M'}{MN \to M'N};$$

8 *b*) $$\frac{N \to N';\, f \in \{(+1), (-1), Z, E, IF\}}{f\,N \to f\,N'};$$

9 *a*) $$\frac{M \in p_1}{(\exists T)M \to \underline{tt}},$$

p_1 given by: $\underline{tt} \in p_1$; if $N \in p_1$, then $\underline{or}_o NN'$, $\underline{or}_o N' N \in p_1$, for any N' of type o;

9 *b*) $$\frac{M \in p_2}{(\exists T) M \to \underline{ff}},$$

p_2 given by : $\underline{ff}, \emptyset_o \in p_2$; $N, N' \in p_2 \Rightarrow \underline{or}_o NN' \in p_2$;

9 *c*) $$\frac{M \in p_3}{(T!) M \to \underline{tt}},$$

p_3 given by: $\underline{tt} \in p_3$; $N, N' \in p_3 \Rightarrow \underline{or}_o NN'$ $\underline{or}$ ${}_o\underline{\Omega} N$, $\underline{or}$ ${}_o N \underline{\Omega} \in p_3$;

9 *d*) $$\frac{M \in p_4}{(T!) M \to \underline{ff}},$$

where $p_4 = p'_4 \cup p''_4$ and p'_4, p''_4 are given by: $\underline{ff} \in p'_4$; $N \in p'_4 \Rightarrow \underline{or}$ ${}_o NN'$, $\underline{or}_o N' N \in p'_4$, for any N'; $\Omega_0 \in p''_4$; $N, N' \in p''_4$ $\underline{or}$ ${}_o N N' \in p''_4$;

10 *a*) $$\frac{M \notin p_1 \cup p_2,\ M \rightsquigarrow M'}{(\exists T) M \rightsquigarrow (\exists T) M'};$$

10 *b*) $$\frac{M \notin p_3 \cup p_4,\ M \rightsquigarrow M'}{(T\,!) M \to (T\,!) M'};$$

where $\rightsquigarrow$ is given by all the above rules, but for 1) and 8 *b*), and the following rules:

(I) $f(\underline{or}\ MN) \rightsquigarrow \underline{or}\ (fM)\ (fN), \qquad f \in \{(+1), (-1), Z, E\}$;

(II) $IF\ (\underline{or}\ MN)\ M' M'' \rightsquigarrow \underline{or}\ (IFMM'M'')\ (IFNM'M'')$;

(III) $$\frac{M \neq \underline{or}\ M_1 M_2,\ M \rightsquigarrow M'}{f M \rightsquigarrow f M'}, \qquad f \in \{(+1), (-1), Z, E, IF\};$$

(IV) $$\frac{M \rightsquigarrow M',\ N \rightsquigarrow N'}{\underline{or}\ NM \rightsquigarrow \underline{or}\ M' N'};$$

(V *a*) $$\frac{M \rightsquigarrow M',\ \underline{c} \text{ ground constant}}{\underline{or}\ M \underline{c} \rightsquigarrow \underline{or}\ M' \underline{c}};$$

(V *b*) $$\frac{M \rightsquigarrow M',\ \underline{c} \text{ ground constant}}{\underline{or}\ \underline{c} M \rightsquigarrow \underline{or}\ \underline{c} M'}.$$

LEMMA 4.1.1: *To any program P we can associate a* unique *tree, the* computation tree for *P*, *with the following properties:*

a) *the root is labeled by P;*

b) *if* α *is a node labeled by Q and* $Q \to Q'$, *without using rule* 1, *then* α *has a unique son labeled by* Q'*;*

c) if α is a node labeled by $Q=\mathscr{C}\,[\underline{or}\ Q_1\,Q_2]$, where $\mathscr{C}\,[\ \ ]$ is a single hole context, and we reduce Q by reducing $\underline{or}\ Q_1\,Q_2$, using rule 1, then α has exactly two sons, $\alpha 1$ and $\alpha 2$, and αi is labeled by $\mathscr{C}[Q_i]$;

d) α is labeled by a ground constant iff α is a leaf.

Proof: It is clear that the thesis of the lemma is equivalent to say that for every program P, ($\circ$) below is true; ($\circ$) one and only one of the following conditions holds for P:

(i) P is a constant;

(ii) a unique program P' exists s.t. $P \rightarrow P'$, without using rule 1);

(iii) two programs P' and P'', uniquely determined, exist s.t. $P \rightarrow P'$ and $P \rightarrow P''$, using rule 1).

To show this, following the style in [8], we prove that any term M in NDL2 (hence in NDL1) verifies property Ψ (we say that M is Ψ) defined by *a*), *b*) and *c*) below.

a) a program P is Ψ iff ($\circ$) above and ($\circ\circ$) below are true; ($\circ\circ$) one and only one of the following conditions holds for P:

(iv) P is in *simple form*, i. e. it is composed of ground constants and $\underline{or}$'s only;

(v) a unique program P' exists s. t. $P \rightsquigarrow P'$;

b) a closed term $M_{\sigma\rightarrow\tau}$ is Ψ iff, for every closed term N_σ which is Ψ, $(M_{\sigma\rightarrow\tau}\,N_\sigma)$ is Ψ;

c) a term M with free variables $\alpha_1, \ldots, \alpha_n$ of type $\sigma_1, \ldots, \sigma_n$ respectively, is Ψ iff, for every closed terms $N_{\sigma_1}, \ldots, N_{\sigma_n}$ which are Ψ, $M\,[N_\sigma/\alpha_1, \ldots, N_\sigma/\alpha_n]$ is Ψ. Any term like the one above will be called a *Ψ-closure of M*.

