A FULLY OBSERVATIONAL MODEL FOR INFINITE BEHAVIOURS OF COMMUNICATING SYSTEMS

Ph. Darondeau *, B. Gamatie*

Abstract *This paper is concerned with the relation of abstraction between operational and observational models for linear time semantics of C.C.S. We construct a compositional and fully abstract model for CCS under infinite (program defined) experiments which give maximal power of separation. The construction is in two stages:*

In the first stage, we use a uniform procedure to translate structural inferential semantics into denotational semantics; terms and operators are interpreted by sets of transition sequences and operators on sets of transition sequences.

In the second stage, we derive the observational model from the operational model through an adequate homomorphism. The morphic images of transition sequences, or observations, are pairs $\langle w,R\rangle$ or $\langle W,\omega\rangle$, where R is a ready set and w (resp W) , is the visible trace of a finite (resp infinite) sequence of transitions.

The observational meanings of programs coincide with the associated set of maximal observations for the order $\langle w,R\cup R'\rangle \subseteq \langle w,R\rangle \subseteq \langle w,\omega\rangle$.

1. INTRODUCTION

This paper is concerned with the relation of abstraction between operational and observational models for 'linear' semantics of 'CCS' transition systems [26] [23]. Observational models mean here 'fully abstract' models for equivalences or preorders induced by (CCS) program defined experiments, such as 'tests' [22] [13,14]. The observational models are naturally homomorphic images of the operational models, but it is not so easy to find out the homomorphism induced by a family of experiments. Indeed, observational models cannot be defined by arbitrary homomorphisms !

A 'concrete' linear model for CCS is the operational model in which programs are rendered by sets of finite and infinite computations. An 'abstract' linear model for CCS is the observational model in which programs are identified if and only if no CCS experiment distinguishes between them. In spite of the effort on models for testing preorders [13,20], that very sensitive model has not yet been worked out, for it has been shown in [10] that some infinite experiments are strictly more sharp than any family of binary tests (finite or infinite). There lays the motivation for the present paper, where we produce the expected model for the full version of pure CCS.

Let us proceed to a rapid review of some existing approaches on the side of abstract models, independently of the homomorphisms which induce them as quotients. First of all, one can distinguish between implicit quotient models and explicit models where explicit representations are dealt with [6] . Further separation may be observed between two classes of explicit models, namely the class of purely order theoretic models and the class of mixed space and order theoretic models. Classic ω-algebraic domains are most commonly used in models of the first class [7,13,19,20,24]. For the second class of models, the order relation is the reverse inclusion between closed subsets in some metric completion of languages, trees or tree-languages [11,18,27], and since the considered spaces have denumerable bases, the resulting complete partial

(*) IRISA - Campus de Beaulieu - F 35042 RENNES CEDEX

orders are again ω-algebraic.

In the framework of explicit models, it is usually assumed that each element or open set of the denumerable base represents a property that may be tested under finite experiments [1] , i.e. under experiments which either succeed in finite time or fail [13]. Therefore, identical meanings are necessarily assigned to CCS programs which differ by their ω-traces[10]. Notice that this situation is less frequent under the assumption of fairness, since fairness intensifies the separation power of tests, but it arises nevertheless. A possible way out is to work within the bounds of 'uniform' concurrency [12] : behaviours are then closed sets, and isomorphic semantics are obtained from infinite streams and from finite observations respectively [11]. In particular, that property holds for the finite state subset of CCS, obtained by forbidding the parallel composition and renaming of open terms[8].

There arises from the above review that no well known method is available for associating CCS programs with their compositional meanings under infinite experiments. We think of experimenting on program u as running a maximal computation from the parallel system (t |C[u]) for some testing program t and program context C. This definition, though adequate to de Nicola and Hennessy's notion of testing, departs from it in the definition of the result of experiments. For us, the result of an experiment is the sequence of transitions of the testing program (along that experiment). On that basis, we equip programs with an 'implementation' preorder as follows: u is an implementation of v if and only if no finite or infinite experiment on u tells that it is not v [9,16,21]. The abstract model which we are looking for is the fully abstract model for this implementation preorder. Moreover, we argue that programs identified in that model must be identified in any other observational model, as soon as it relies on the above stated observation principles.

Our approach towards the abstract model is progressive and rests entirely on the concrete model which we establish in a preliminary stage. The first stage is to discover the homomorphism which induces the abstract model. The second stage is to construct that model. The first stage amounts essentially to transfer the implementation preorder from programs to sets of computations and then to represent the latters by sets of pairs $\langle w,\omega \rangle$ or $\langle w,R \rangle$ where w is a 'trace' and R is a 'ready set'. On that way, we define an ordered algebra of abstract computations and prove that the abstract meaning of a program is the set of maximal representations of its concrete computations. The second stage amounts to supply a fixpoint characterisation of the abstract meanings of programs in the power algebra of abstract computations. Special difficulties are encountered here, for the 'suitable' fixed points are neither least nor greatest nor optimal fixed points. A plausible explanation for that misfit is the apparent contradiction between two opposite requirements imposed by full abstraction : inobservable actions (e.g.τ) must be ignored in the domain of meanings, but they must be accounted for in the fixed point characterization of sets of infinitary traces. To solve the difficulty, we suggest to compute the 'suitable' fixed points as combinations of a least fixed point in the power algebra of abstract computations and of a greatest fixed point in the power algebra of coarse traces (with explicit τ's). That method has the default to split the meanings of open terms in two completely disjoint functions, but full abstraction is obtained at the level of closed terms which are given an ordinary compositional meaning in a single domain. The paper contains nothing about the proof theory for the equivalence in the resulting model. That issue will be considered in forthcoming papers.

2. AN OPERATIONAL SEMANTICS FOR PURE CCS

We present here a slightly augmented version of Milner's 'pure' calculus of communicating systems [13,19,23]. The extensions affect unguarded recursion and non deterministic choice. As regards syntax, programs are the closed terms of a recursive term algebra. As regards semantics, the operational meanings of programs are sets of transition sequences, defined by a logical system a la Plotkin [25].

2.1. A syntax of programs

We assume given a set $\mathcal{X}$ (ranged over by x) of variables, and two disjoint sets Δ and $\overline{\Delta}$ of complementary action names, linked by reciprocal bijections $\overline{\eta}$: $\overline{\eta}(\lambda) = \overline{\lambda}$ and $\overline{\eta}(\overline{\lambda}) = \lambda$. Throughout the paper, we let λ resp. μ resp. υ range over $\Lambda = \Delta \cup \overline{\Delta}$ resp. $\mathcal{M} = \Lambda \cup \{\tau\}$ resp. $\mathcal{N} = \mathcal{M} \cup \{\sigma\}$, where τ ($\notin \Lambda$) is the invisible action and σ ($\notin \mathcal{M}$) is the inaction. We call a renaming function any one-one partial function ρ from $\mathcal{N}$ to $\mathcal{N}$ such that $\rho(\upsilon)=\upsilon$ almost every-where, $\rho(\sigma)=\sigma$, $\rho(\tau)=\tau$, and $\rho(\lambda) = \overline{\eta}(\rho(\overline{\lambda}))$ unless both entities are undefined.

The signature of our recursive term algebra is $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, where the Σ_i (ranged over by op_i) are the following sets of i-ary symbols :

Σ_0 = {NIL},

Σ_1 = $\{\mu / \mu \in \mathcal{M}\} \cup \{\rho / \rho$ is a renaming function$\}$,

Σ_2 = $\{|, +, \oplus\}$.

Binary operators |, +, $\oplus$ are the usual asynchronous composition, external choice and internal choice. The binary operators are used in infixed form, the guarding operators (μ) are prefixed, and the renaming operators (ρ) are postfixed.
Our set of recursive open terms is the set TERM with typical elements t as follows :

$$t = x \mid op_i\ (t_1, \ldots t_i) \mid t\ \underline{wh}\ (x_1 = t_1, \ldots x_n = t_n).$$

In the above line, $(x_1=t_1, \ldots x_n=t_n)$ stands for a 'declaration', i.e.for a function D with finite domain $dom(D)=\{x_1, \ldots x_n\}$ ($\subseteq \mathcal{X}$) and corresponding valuations $D(x_i)= t_i$ ($\in$ TERM).

The operation (.) wh D, when applied to term t, binds the free occurences of the x_i to the t_i in t and in the t_j. Recursive open terms are defined up to the α-conversion of their bound variables. For all terms, we shall assume throughout the paper that any two declarations have disjoint domains and that no conflict arises between free variables and bound variables. Owing to that convention, the set theoretic union (D $\cup$ D') of two declarations occuring in a given term is always a declaration. Terms without free variables (i.e. closed terms) are called programs. The meta variables s, t, u, v will be used to denote general terms, and in particular programs ($\in$ PROG). We are mainly interested in programs and in their computations.

2.2. An operational semantics for programs

A computation is a finite or infinite sequence of transitions $t_i \xrightarrow{v_i} t_{i+1}$ between programs. Transitions are either steps of evolution resulting from visible or invisible actions (λ or τ), or self-transformations of programs with no action involved. In the last case, label σ appears as a witness for the flow of time.

According to Plotkin's method of structural inferential semantics, transitions are defined by a finite family of schemes of axioms and rules of inference for relations $u \xrightarrow{v} v$ in PROG x $\mathbb{N}$ x PROG. The set $\mathcal{T}$ of transitions is the least subset of PROG x $\mathbb{N}$ x PROG closed under logical inference in the following system, stated in Gentzen like style.

AXIOMS

$\mu t \xrightarrow{\mu} t$

$t \oplus u \xrightarrow{\sigma} t \ , \quad t \oplus u \xrightarrow{\sigma} u$

NIL wh D $\xrightarrow{\sigma}$ NIL

x wh D $\xrightarrow{\sigma}$ D(x) wh D for D(x) defined

(μt) wh D $\xrightarrow{\sigma}$ μ(t wh D)

(tρ) wh D $\xrightarrow{\sigma}$ (t wh D) ρ

(t op_2 u) wh D $\xrightarrow{\sigma}$ (t wh D) op_2 (u wh D)

(t wh D) wh D' $\xrightarrow{\sigma}$ t wh (D $\cup$ D')

RULES

$(t \xrightarrow{\mu} t') \Rightarrow (t+u \xrightarrow{\mu} t') \ , \ (u+t \xrightarrow{\mu} t')$

$(t \xrightarrow{\mu} t') \Rightarrow (t|u \xrightarrow{\mu} t'|u) \ , \ (u|t \xrightarrow{\mu} u|t')$

$((t \xrightarrow{\lambda} t') \ , \ (u \xrightarrow{\bar{\lambda}} u')) \Rightarrow (t|u \xrightarrow{\tau} t'|u')$

$(t \xrightarrow{v} t') \Rightarrow (t\rho \xrightarrow{\rho(v)} t'\rho)$ if $\rho(v)$ defined

$(t \xrightarrow{\sigma} t') \Rightarrow (t\ op_2\ u \xrightarrow{\sigma} t'\ op_2\ u), \ (u\ op_2\ t \xrightarrow{\sigma} u\ op_2\ t')$

Except for the axioms of internal choice, all the axioms and rules for σ-transitions are meant to give an operational flavour to the notion of divergence due to unguarded recursivity : infinite sequences of σ-transitions are the characteristic mark of unguarded divergence.

2.3. A concrete linear model

A denotational model where programs are interpreted by sets of infinitary computations can be derived from the above definitions (this is achieved in the full version of the paper). Such a model is a bridge between the operational definition and the observational model which we are looking for, since the latter is nothing but a morphic image of the former. For the sake of conciseness, the presentation of the operational model is reduced here to the necessary description of the corresponding domain of interpretation.