The proof is by structural induction and by cases; we develop here only a few of them.

$M=\underline{or}_\chi$. Let P be $\underline{or}_\chi\,N_1\,N_2\,(N_1,\,N_2$ closed and Ψ). For (o) we have that only (iii) holds for P; indeed only rule 1) applies to P (for instance, rule 8 *a* cannot be used as no reduction applies to $\underline{or}_\chi\,N_1$). For ($\circ\circ$) we have: either P is in normal form, or just one out of rules IV, V *a*, V *b* applies to P.

$M=(-1)$. Let P be $(-1)\,N$, where N is closed (hence it is a program) and Ψ. By hypothesis ($\circ$) is true for N, so we obtain: if (i) holds for N, then (ii) is the only condition true for P (we use rule 2); if (ii) holds for N then (ii) holds for P (we rule 8 *b*); otherwise, (iii) holds for both P and N (still using rule 8 *b*). For ($\circ\circ$), suppose (iv) holds for N; then: if $N=\underline{n}$, we can apply rule 2), otherwise $N=\underline{or}_\iota\,N_1\,N_2\,(N_1,\,N_2$ simple forms) and we can apply to P rule (I) only; in both cases (v) is true for P. Otherwise, suppose that (v) holds for N; then either $N=\underline{or}_\iota\,N_1\,N_2$ and we apply rule (I) to P, or $N\neq\underline{or}_\iota\,N_1\,N_2$ and there is a unique N' s. t. $N \rightsquigarrow N'$, hence $P \rightsquigarrow (-1)\,N'$ and this is the only possibility.

$M=(\exists T)$. Let P be $(\exists T)N$, N program and Ψ; we can prove (∘) and (∘∘) together. Either $N \in p_1 \cup p_2$ — *see* rules 9 — and then $P \to \underline{c}$, $P \rightsquigarrow \underline{c}$, where $\underline{c} \in \{\underline{tt}, \underline{ff}\}$, or $N \notin p_1 \cup p_2$ and therefore N cannot be a simple form and (v) must hold for it so $N \rightsquigarrow N'$, $P \rightsquigarrow (\exists T)N'$, $P \to (\exists T)N'$ and (v) holds for P.

$M=Y$. Let $P=YN_1 \ldots N_n$ be a program and $N_1, \ldots, N_n$ be Ψ. By inspection of the rules defining $\to$ and $\rightsquigarrow$ which concern application and Y we see that only (ii) and (v) hold P; precisely: $P \to P'$ and $P \rightsquigarrow P'$, where $P'=N_1(YN_1)N_2 \ldots N_n$.

$M=M_1 M_2$, with M_1 and M_2 which are Ψ. If M is closed, so are M_1 and M_2; hence M is Ψ as M_1 is such (by definition). If M is open, let $\overline{M}$ be any Ψ-closure of M; $\overline{M}$ can be decomposed as $\overline{M}_1 \overline{M}_2$, where $\overline{M}_j$ is a Ψ-closure of $M_j, j=1, 2$. By definition: $\overline{M}_1$ and $\overline{M}_2$ are Ψ as M_1 and M_2 are such; hence $\overline{M}$ is Ψ and so is M.

$M=\lambda X^\sigma M_1$, with M_1 which is Ψ. If M is closed, let $P=MN_1 \ldots N_n$ be a program where $N_1, \ldots, N_n$ are Ψ; then by inspection of the rules for $\to$ and $\rightsquigarrow$ we see that the only possibilities are: $P \to P'$ and $P \rightsquigarrow P'$, where $P'=M_1[N_1/X^\sigma]N_2 \ldots N_n$. If M is open, any of its Ψ-closures has the form $\lambda X^\sigma M'_1$, where M'_1 is such that if N_σ is closed and Ψ, then $M'_1[N_\sigma/X^\sigma]$ is a Ψ-closure of M_1. Hence M'_1 is Ψ, so we can argue as above. □

4.2. Operational preorder.

If we denote by $\overset{*}{\to}$ the reflexive and transitive closure of $\to$, the above lemma allows us to define **the set of (possible) results of a program** P, **Eval** (P), **by:**

$$\underline{c} \in \mathrm{Eval}(P) \quad \text{iff} \quad P \overset{*}{\to} \underline{c};$$

$\infty \in \mathrm{Eval}(P)$ iff there is an infinite sequence $P=P_0, P_1, \ldots, P_n, \ldots$ s.t. $P_i \to P_{i+1}$ (we say that P may diverge, or simply diverges). Notice that ∞ is a new symbol.

We have suppressed subscripts 1 and 2 in $\to$ and Eval; we do the same for $\underset{\sim}{\sqsubset}_{\text{op}}$ and $\sqsubseteq_{\text{op}}$ below.

Let P, Q be programs and M, N be terms (of the same type), the **operational pre-order** for them is given by:

$$P \sqsubseteq_{\text{op}} Q \quad \text{iff} \quad \begin{cases} \text{either } \infty \notin \mathrm{Eval}(P) \text{ and } \mathrm{Eval}(P) \setminus \{\underline{\Omega}\} = \mathrm{Eval}(Q) \setminus \{\underline{\Omega}\}, \\ \text{or } \infty \in \mathrm{Eval}(P) \text{ and } \mathrm{Eval}(P) \setminus \{\underline{\Omega}, \infty\} \subseteq \mathrm{Eval}(Q) \setminus \{\underline{\Omega}\}, \end{cases}$$

$$M \underset{\sim}{\sqsubset}_{\text{op}} N \quad \text{iff} \quad \mathscr{C}[M] \sqsubseteq_{\text{op}} \mathscr{C}[N]$$

for all contexts $\mathscr{C}[\ ]$ s.t. $\mathscr{C}[M]$ and $\mathscr{C}[N]$ are programs.

We denote by $\equiv_{\text{op}}$ the equivalence associated with $\underset{\sim}{\sqsubset}_{\text{op}}$.

5. EXTRINSIC FULL ABSTRACTION

5.1. Main theorem

What follows applies uniformly to NDL1 and NDL2, together with their models $\mathcal{M}_1$ and $\mathcal{M}_2$, therefore we shall drop subscripts 1 and 2, as we did in Eval, and write $\mathcal{M}$ instead of $\mathcal{M}_J$ and also $[\ \ ]$ instead of $\mathcal{M}_J[\ \ ]$.

THEOREM 5.1.1: *The two preorders* $\underset{\sim}{\sqsubset}_{\mathrm{op}}$ *and* $\sqsubseteq_{\mathcal{M}}$ *coïncide, i. e. the model is fully abstract w.r.t. the operational semantics* ([4]; *in this sense it is extrinsicly fully abstract*).

The proof of this theorem falls into three parts: first we show that the denotational and the operational semantics are equivalent on finite programs, then we extend this equivalence to all programs and finally we easily derive the extrinsic full abstraction from the intrinsic one.

We point out that the last two parts rely on very general properties of $\rightarrow$ and $[\ \ ]$; it is the first part only which is specific to the language (s) we consider, namely one has to prove that the δ-rules which characterize $\rightarrow$ are "correct" with respect to the interpretation of first order primitives.

5.2. Equivalence of the two semantics on finite programs

We recall that the definition of finite term has been given in Section 2.3. For finite terms, we can define a relation $\underset{F}{\rightarrow}$, as follows:

a) rules 1) through 9 *d*) in the definition of $\rightarrow$, *but for rule* 6) (about *Y*);

b) the following rules (we call them Ω-rules):

$$(+1)\Omega_\iota \underset{F}{\rightarrow} \Omega_\iota;\ (-1)\Omega_\iota \underset{F}{\rightarrow} \Omega_\iota;$$

$$f\,\Omega_\iota \underset{F}{\rightarrow} \Omega_o,\ f\in\{E, Z\};\ IF_\varkappa\,\Omega_o\,MN \underset{F}{\rightarrow} \Omega_\varkappa;$$

$$\frac{M\in p_5}{(\exists\, T)M \underset{F}{\rightarrow} \Omega_o},\qquad \frac{M\in p_6}{(T\,!)M \underset{F}{\rightarrow} \Omega_o},$$

where p_5, p_6 are given by:

- $\Omega_o \in p_5$; $N\in p_5,\ N'\in p_2 \Rightarrow \underline{or}_o NN',\ \underline{or}_o N'N \in p_5$;
- $\Omega_o \in p_6$; $N\in p_6,\ N'\in p_3 \Rightarrow \underline{or}_o NN',\ \underline{or}_o N'N \in p_6$;

c) the following rules, which replace 10 *a*) and 10 *b*):

$$\frac{M\notin p_1\cup p_2\cup p_5,\ M \underset{F}{\rightsquigarrow} M'}{(\exists\, T)M \underset{F}{\rightarrow} (\exists\, T)M'};\qquad \frac{M\notin p_3\cup p_4\cup p_6,\ M \underset{F}{\rightsquigarrow} M'}{(T!)M \underset{F}{\rightarrow} (T!)M'};$$

where $\underset{F}{\rightsquigarrow}$ is given by: all the rules for $\underset{F}{\rightarrow}$, but for 1) and 8 *b*); rules (I) to (V) given for $\rightsquigarrow$; the rules (VI *a*) and (VI *b*) obtained from rules (V *a*) and (V *b*) by replacing $\underline{c}$ with $\Omega_\varkappa$.

LEMMA 5.2.1: *Every finite program admits, w.r.t.* $\underset{F}{\rightarrow}$, *a computation tree (i.e. a unique tree satisfying conditions a), b), c) of lemma* 4.1.1 *and d'*: α *is labeled by a constant or* $\Omega_\varkappa$ *iff it is a leaf) which is* finite.