We define $^*\mathcal{T}$ (resp. $^\infty\mathcal{T}$) as the set of sequences ε_t or $(t_i \xrightarrow{v_i} t_{i+1})_{i<\gamma}$ such that $\gamma \in \mathbb{N}$ (resp. $\gamma \in \mathbb{N} \cup \{\omega\}$) and $\forall i < \gamma$, $(t_i \xrightarrow{v_i} t_{i+1}) \in \mathcal{T}$.

The finitary resp. infinitary meanings of programs are given by the following functions, where $t \to \boldsymbol{t}$ means $\boldsymbol{t} = \varepsilon_t$ or $\boldsymbol{t} = (t \xrightarrow{v} t')\boldsymbol{t}'$:

$[\![\ ]\!]^*_{op}$: PROG $\to \mathcal{P}(*\mathcal{T})$: $[\![t]\!]^*_{op} = \{\boldsymbol{t} \in *\mathcal{T} / t \to \boldsymbol{t}\}$,

$[\![\ ]\!]^\infty_{op}$: PROG $\to \mathcal{P}(^\infty\mathcal{T})$: $[\![t]\!]^\infty_{op} = \{\boldsymbol{t} \in {}^\infty\mathcal{T} / t \to \boldsymbol{t}\}$.

In the sequel, we let $\boldsymbol{t}$ range over $^\infty\mathcal{T}$, and we let Q range over $\mathcal{P}(^\infty\mathcal{T})$.

The set $^\infty\mathcal{T}$ may be ordered by the prefix order on sequences ; it may also be considered as a complete metric space, under the metric topology induced by the ultrametric distance of [5]. Limits in the Scott topology induced by the order prefix coincide indeed with limits in that metric topology. Now, for t in PROG, $[\![t]\!]^\infty_{op}$ is prefix closed and topologically closed, and $[\![t]\!]^*_{op}$ is equal to $[\![t]\!]^\infty_{op} \cap *\mathcal{T}$, whence $[\![t]\!]^\infty_{op}$ is the topological closure of $[\![t]\!]^*_{op}$.

The interpretations $\underline{op}_k : \mathcal{P}(*\mathcal{T})^k \to \mathcal{P}(*\mathcal{T})$ and $\overline{op}_k : \mathcal{P}(^\infty\mathcal{T})^k \to \mathcal{P}(^\infty\mathcal{T})$ for the Σ-operators are the union additive extensions of 'elementary' operators $\underline{op}_k : (*\mathcal{T})^k \to \mathcal{P}(*\mathcal{T})$ and $\overline{op}_k : (^\infty\mathcal{T})^k \to \mathcal{P}(^\infty\mathcal{T})$ (so they are continuous w.r.t. $\subseteq$ in the complete lattices $\mathcal{P}(*\mathcal{T})$ and $\mathcal{P}(^\infty\mathcal{T})$).

In the sequel, we make heavy use of the convention that functions and operators are implicitly extended, whenever necessary, along union additive extensions. However there is an exception in the case of the 'funny concatenation' denoted "•", an operation which has a central importance in our model and which is defined as follows : • : $\mathcal{T} \times {}^\infty\mathcal{T} \to \mathcal{P}(^\infty\mathcal{T})$:

$(u\text{-}^{\upsilon}\!\to v) \bullet \varepsilon_t = \{\varepsilon_u, (u\text{-}^{\upsilon}\!\to v)\}$,

$(u\text{-}^{\upsilon}\!\to v) \bullet (t_i\text{-}^{\upsilon_i}\!\to t_{i+1})_{i<\gamma} = \{\varepsilon_u, (u\text{-}^{\upsilon}\!\to v)\} \cup \{(u\text{-}^{\upsilon}\!\to v)(t_i\text{-}^{\upsilon_i}\!\to t_{i+1})_{i<\beta} / (v=t_o, \beta<\gamma)\}$.

For the extended operation • : $\mathcal{T} \times \mathcal{P}(^\infty\mathcal{T}) \to \mathcal{P}(^\infty\mathcal{T})$, and in the special case $Q = \emptyset$, we set : $(u\text{-}^{\upsilon}\!\to v) \bullet Q = \{\varepsilon_u, (u\text{-}^{\upsilon}\!\to v)\}$.

The infinitary operations $\overline{op}_k : (^\infty\mathcal{T})^k \to \mathcal{P}(^\infty\mathcal{T})$ are strongly dependent on the finitary operations $\underline{op}_k : (*\mathcal{T})^k \to \mathcal{P}(*\mathcal{T})$, for they are defined as the continuous extensions of the latter. The precise statement of the dependence is as follows, letting $CL(Q)$ and $Pref(\boldsymbol{t})$ mean respectively the topological closure of Q and the set of finite prefixes of $\boldsymbol{t}$:

$\overline{op}_k(\boldsymbol{t}_1, \ldots \boldsymbol{t}_k) = CL(\underline{op}_k(Pref(\boldsymbol{t}_1), \ldots Pref(\boldsymbol{t}_k)))$.

The finitary operations $\underline{op}_k : (*\mathcal{T})^k \to \mathcal{P}(*\mathcal{T})$ are defined by six families D1-D6 of inductive relations (drawn from the operational definition of (2.2)). These relations are also valid for the infinitary operations $\overline{op}_k$, although they do not define them. For that reason, the underlining or surlining of operator symbols is omitted from the following table. To keep reasonable notations, we let $\boldsymbol{t}_s = (s\text{-}^{\upsilon}\!\to t)\boldsymbol{t}_t$ and $\boldsymbol{t}_u = (u\text{-}^{\upsilon'}\!\to v)\boldsymbol{t}_v$ everywhere in the definitions, with $t \to \boldsymbol{t}_t$ and $v \to \boldsymbol{t}_v$. The notation "[cond, Q_1, Q_2]" reads as : 'if cond then Q_1 else Q_2'.