Hint to the proof. The proof is similar to that of lemma 4.1.1; one uses property Φ whose definition on programs is: a program P is Φ iff the thesis of the lemma is true for P and moreover P admits a (unique) finite computation sequence w.r.t. $\underset{F}{\rightsquigarrow}$. $\square$

If P is a finite program, $\mathrm{Eval}^F(P)$ is defined by:

$$\underline{c} \in \mathrm{Eval}^F(P) \quad \text{iff } P \underset{F}{\overset{*}{\rightarrow}} \underline{c}; \qquad \Omega_\varkappa \in \mathrm{Eval}^F(P) \quad \text{iff } P \underset{F}{\overset{*}{\rightarrow}} \Omega_\varkappa.$$

PROPOSITION 5.2.2: *For any finite program P we have:*

$$c \in [P](\perp) \quad \textit{iff } \underline{c} \in \mathrm{Eval}^F(P), \quad \textit{for } \underline{c} \neq \underline{\Omega}_\varkappa;$$
$$\perp \in [P](\perp) \quad \textit{iff } \Omega_\varkappa \in \mathrm{Eval}^F(P).$$

In proving this proposition we use an intermediate step (and assume that the reader is familiar with Milner's paper [5]).

We say that a term is in *normal form* iff it is composed of ground and first order constant symbols and ground Ω's only; we denote n.f.'s by w's. Moreover, let $\hat{\mathscr{I}}$ be the homomorphic extension of $\mathscr{I}$ to n.f.'s, setting $\hat{\mathscr{I}}(\Omega_\varkappa) = \perp$.

Given the nature of Milner model, the proof of the proposition above consists mainly in showing that, for finite programs, our operational semantics is equivalent to one in which they are reduced to their (unique) normal form, using normal reduction (i.e. left-most β-reduction only). The following lemma is the main step to prove this fact.

LEMMA 5.2.3: *If P is a finite program, then:*

a) *if* $P \underset{F}{\rightarrow} P'$ *without using rule* 1, *let* w, w' *be the normal forms to which P and P'*, *respectively, reduce by normal reduction; then* $\hat{\mathscr{I}}(w) = \hat{\mathscr{I}}(w')$;

a') *the analogue of a) if* $P \underset{F}{\rightsquigarrow} P'$;

b) *if* $P \underset{F}{\rightarrow} P'$ *and* $P \underset{F}{\rightarrow} P''$ *(using rule* 1) *and* w, w', w'' *are the n.f.'s for* P, P', P'', *then* $\hat{\mathscr{I}}(w) = \hat{\mathscr{I}}(w') \cup \hat{\mathscr{I}}(w'')$.

Proof (following once more the style in [8]): Let θ be the predicate on finite terms defined by:

(i) if M is a finite program then M is θ (i. e. $M \in \theta$) iff *a*), *a'*), *b*) above hold for M;

(ii) if $M_{\sigma \to \tau}$ is closed then $M_{\sigma \to \tau}$ is θ iff so is $(M_{\sigma \to \tau} N_\sigma)$ for every closed N_σ which is θ;

(iii) if M is open with free variables $\alpha_1, \ldots, \alpha_n$ of type $\sigma_1, \ldots, \sigma_n$ then M is θ iff so is $M[N_1/\alpha_1, \ldots, N_n/\alpha_n]$ for every closed $N_1, \ldots, N_n$, of type $\sigma_1, \ldots, \sigma_n$, which are θ.

We show that any finite M is θ; here we detail a few cases only..

$\underline{M = f \in \{(+1), (-1), Z, E\}}$: Let P be $f\,N$, for N closed (hence program) and θ. We have to prove that *a*), *a'*), *b*) hold for P.

Part *a*) is easily proved (two cases: $N = \underline{c}$, $N \underset{F}{\to} N'$).

Part *a'*). If $N \neq \underline{or}\, N_1 N_2$, then either $N = \underline{c}$, or $N \underset{E}{\leadsto} N'$; in both cases, as N is θ and it is a program, *a'*) is clearly true for P.

If $N = \underline{or}\ N_1 N_2$, then $f\,N \underset{F}{\leadsto} \underline{or}\ (f\,N_1)\ (f\,N_2)$; now: $f\,(\underline{or}\ N_1 N_2) \overset{*}{\underset{\beta}{\to}} f\,(\underline{or}\ w_1 w_2) = w$ and $\underline{or}\ (f\,N_1)\ (f\,N_2) \overset{*}{\underset{\beta}{\to}} \underline{or}\ (fw_1)\ (fw_2) = w'$; clearly $\mathscr{I}\ (w) = \hat{\mathscr{I}}\ (w')$.

Part *b*). $[f\,N \underset{F}{\to} P_1$ and $f\,N \underset{F}{\to} P_2]$ iff $[N \underset{F}{\to} N_1$ and $N \underset{F}{\to} N_2$ and $P_i = f\,N_i]$; hence as *b*) holds for N it clearly holds for $f\,N$.

$\underline{M = (\exists\, T)}$: Let P be $(\exists\, T)\ N$, N program and θ.

Part *a*). By lemma 5.2.1 we have only two cases:

a 1) $N \in p_j$, $j = 1, 2, 5$; then *a*) is proved by induction on the number of $\underline{or}$'s in N;

a 2) $N \notin p_1 \cup p_2 \cup p_5$ and $N \underset{F}{\leadsto} N'$ and $P \to P' = (\exists\, T)\, N'$; we easily get *a*) for P, as *a'*) holds for N.

This proves also part *a'*), as $P \underset{F}{\leadsto} P'$ iff $P \underset{F}{\to} P'$.

Part *b*) is trivially true as rule 1) does not apply to P.

For the cases $M = M_1 M_2$ and $M = \lambda\, X M_1$, we argue as we did in the proof of lemma 4.1.1 □

Proof of Prop. 5.2.2: By the definition of Milner's model we have: $[P]\,(\perp) = [w]\,(\perp)$, where $P \overset{*}{\underset{\beta}{\to}} w$; moreover if $\varphi_\varkappa$ is the isomorphism between $D''_\varkappa$ and $D_\varkappa$ (remember we have suppressed subscripts 1 and 2), then $\varphi\ ([w]\,(\perp)) = \hat{\mathscr{I}}\,(w)$.

From lemma 5.2.3 we have: $\hat{\mathcal{I}}(w) = \cup \{\mathcal{I}(\alpha) \mid \alpha \in \text{Eval}(P)\}$ (finite union), which concludes the proof. Notice that a main point in the proof is that both w and a finite computation tree for P exist and are unique. □

5.3. Equivalence of the two semantics on (general) programs

To prove the analogue of proposition 5.2.1 for arbitrary programs, we need two lemmas. Let P be any program and recall that $P^{(n)} = P[Y^{(n)}/Y]$.

LEMMA 5.3.1: $[P](\perp) = \sqcup \{[P^{(n)}](\perp), n \geqq 0\}$.

Proof: This fact is a consequence of the general property of models: $[Y](\perp) = \sqcup \{[Y^{(n)}](\perp), n \geqq 0\}$. □

LEMMA 5.3.2: $P \overset{*}{\rightarrow} \underline{c}$ *iff* $\exists\, \bar{n}$ s.t. $P^{(n)} \underset{F}{\overset{*}{\rightarrow}} \underline{c}, \forall n > \bar{n}$; *P diverges iff* $P^{(n)} \underset{F}{\overset{*}{\rightarrow}} \Omega_\chi, \forall n \geqq 0$.

The proof of this lemma requires a few steps. Let $\leqq$ be the least relation between terms [8] s. t.:

(i) $\Omega_\sigma \leqq M_\sigma$ and $Y_\sigma^{(n)} \leqq Y_\sigma$, for all σ, $n \geqq 0$;

(ii) $M \leqq M$;

(iii) $M_{\sigma \to \tau} \leqq M'_{\sigma \to \tau}$,
$N_\sigma \leqq N'_\sigma \Rightarrow \lambda X N_\sigma \leqq \lambda X N'_\sigma$ and $M_{\sigma \to \tau} N_\sigma \leqq M'_{\sigma \to \tau} N'_\sigma$.

LEMMA 5.3.3 [8]: *If M is finite and* $M \leqq N$, *then:*

a) $M \underset{F}{\rightarrow} M'$ *implies: either* $M' \leqq N$, *or* $N \rightarrow N'$ *and* $M' \leqq N'$;

a') *the same as a) replacing* $\underset{F}{\rightarrow}$ *and* $\rightarrow$ *by* $\underset{F}{\rightsquigarrow}$;

b) $N \rightarrow N'$ *implies: either M is irreducible and* $M \leqq N'$, *or* $M \underset{F}{\rightarrow} M' \leqq N'$;

b°) *the same as b) replacing* $\underset{F}{\rightarrow}$ *and* $\rightarrow$ *by* $\underset{F}{\rightsquigarrow}$ *and* $\rightsquigarrow$.

We omit the proof which is rather tedious and it is done by cases (corresponding to the definition of $\leqq, \rightarrow, \rightsquigarrow, \underset{F}{\rightarrow}, \underset{F}{\rightsquigarrow}$). Notice that *a')* and *b')* are just auxiliary to the proof of *a)* and *b)*. □

COROLLARY 5.3.4: *If P is any program, then:*

α) *if* $P^{(n)} \underset{F}{\overset{*}{\rightarrow}} \underline{c}$, *some n, then* $P \overset{*}{\rightarrow} \underline{c}$;

β) *if P diverges, then* $P^{(n)} \underset{F}{\overset{*}{\rightarrow}} \Omega_\chi, \forall n$.

Proof: Part α) derives immediately from part *a*) of lemma 5.3.3 and the definition of $\leqq$ ($\underline{c} \leqq Q \Rightarrow Q = \underline{c}$); part β) is a consequence of part *b*) of the lemma, the def. of $\leqq$ and the fact that all computations from $P^{(n)}$ terminate. □

Proof of lemma 5.3.2: The if part of *a*) and the only if part of *b*) have already been proved (5.3.4).

If part of b): The derivation $P^{(n)} \underset{F}{\overset{*}{\rightarrow}} \Omega$ can clearly be decomposed as: $P^{(n)} = Q_0 \underset{F}{\rightarrow} Q_1 \ldots \underset{F}{\rightarrow} Q_n \underset{F}{\overset{*}{\rightarrow}} \Omega$, where $Q_j \underset{F}{\rightarrow} Q_{j+1}$ without using Ω-rules. Therefore there is a "corresponding" (segment of) derivation $P = Q'_0 \rightarrow \ldots \rightarrow Q'_n$, where Q'_j is obtained from Q_j by replacing all $Y_\sigma^{(m)}$'s with Y_σ's. Hence the computation tree for P has arbitrarily long branches; by König's lemma this implies it has an infinite branch.