THE INTERPRETATION OF Σ IN $*\mathcal{T}$ AND $^\infty\mathcal{T}$

D1 NIL = $\{\varepsilon_{NIL}\}$

D2 $\mu(\boldsymbol{t}_t) = (\mu t \text{-}^{\mu}\!\to t) \bullet \boldsymbol{t}_t$

D3 $(\varepsilon_t)\rho = \{\varepsilon_{t\rho}\}$,

$(\boldsymbol{t}_s)\rho = [\upsilon \notin dom(\rho), \{\varepsilon_{s\rho}\}, (s\rho \text{-}^{\rho(\upsilon)}\!\to t\rho) \bullet (\boldsymbol{t}_t)\rho]$

D4 $\varepsilon_t + \varepsilon_v = \{\varepsilon_{t+v}\}$,

$$t_s + \varepsilon_v = [\upsilon \neq \sigma,\ (s+v \xrightarrow{\upsilon} t)\bullet t_t,\ (s+v \xrightarrow{\upsilon} t+v)\bullet(t_t + \varepsilon_v)],$$

$$\varepsilon_t + t_u = [\upsilon' \neq \sigma,\ (t + u \xrightarrow{\upsilon'} v)\bullet t_v,\ (t+u \xrightarrow{\upsilon'} t+v)\bullet(\varepsilon_t + t_v)],$$

$$t_s + t_u = [\upsilon \neq \sigma,\ (s + u \xrightarrow{\upsilon} t)\, t_t,\ (s+u \xrightarrow{\upsilon} t+u)\bullet(t_t + t_u)]$$
$$\cup\ [\upsilon' \neq \sigma, (s + u \xrightarrow{\upsilon'} v)\, t_v,\ (s+u \xrightarrow{\upsilon'} s+v)\bullet(t_s + t_v)]$$

D5 $\varepsilon_t \oplus \varepsilon_v = \{\varepsilon_{t\oplus v},\ (t\oplus v \xrightarrow{\sigma} t),\ (t\oplus v \xrightarrow{\sigma} v)\},$

$$t_s \oplus \varepsilon_v = \{(s \oplus v \xrightarrow{\sigma} v\} \cup (s \oplus v \xrightarrow{\sigma} s)\bullet t_s$$
$$\cup[\upsilon = \sigma,\ (s \oplus v \xrightarrow{\sigma} t \oplus v)\bullet(t_t \oplus \varepsilon_v),\ \emptyset],$$

$$\varepsilon_t \oplus t_u = \{(t \oplus u \xrightarrow{\sigma} t)\} \cup (t \oplus u \xrightarrow{\sigma} u)\bullet t_u$$
$$\cup[\upsilon' = \sigma,\ (t \oplus u \xrightarrow{\sigma} t \oplus v)\bullet(\varepsilon_t \oplus t_v),\ \emptyset],$$

$$t_s \oplus t_u = (s \oplus u \xrightarrow{\sigma} s)\bullet t_s \cup (s \oplus u \xrightarrow{\sigma} u)\bullet t_u$$
$$\cup[\upsilon = \sigma,\ (s \oplus u \xrightarrow{\sigma} t \oplus u)\bullet(t_t \oplus t_u),\ \emptyset]$$
$$\cup[\upsilon' = \sigma,\ (s \oplus u \xrightarrow{\sigma} s \oplus v)\bullet(t_s \oplus t_v),\ \emptyset]$$

D6 $\varepsilon_t \,|\, \varepsilon_v = \{\varepsilon_{t|v}\},$

$$t_s \,|\, \varepsilon_v = ((s|v) \xrightarrow{\upsilon} (t|v))\bullet(t_t|\varepsilon_v),$$

$$\varepsilon_t \,|\, t_u = ((t|u) \xrightarrow{\upsilon'} (t|v))\bullet(\varepsilon_t \,|\, t_v),$$

$$t_s \,|\, t_u = ((s|u) \xrightarrow{\upsilon} (t|u))\bullet(t_t \,|\, t_u))$$
$$\cup\ ((s|u) \xrightarrow{\upsilon'} (s|v))\bullet(t_s \,|\, t_v))$$
$$\cup\ [\upsilon' = \overline{\upsilon} \in \Lambda,\ ((s|u) \xrightarrow{\tau} (t|v))\bullet(t_t \,|\, t_v), \emptyset]$$

The following propositions hold :

proposition $\quad \forall\ op_k \in \Sigma_k\ ,\ \forall\ t_i \in PROG\ ,$

$[\![\ op_k(t_1, \ldots\ t_k)\ ]\!]^*_{op} = \underline{op}_k([\![\ t_1\ ]\!]^*_{op}, \ldots\ [\![\ t_k\ ]\!]^*_{op})\ .$

proposition $\quad \forall\ op_k \in \Sigma_k,\ \forall t_i \in PROG\ ,$

$[\![\ op_k(t_1, \ldots\ t_k)\ ]\!]^\infty_{op} = \overline{op}_k([\![\ t_1\ ]\!]^\infty_{op}, \ldots\ [\![\ t_k\ ]\!]^\infty_{op})\ .$

3. AN OBSERVATIONAL PREORDER WITH HIGH POWER OF SEPARATION

Our foretold objective was to develop for CCS a model in which programs are identified if and only if no (CCS) program defined experiment makes the difference between them. In fact, we construct a model for an 'implementation' preorder between programs, where u is said to be an implementation of v if and only if no finite or infinite experiment on u tells that it is not v. We think of 'experimenting' on program u as running a maximal computation on the parallel system (t |C[u]), for some testing program t and context C[?]. The result of the experiment is the sequence of transitions of the testing program (along that experiment). The introduction of infinitary results, as opposed to the binary results of "classical" tests, is motivated by the fact that

infinitary results yield a strictly higher power of separation, and indeed the highest possible power of separation as regards CCS-programmed experiments. So, any observational model (based on CCS-experiments) is necessarily a quotient (by some equivalence) of the model which we are looking for. In the present section, we give a precise definition of the relation of implementation between programs, and then show how that relation may be transferred from programs to their associated sets of computations.

We introduce in this alinea precise terminology about computations. Let t denote some computation $(u_i \xrightarrow{v_i} u_{i+1})_{i<\gamma}$ in ${}^{\infty}\mathcal{T}$. The coarse trace of t (denoted $tr_{\mathcal{N}}(t)$ is the word $(v_i)_{i<\gamma}$ of $\mathcal{N}^{\infty}$, and the refined trace of t (denoted $tr_{\Lambda}(t)$) is the projection $\Pi_{\Lambda}(tr_{\mathcal{N}}(t))$, where Π_{Λ} is the usual morphism projecting $\mathcal{N}^{\infty}$ on Λ^{∞}. If $tr_{\Lambda}(t)$ is ε (the empty trace), then computation t is said to be silent. For t in PROG, we let MSC(t) be the family of Maximal Silent Computations from t, that is to say, the set of computations which are maximal w.r.t. $\leq$ (the order prefix) in $\{t \;/\; t \in [\![t]\!]^{\infty}_{op} \wedge tr_{\Lambda}(t) = \varepsilon\}$. For $t \in$ MSC(u|v), a pair of computations (t_u, t_v) is a parallel decomposition of t if it is minimal (w.r.t. $\leq^2$) in the set $\{(t_u, t_v) \;/\; t \in (t_u \mid t_v)\}$. (For the ease of notation, the operations $\overline{op_k} : ({}^{\infty}\mathcal{T})^k \to \mathcal{P}({}^{\infty}\mathcal{T})$, and their union additive extensions, are given the simpler notation op_k in the present section). For $t \in$ MSC (u|v), we let $\Pi_2(t)$ denote the set of computations t_v which are the second projection of some parallel decomposition (t_u, t_v) of t. Finally, we let Twincomp be the set of parallel decompositions of maximal silent computations.

definition Let u, v $\in$ PROG.
Program u is observationally less than program v ($u \lesssim v$) iff $\forall$ t $\in$ PROG, Π_2 (MSC(u|t)) $\subseteq \Pi_2$ (MSC(v|t)).

Program u is an implementation of program v ($u \sqsubseteq v$) iff C[u] $\lesssim$ C[v] for all program contexts C[?] , where a program context is a closed term with possible occurences of the dummy constant "?".

Examples The reader can convince himself that: $u \sqsubseteq u \oplus v$, $u \sqsubseteq \tau u$, $\tau(u+v) \sqsubseteq \tau u + \tau v$, $\tau v + u \sqsubseteq \tau(u+v) + \tau v$, (x wh x = τx) $\approx$ (x wh x = x).

A natural way to find out a fully abstract model for $\lesssim$ is to transfer that preorder from programs to sets of computations and then to analyse the resulting equivalence of sets of computations : the canonical morphism induced by the equivalence yields the fully abstract model. This program is undertaken in the next series of statements.

definition Let t, $t' \in {}^{\infty}\mathcal{T}$.
Computation t is observationally less than computation t' ($t \lesssim t'$) if and only if $\forall t'' \in {}^{\infty}\mathcal{T}$, $(t, t'') \in$ Twincomp $\Rightarrow$ $(t', t'') \in$ Twincomp.

We let $\lesssim$ denote also the Hoare like extension of $\lesssim$ to $\mathcal{P}(^{\infty}\mathcal{T}) \times \mathcal{P}(^{\infty}\mathcal{T})$, that is the binary relation $Q_1 \lesssim Q_2 \Leftrightarrow (\forall t_1 \in Q_1, \exists\, t_2 \in Q_2,\ t_1 \lesssim t_2)$.

proposition 1 $\forall\, u,v \in PROG : u \lesssim v \Leftrightarrow [\![u]\!]^{\infty}_{op} \lesssim [\![v]\!]^{\infty}_{op}$.

definition Let t be the computation $(u_i \xrightarrow{v_i} u_{i+1})_{i<\gamma}$. Then t has the property of stability (Stable(t)) if it is finite ($|t| = \gamma \neq \omega$) and cannot be extended by any silent transition (for all $(u \xrightarrow{v} v) \in \mathcal{T}$, $u = u_\gamma \Rightarrow v \in \Lambda$). If t is stable, then t has a residue Res(t), defined as the ready set of u_γ, i.e. as the set of labels $\{\lambda \in \Lambda \ / (\exists\, v,\ (u_\gamma \xrightarrow{\lambda} v) \in \mathcal{T})\}$.

proposition 2 $\forall\ t, t' \in {}^{\infty}\mathcal{T}$,
$t \lesssim t'$ iff the following relations hold :
- $tr_\Lambda(t) = tr_\Lambda(t')$,
- $|t| = \omega \Rightarrow |t'| = \omega$,
- (Stable(t) $\wedge$ $|t'| \neq \omega$) $\Rightarrow$ (Stable(t') $\wedge$ Res(t') $\subseteq$ Res(t)).

The interest of propositions 1 and 2 is to suggest an abstract model for CCS, and a way to derive the semantics in that model from the operational semantics studied in section 2. In fact, the observational preorder $\lesssim$ on programs is not preserved by all context operations and differs from the implementation preorder $\sqsubseteq$ (of course, the discrepancy between $\lesssim$ and $\sqsubseteq$ stems from the operation of sum (+)). In order to cope with the problem, we shall now try to guess the adequate transposition of the implementation preorder (from programs) to computations and sets of computations.

definition Let $t, t' \in {}^{\infty}\mathcal{T}$.
($t \sqsubseteq t'$) if and only if the following relations hold :
- $t \lesssim t'$,
- (Stable(t) $\wedge$ Stable(t') $\wedge$ $tr_\Lambda(t) = \varepsilon$) $\Rightarrow$ ($\Pi_\tau(tr_{\mathcal{N}}(t')) = \varepsilon \Rightarrow \Pi_\tau(tr_{\mathcal{N}}(t)) = \varepsilon$).