Only if part of a). In the derivation $P \overset{*}{\rightarrow} \underline{c}$, rule 6 (about Y) is used a finite number of times, say $\bar{n}$. It is clear that there is a "corresponding" derivation $P^{(n)} \underset{F}{\overset{*}{\rightarrow}} \underline{c}$, for all $n \geqq \bar{n}$. □

THEOREM 5.3.5: $c \in [P]\ (\perp)$ *iff* $\underline{c} \in Eval\ (P)$, *for* $\underline{c} \neq \underline{\Omega}$; $\perp \in [P]\ (\perp)$ *iff* $\infty \in Eval(P)$.

The proof follows easily from lemma 5.3.1 (and the definition of the order between subsets) and from lemma 5.3.2. □

5.4. Proof of the main theorem

We have to show that, for any two terms M, N of the same type, $M \sqsubseteq_{\mathcal{M}} N$ iff $M \underset{\sim}{\sqsubset}_{\text{op}} N$. Indeed:

$M \sqsubseteq_{\mathcal{M}} N$ iff $\mathscr{C}[M] \sqsubseteq_{\mathcal{M}} \mathscr{C}[N]$ for all $\mathscr{C}[\ \ ]$ s. t. $\mathscr{C}[M]$ and $\mathscr{C}[N]$ are programs (by intrinsic full abstraction of $\mathcal{M}$) iff $\mathscr{C}[M] \sqsubseteq_{\text{op}} \mathscr{C}[N]$ for all $\mathscr{C}[\ \ ]$ s. t. $\mathscr{C}[M]$ and $\mathscr{C}[N]$ are programs (by theorem 5.3.5 and the definitions of $\sqsubseteq_{\text{op}}$ and of the order in powerdomains) iff $M \sqsubseteq_{\text{op}} N$. □

ACKNOWLEDGEMENTS

The authors wish to thank G. Plotkin for showing them his techniques of logical relations to prove non definability [6, 9].

REFERENCES

1. E. ASTESIANO and G. COSTA, *Sharing in Nondeterminism*, Proc. of the 6th I.C.A.L.P., Graz, 1979, Lecture Notes in Comput. Sc., Springer, Berlin, Vol. 71, 1979, pp. 1-15.
2. G. BERRY, *Stable Models of Typed λ-calculi*, Proc. of the 5th I.C.A.L.P., Udine, 1978, Lecture Notes in Comput. Sc., Springer, Berlin, Vol. 62, 1978, pp. 72-89.
3. M. HENNESSY and E. A. ASHCROFT, *The Semantics of Nondeterminism*, Proc. of the 3rd I.C.A.L.P., Edinburg, 1976, Edinburg University Press, 1976, pp. 478-493.
4. R. MILNER, *Processes, a Mathematical Model for Computing Agents*, Logic Colloquium '73, Studies in Logic and the Foundat. of Math., North Holland-American Elsevier, Vol. 80, 1975, pp. 157-174.
5. R. MILNER, *Fully Abstract Models of Typed λ-Calculi*, Theoret. Comput. Sc., Vol. 4, 1977, pp. 1-22.
6. G. PLOTKIN, *Lambda Definability and Logical Relations*, Memo SAI-RM-4, School of Artif. Intell., Edinburg, 1973.
7. G. PLOTKIN, *A Powerdomain Construction*, S.I.A.M. J. Comput., Vol. 5, 1976, pp. 453-487.
8. G. PLOTKIN, *LCF as a Programming Language*, Theoret. Comput. Sc., Vol. 5, 1977, pp. 223-255.
9. G. PLOTKIN, Personal communication, 1978.
10. M. B. SMYTH, *Power Domains*, J. Comput. System Sc., Vol. 16, 1978, pp. 23-36.

APPENDIX

Here, we specialize to our case a method for proving undefinability due to Plotkin [6, 9], and use it to show that the natural extension to sets of the parallel disjunction is not definable in NDL0 and NDL1 (lemma A.1) and that finite projections are not definable in NDL0 (lemma A.2).

Let $D_\varkappa$ be $D_\varkappa^J$, $J=0, 1, 2$, $\varkappa = o, \iota$; (we keep J fixed) and $D_{\sigma\to\tau}$ be $[D_\sigma \to D_\tau]$. For a fixed n, $n \geqq 1$, consider a family of relations $R_\sigma \subseteq D_\sigma^n$, such that:

(i) $\langle A, \ldots, A\rangle \in R_\varkappa$, for any finite A in $D_\varkappa$;

(ii) $\langle \mathscr{I}_J(f), \ldots, \mathscr{I}_J(f)\rangle \in R_{\sigma(f)}$, for every first order primitive symbol f [such as $(+1)$, <u>or</u>, ...] of type $\sigma(f)$, in NDL J;

(iii) $\langle f_1, \ldots, f_n\rangle \in R_{\sigma\to\tau}$ iff $\langle f_1(x_1), \ldots, f_n(x_n)\rangle \in R_\tau$, for every $\langle x_1, \ldots, x_n\rangle$ in R_σ.

It is easy to show that $\langle \mathscr{M}_J\,[M](\perp), \ldots, \mathscr{M}_J\,[M](\perp)\rangle \in R_\sigma$, for all *finite* closed M, of type σ, in NDL J.

Consider $\bar{\varphi} \in D_{\bar{\sigma}}$, where $\bar{\sigma} = \varkappa_1 \times \ldots \times \varkappa_r \to \varkappa$ (i. e. $\bar{\varphi}$ is first order), and suppose that for some finite A_{ij}'s and B_i's, condition (∘) below is true when $\varphi = \bar{\varphi}$.