We let $\sqsubseteq$ denote also the Hoare like extension of $\sqsubseteq$ to $\mathcal{P}(^{\infty}\mathcal{T}) \times \mathcal{P}(^{\infty}\mathcal{T})$, that is the binary relation $Q_1 \sqsubseteq Q_2 \Leftrightarrow (\forall t_1 \in Q_1, \exists\, t_2 \in Q_2,\ t_1 \sqsubseteq t_2)$.
proposition 3 $\forall\ u, v \in PROG : u \sqsubseteq v \Rightarrow [\![u]\!]^{\infty}_{op} \sqsubseteq [\![v]\!]^{\infty}_{op}$.

One of the main goals of the next section is to establish the reverse proposition, i.e. the implication : $[\![u]\!]^{\infty}_{op} \sqsubseteq [\![v]\!]^{\infty}_{op} \Rightarrow u \sqsubseteq v$. Anticipating that result, we conclude the present section on a short remark: the implementation preorder studied here coincides with a variant of that of [17] (see also [15] for a stronger form), where the inclusion between finitary trace languages is replaced by an inclusion between infinitary trace languages. The variation concerns the phenomenon of divergence. More precisely, the congruence relation induced by the preorder of [17] separates unguardedness of recursive definitions from "pure" divergence, i.e. the presence of infinite sequences of internal actions. This separation corresponds to a further extended notion of testing where termination can be observed.

4. ABSTRACT COMPUTATIONS

The section prepares the construction of a model for the implementation preorder on programs. We define 'abstract computations', which represent actual computations by pairs encoding their properties w.r.t. preorder $\lesssim$. We then construct an ordered Σ-algebra of abstract computations and lift the operators to the power algebra. We show the monotony of the lifted operators, and study some properties of the resulting Σ-interpretation. We then turn ourselves to establish a connection between the operational interpretation $(\mathcal{P}(^{\infty}\mathcal{T}), \{\overline{op}_k\}, \sqsubseteq)$ and the abstract interpretation, and settle a morphic connection between the two ordered Σ-interpretations.

definition An 'abstract computation' is a pair $\langle w,R\rangle$ of one of the forms $\langle Z,\omega\rangle$ or $\langle z,S\rangle$ or $\langle z,\perp\rangle$ or $\langle \tau,S\rangle$ where $Z \in \Lambda^{\infty}$, $z \in \Lambda^{*}$ and S is a finite subset of Λ. The set $(\mathcal{A}, \sqsubseteq)$ of abstract computations is partially ordered by the least order relation satisfying the following axioms, for $z \in \Lambda^{*}$ and $S'' \subseteq S' \subseteq S \subseteq \Lambda$

. $\langle z,\perp\rangle \sqsubseteq \langle z,S\rangle \sqsubseteq \langle z,S'\rangle \sqsubseteq \langle z,\omega\rangle$,

. $\langle \varepsilon,S'\rangle \sqsubseteq \langle \tau,S'\rangle \sqsubseteq \langle \tau,S''\rangle \sqsubseteq \langle \varepsilon,\omega\rangle$.

The intentions behind that definition are made completely clear in the following statement.

proposition 1 $(\forall t,t' \in {}^{\infty}\mathcal{T})\ t \lesssim t' \Leftrightarrow \phi(t) \sqsubseteq \phi(t')$, for $\phi : {}^{\infty}\mathcal{T} \to \mathcal{A}$ defined as follows :

. $|t| = \omega \Rightarrow \phi(t) = \langle tr_{\Lambda}(t), \omega\rangle$

. $|t| \neq \omega \wedge \neg Stable(t) \Rightarrow \phi(t) = \langle tr_{\Lambda}(t), \perp\rangle$

. $(Stable(t) \wedge (tr_{\Lambda}(t) \neq \varepsilon \vee \Pi_{\tau}(tr_{\mathcal{N}}(t)) = \varepsilon)) \Rightarrow \phi(t) = \langle tr_{\Lambda}(t), Res(t)\rangle$

. $(Stable(t) \wedge tr_{\Lambda}(t) = \varepsilon \wedge \Pi_{\tau}(tr_{\mathcal{N}}(t)) \neq \varepsilon)) \Rightarrow \phi(t) = \langle \tau, Res(t)\rangle$

proof. The verification is straightforward, and we leave it to the reader as a useful exercise for a full understanding of the above definition.

In all the sequel, we let a (resp. A) denote abstract computations (resp. sets of abstract computations). We also use $\sqsubseteq$ to denote the Hoare extension of $\sqsubseteq$ on $\mathcal{P}(\mathcal{A})$, i.e. the relation $A_1 \sqsubseteq A_2 \Leftrightarrow (\forall a_1 \in A_1,\ \exists a_2 \in A_2,\ a_1 \sqsubseteq a_2)$. Our next goal is to introduce a series of elementary operations $op_k : \mathcal{A}^k \to \mathcal{P}(\mathcal{A})$, for $op_k \in \Sigma$. Before the corresponding definitions are given, we need still introducing an auxiliary operator on words, namely the 'parallel composition' $(\odot)$ which we define as follows.

Let $N = (\mathcal{N} \times \{\varepsilon\}) \cup (\{\varepsilon\} \times \mathcal{N}) \cup \{(\lambda,\overline{\lambda}) \ / \ \lambda \in \Lambda\}$. The parallel composition $\odot : \mathcal{N}^{\infty} \times \mathcal{N}^{\infty} \to \mathcal{P}(\mathcal{N}^{\infty})$ is the function $w' \odot w'' = \{w \ / \ (w',w'') \in \psi_2(\psi_1^{-1}(w))\}$ for $\psi_1 : N^{\infty} \to \mathcal{N}^{\infty}$ and $\psi_2 : N^{\infty} \to \mathcal{N}^{\infty} \times \mathcal{N}^{\infty}$ defined as the continuous extensions of morphisms $\psi_1 : N^{*} \to \mathcal{N}^{*}$ and $\psi_2 : N^{*} \to \mathcal{N}^{*} \times \mathcal{N}^{*}$ as follows :

. $\psi_1(\upsilon,\varepsilon) = \upsilon, \quad \psi_1(\varepsilon,\upsilon) = \upsilon, \quad \psi_1(\lambda,\overline{\lambda}) = \tau$.

. $\psi_2(n) = n \qquad$ for $n \in N$,

. $\psi_2(nn') = (n_1n'_1, n_2n'_2)$ for $\psi_2(n)=(n_1,n_2)$ and $\psi_2(n')=(n'_1,n'_2)$

The elementary operations $op_k : \mathcal{A}^k \to \mathcal{P}(\mathcal{A})$ are given by the following table of statements, where set brackets are omitted from singleton sets.

THE INTERPRETATION OF Σ IN $\mathcal{A}$

D'1 $NIL = \langle\varepsilon,\emptyset\rangle$,

D'2 $\mu\langle w,R\rangle = [\mu \in \Lambda, \langle\varepsilon, \{\mu\}\rangle, \langle\varepsilon,\bot\rangle] \cup$ if $(R \in \{\bot,\omega\} \quad \vee w \notin \{\varepsilon,\tau\})$
then $[\mu = \tau, \langle w,R\rangle, \langle\mu w,R\rangle]$
else $[\mu =\tau, \langle\tau,R\rangle, \langle\mu,R\rangle]$,

D'3 $\langle w,R\rangle\rho =$ if $w \notin (dom(\rho))^\infty$ then $\emptyset$ else
$[R \in \{\bot,\omega\}, \langle\rho(w),R\rangle, \langle\rho(w), \rho(R \cap dom(\rho))\rangle]$,

D'4 $\langle w,R\rangle + \langle w',R'\rangle = [(w \neq\varepsilon \vee R = \omega), \langle w,R\rangle , [R = \bot, \langle\varepsilon,\bot\rangle , \emptyset]] \cup$
$[(w' \neq\varepsilon \vee R' = \omega), \langle w',R'\rangle, [R' = \bot, \langle\varepsilon,\bot\rangle, \emptyset]] \cup$
$[(w = \varepsilon \wedge w' = \varepsilon \wedge \{R,R'\}\cap \{\bot,\omega\} = \emptyset), \langle\varepsilon, R \cup R'\rangle, \emptyset]$,

D'5 $\langle w,R\rangle \oplus \langle w',R'\rangle = \{\langle\varepsilon,\bot\rangle, \langle w,R\rangle, \langle w',R'\rangle\}$,

D'6 $\langle w,R\rangle \mid \langle w',R'\rangle =$ if $(R = \omega \vee R' = \omega)$ then $\{\langle w'', \omega\rangle / w'' \in \Pi_\Lambda(w \Theta w')\}$
else if $(R = \bot \vee R' = \bot \vee \exists\lambda, \lambda\in R \wedge \overline{\lambda}\in R')$
then $\{\langle w'',\bot\rangle / w'' \in \Pi_\Lambda (w \Theta w')\}$
else $\{\langle h(w''), R \cup R'\rangle / w'' \in \Pi_\Lambda(w \Theta w')\}$,
letting $h(w'') = [\Pi_\Lambda(w'')\neq\varepsilon, \Pi_\Lambda (w''), [\Pi_\tau(w'')\neq\varepsilon, \tau, \varepsilon]]$.

In order to construct the interpretation of the fully abstract model in prospect, we lift the above definitions to sets. However, we put special restrictions on the lifting process, for we care to obtain a complete partial order with monotone and continuous operations. Definitions follow.

Our domain of interpretation $\mathcal{D}$ is the set of subsets of $\mathcal{A}$ with pairwise incomparable elements, i.e.
$\mathcal{D} = \{A \in \mathcal{P}(\mathcal{A}) / (\forall a_1,a_2 \in A) (a_1 \sqsubseteq a_2 \Rightarrow a_1 = a_2)\}$.
Thus, $\mathcal{D}$ is the image of $\mathcal{P}(\mathcal{A})$ through function 'roof': $\mathcal{P}(\mathcal{A}) \twoheadrightarrow \mathcal{D}$:
$\hat{A} = \{a \in A / (\forall a' \in A)\{a \sqsubseteq a' \Rightarrow a'=a)\}$.
The interpretations $op_k : \mathcal{D}^k \to \mathcal{D}$ of operators in Σ are given by the generic formula : $op_k(A_1,\ldots A_k) =$ if $(\exists i, A_i = \emptyset)$ then $\emptyset$
else roof $((\langle\varepsilon,\bot\rangle) \cup (\cup op_k (a_1,\ldots a_k) / a_i \in A_i))$.
Some properties of the resulting Σ-interpretation are stated below.

proposition 2 $(\mathcal{D}, \sqsubseteq, \emptyset)$ is a complete upper semi-lattice, with the least upper bound of subsets given by $\bigsqcup D = roof(\cup A / A \in D)$.
proof It is enough to establish the simpler proposition :
$(\forall A' \in D) (A' \sqsubseteq roof(\cup A / A \in D))$. Let's suppose it is false; then the only possibility is that, for some $a \in A'$, there is in $(\cup A/A \in D)$ a strictly increasing ω-chain of abstract computations (a_i) with $a_o = a$. But no such chain can exist by the definition of $\sqsubseteq$ on $\mathcal{A}$.

proposition 3 $(\mathcal{D}, \sqsubseteq, \{op_k\})$ is a continuous Σ-algebra.
proof The lemma given afterwards shows that operations op_k are monotone. Continuity follows by the implication : $a \in op_2(\bigsqcup D, A') \Rightarrow \exists\, A \in D,\ a \in op_2(A, A')$ and variants of it.

lemma For any op_k in Σ, for any a_i, a'_i in $\mathcal{A}$, and for any a in op_k $(a_1, .. a_k)$: $(a \neq \langle \varepsilon, \perp \rangle) \wedge (\forall i,\ a_i \sqsubseteq a'_i) \Rightarrow (\exists a' \in op_k(a'_1, \ldots a'_k),\ a \sqsubseteq a')$.
proof by systematic examination of all the possible cases (left to the reader).