(∘) $\{ \varphi (A_{i1}, \ldots, A_{ir}) = B_i,\ 1 \leqq i \leqq n.$

If now a family of relations $\{ R_\sigma,\ \sigma \in T \}$ exists which satisfies (i), (ii), (iii) and such that:

(iv) $\langle A_{1j}, \ldots, A_{nj} \rangle \in R_{\varkappa_j},\ 1 \leqq j \leqq r,$

(v) $\langle B_1, \ldots, B_n \rangle \in R_{\bar{\varkappa}}.$

then $\bar{\varphi}$ is not definable in NDL J. Indeed, if it was definable by a term M, we could find $q \geqq 0$ such that $\mathscr{M}_J\ [M^{(q)}](\perp)$ verifies (∘), in contradiction with the definition and properties of $\{ R_\sigma \}$ and conditions (iv) and (v).

A (minimal) family of relations satisfying (i) through (iv) can be defined (constructively for ground types) as follows:

$$(\star) \qquad R_\varkappa = \bigcup_{m \geqq 0} U_\varkappa^m,$$

where

$$U_\varkappa^0 = \{ \langle A, \ldots, A \rangle \mid A \text{ is finite in } D_\varkappa \} \cup \{ \langle A_{1j}, \ldots, A_{nj} \rangle \mid \varkappa_j = \varkappa,\ 1 \leqq j \leqq r \};$$

$$U_\varkappa^{m+1} = U_\varkappa^m \cup \{ \langle \tilde{f}(x_1^1, \ldots, x_1^m), \ldots, \tilde{f}(x_n^1, \ldots, x_n^m) \rangle \mid \tilde{f} = \mathscr{I}_J(\mathrm{f}),$$

f primitive symbol of type

$$\varkappa_1' \times \ldots \times \varkappa_m' \to \varkappa,\ \langle x_1^i, \ldots, x_n^i \rangle \in U_{\varkappa_i'}^m,\ 1 \leqq i \leqq m \}.$$

(We say that $R_\varkappa$ is the *closure of* $U_\varkappa^0$ *under primitive functions*.)

($\star\star$) $R_{\sigma \to \tau}$ is such that $\langle f_1, \ldots, f_n \rangle \in R_{\sigma \to \tau}$ *iff* for every $\langle x_1, \ldots, x_n \rangle$ in R_σ, $\langle f_1(x_1), \ldots, f_n(x_n) \rangle$ is in R_τ.

Then a proof of non definability consists in selecting a suitable instance of (∘), thus selecting n, and in showing that $\{ R_\sigma, \sigma \in T \}$ "constructed" as above verifies condition (v). The two following lemmas are examples of this technique.

LEMMA A.1: *The natural extension (to subsets) of the parallel disjunction is not definable in* NDL0 *and* NDL1.

(*The parallel disjunction, pd, is the least monotonic function such that:*

$$pd(\perp, tt) = pd(tt, \perp) = tt; \qquad pd(ff, ff) = ff.)$$

Proof: We show that no (finite) function φ such that:

$$\varphi(\{\perp\}, \{tt\}) = \{tt\}; \qquad \varphi(\{tt\}, \{\perp\}) = \{tt\};$$
$$\varphi(\{ff\}, \{ff\}) = \{ff\},$$

can be defined in NDL1 (hence in NDL0).

Let R_o and R_ι be the ternary relations which are the closure under primitive functions of U_o^0 and U_ι^0 below.

$$U_\iota^0 = \{ \langle A, A, A \rangle \mid A \text{ finite in } D_\iota^1 \};$$

$$U_o^0 = \{ \langle A, A, A \rangle \mid A \text{ finite in } D_o^1 \} \cup \{ \langle \{ \perp \}, \{ tt \}, \{ ff \} \rangle, \langle \{ tt \}, \{ \perp \}, \{ ff \} \rangle \}.$$

Then $\langle \{ tt \}, \{ tt \}, \{ ff \} \rangle \notin R_o$; indeed, by straightforward induction on m and by cases, one can prove that: if $\langle A, B, C \rangle \in U_\varkappa^m$ then: either $A = B = C$ or $\perp \in A \cup B \cup C$.

Just to exhibit one case of the proof, let $\langle A, B, C \rangle \in U_o^{m+1}$ and $\langle A_j, B_j, C_j \rangle \in U_o^m$, $j = 1, 2, 3$.

If $\langle A, B, C \rangle = \langle \widehat{\text{cond}}(A_1, A_2, A_3), \widehat{\text{cond}}(B_1, B_2, B_3), \widehat{\text{cond}}(C_1, C_2, C_3) \rangle$ then:

$$\perp \notin A \cup B \cup C \quad \Rightarrow \quad \perp \notin A_1 \cup B_1 \cup C_1 \quad \Rightarrow \quad A_1 = B_1 = C_1 \quad \Rightarrow \quad \langle A, B, C \rangle$$

is either

$$\langle A_2, B_2, C_2 \rangle \quad \text{or} \quad \langle A_3, B_3, C_3 \rangle \quad \text{or} \quad \langle A_2 \cup A_3, B_2 \cup B_3, C_2 \cup C_3 \rangle. \quad \square$$

LEMMA A.2: *No finite function* φ *s. t.* $\varphi(\{0, 1\}) = \{0, 1\}$ *and* $\varphi(\{\bar{n}\}) = \{\perp\}$, *some* $\bar{n} \geqq 2$, *can be defined in* NDL0 (*hence finite projections are not definable in* NDL0).

Proof: Let $\bar{n}$ be a fixed integer, $\bar{n} \geqq 2$, and R_o, R_ι be the binary relations obtained by closure under primitive functions of

$$U_o^0 = \{ \langle A, A \rangle \mid \text{A finite in } D_o^0 \},$$
$$U_\iota^0 = \{ \langle A, A \rangle \mid \text{A finite in } D_\iota^0 \} \cup \{ \langle \{0, 1\}, \{\bar{n}\} \rangle \}.$$

We show that $\langle \{0, 1\}, \{\perp\} \rangle \notin R_\iota$, proving by induction on m that:

a) if $\langle A, B \rangle \in U_o^m$ then:

$$\perp \in B \quad \Rightarrow \quad \perp \in A$$

and

$$b \in B, \qquad b \neq \perp \quad \Rightarrow \quad (b \in A) \vee (\perp \in A);$$

b) if $\langle A, B \rangle \in U_\iota^m$ then:

$$\perp \in B \quad \Rightarrow \quad \perp \in A$$

and

$$p \in B, \qquad p \neq \bot \quad \Rightarrow \quad (p \in A) \vee (\bot \in A) \vee (p \dot{-} \bar{n},\ p \dot{-} \bar{n} + 1 \in A);$$

where $x \dot{-} y = \text{if } x \geqq y \text{ then } x - y \text{ else } \bot$.

We detail three cases of the induction step for *a*) and *b*); notice that both *a*) and *b*) are clearly true for $m = 0$.

Consider $\langle A, B \rangle$ in U_ι^{m+1}; $\langle A', B' \rangle$, $\langle A'', B'' \rangle$ in U_ι^m and $\langle A_o, B_o \rangle$ in U_o^m.

Case $\langle A, B \rangle = \langle \widehat{pred}\,(A'), \widehat{pred}\,(B') \rangle$

If $\bot \in B$ then: either $\bot \in B'$ and then $\bot \in A'$, hence $\bot \in A$; or $0 \in B'$ and then $(0 \in A') \vee (\bot \in A')$, hence $\bot \in A$.

If $p \in B$, $p \neq \bot$, then $p+1 \in B'$ and then we have:

either $(\bot \in A') \vee (p+1 \in A')$;

hence $(\bot \in A) \vee (p \in A)$;

or $p+1 \dot{-} \bar{n},\ p+2 \dot{-} \bar{n} \in A'$;

hence $(p \dot{-} \bar{n},\ p \dot{-} \bar{n} + 1 \in A) \vee (\bot \in A)$.

Case $\langle A, B \rangle = \langle \widehat{cond}\,(A_o, A', A''), \widehat{cond}\,(B_o, B', B'') \rangle$

If $\bot \in B$ then either (α) or (β) or (γ) holds, where:

(α) $\bot \in B_o$; then $\bot \in A_o$, hence $\bot \in A$;

(β) $(tt \in B_o) \wedge (\bot \in B')$; then $((tt \in A_o) \wedge (\bot \in A')) \vee (\bot \in A_o)$, hence $\bot \in A$;

(γ) $(ff \in B_o) \wedge (\bot \in B'')$; then we conclude as in (β).

If $p \in B$, $p \neq \bot$, then either (δ) or (ε) below holds:

(δ) $(tt \in B_o) \wedge (p \in B')$; then one out of ($\delta$1), ($\delta$2), ($\delta$3), ($\delta$4) holds;

(δ1) $\bot \in A_o$; then $\bot \in A$;

(δ2) $(tt \in A_o) \wedge (\bot \in A')$; then $\bot \in A$;

(δ3) $(tt \in A_o) \wedge (p \in A')$; then $p \in A$;

(δ4) $(tt \in A_o) \wedge (p \dot{-} \bar{n},\ p \dot{-} \bar{n} + 1 \in A')$; then $p \dot{-} \bar{n},\ p \dot{-} \bar{n} + 1 \in A$;

(ε) $(ff \in B_o) \wedge (p \in B'')$; then we conclude as in (δ).

Case $\langle A, B \rangle = \langle \widehat{zero}\,(A'), \widehat{zero}\,(B') \rangle$, when now $\langle A, B \rangle$ is in U_o^{m+1}. We have:

– $\bot \in B \Rightarrow \bot \in B' \Rightarrow \bot \in A' \Rightarrow \bot \in A$;

– $tt \in B \Rightarrow 0 \in B' \Rightarrow (0 \in A') \vee (\bot \in A') \Rightarrow (tt \in A) \vee (\bot \in A)$;

– $ff \in B \Rightarrow p \in B',\ p > 0 \Rightarrow$ either (μ) or (ν) holds:

(μ) $(p \in A') \vee (\bot \in A')$; then $(ff \in A) \vee (\bot \in A)$;

(ν) $p \dot{-} \bar{n},\ p \dot{-} \bar{n} + 1 \in A'$; then $(ff \in A) \vee (\bot \in A)$.

The remaining cases are similar. □