We finally establish the morphic connection between the operational Σ-interpretation of section 2 and the abstract Σ-interpretation of the present section.

proposition 4 Let $\mathcal{P}_{\leq}(^{\infty}\mathcal{T})$ denote the set of prefix closed subsets of computations, and let $\hat{\phi} : \mathcal{P}_{\leq}(^{\infty}\mathcal{T}) \to \mathcal{D}$ be the function:
$\hat{\phi}(Q) = \mathrm{roof}(\cup\{\phi(t)/t \in Q\})$. Then $\hat{\phi}$ acts like a morphism between structures $(\mathcal{P}_{\leq}(^{\infty}\mathcal{T}),\ \{\overline{op}_k / op_k \in \Sigma\}, \lesssim)$ and $(\mathcal{D},\ \{op_k\},\ \sqsubseteq)$.

5. FULL ABSTRACTION AND TRACES

Our present goal is to establish full abstractness (w.r.t. the observational preorder $\lesssim$) of the function $[|.|]_{obs} : PROG \to \mathcal{D} : [|t|]_{obs} = \hat{\phi}\ [|t|]^{\infty}_{op}$.
We recall from [22] that full abstractness is expressed by the logical equivalence

(FA) $\quad [|u|]_{obs} \lesssim [|v|]_{obs} \quad \Leftrightarrow \quad \forall\ C[?],\ C[u] \lesssim C[v]$.

Due to propositions 3.1, 3.3 and 4.1, FA follows easily from the following property of monotony :

(MON) $\quad \forall\ C[?], \quad [|u|]_{obs} \sqsubseteq [|v|]_{obs} \Rightarrow [|C[u]|]_{obs} \sqsubseteq [|C[v]|]_{obs}$.

Indeed, the general version of MON follows from the specialized version given below for declaration contexts D[?] :

(MON-D) $\quad \forall\ D[?],\ \hat{\phi}[|u|]^{\infty}_{op} \sqsubseteq \hat{\phi}[|v|]^{\infty}_{op} \Rightarrow \hat{\phi}\,[|\, x \text{ wh } D[u]\ |]^{\infty}_{op} \sqsubseteq \hat{\phi}[|x \text{ wh } D[v]\ |]^{\infty}_{op}$.

In the operational model sketched in (2.3), $[|\, x \text{ wh } D[u]\ |]^{\infty}_{op}$ turns out to be the projection along the x-component of the greatest solution of a parameterized system $\mathbb{G}_u(\mathbb{X})$ of Σ-equations in $\mathcal{P}(^{\infty}\mathcal{T})^n$. The parameter of that system is $[|\, u\ |]^{\infty}_{op}$ and the interpretation of Σ is according to the definition given in (2.3). Now, MON-D is a direct consequence of the following two implications, where l.f.p. (g.f.p.) is the combinator of least (greatest) fixed point in $(\mathcal{P}(^{\infty}\mathcal{T}), \subseteq)^n$, Π_x is the projection along x, and $\setminus$ is the componentwise difference between vectors of sets :

(MON-LFP) $\quad \hat{\phi}[|u|]^{\infty}_{op} \sqsubseteq \hat{\phi}[|v|]^{\infty}_{op} \Rightarrow \hat{\phi}(\Pi_x(\text{l.f.p. } \mathbb{G}_u(\mathbb{X}))) \sqsubseteq \hat{\phi}(\Pi_x(\text{l.f.p. } \mathbb{G}_v(\mathbb{X})))$,

(MON-GFP) $\quad \hat{\phi}[|u|]^{\infty}_{op} \sqsubseteq \hat{\phi}[|v|]^{\infty}_{op} \Rightarrow$
$\hat{\phi}(\Pi_x(\text{g.f.p.}\mathbb{G}_u(\mathbb{X}) \setminus \text{l.f.p. } \mathbb{G}_u(\mathbb{X}))) \sqsubseteq \hat{\phi}(\Pi_x(\text{g.f.p. } \mathbb{G}_v(\mathbb{X})))$.

Relation MON-LFP follows easily from the usual properties of monotony of least fixed points (of continuous operators) : the trick is to shift $\hat{\phi}$ on the right of the l.f.p. combinator (using the morphic properties of $\hat{\phi}$ w.r.t. Σ). Relation MON-GFP is not so easily proved. The remaining of the section is devoted to that task.

Traces

We introduce a Σ power algebra of traces $(\mathcal{P}(\mathcal{N}^\infty), \{\overline{op}_k\}, \subseteq)$. The construction is the same as for the Σ power algebra of computations, but with program states omitted. As regards continuity and fixed points, all the properties which are valid for computations remain valid in the new framework. The operations $\overline{op}_k : (\mathcal{N}^\infty)^k \to \mathcal{P}(\mathcal{N}^\infty)$ are the extensions of finitary operations $\underline{op}_k : (\mathcal{N}^*)^k \to \mathcal{P}(\mathcal{N}^*)$, according to the formula $\overline{op}_k(w_1, \ldots w_k) = CL(\underline{op}_k(Pref(w_1), \ldots Pref(w_k)))$.
The finitary operations $\underline{op}_k$ are defined by the following inductive relations, also valid for the infinitary operations $\overline{op}_k$, where $\bullet : \mathcal{N} \times \mathcal{P}(\mathcal{N}^\infty) \to \mathcal{P}(\mathcal{N}^\infty)$ is the function: $\upsilon \bullet B = \{\varepsilon, \upsilon\} \cup (\upsilon B) \cup (\upsilon\, Pref(B))$.

THE INTERPRETATION OF Σ IN $\mathcal{N}^*$ AND $\mathcal{N}^\infty$

D"1 $NIL = \{\varepsilon\}$

D"2 $\mu(w) = \mu \bullet \{w\}$

D"3 $(\varepsilon)\rho = \{\varepsilon\}$,
$(\upsilon w)\rho = [\upsilon \notin dom(\rho),\ \{\varepsilon\},\ \rho(\upsilon) \bullet (w)\rho\ ]$

D"4 $\varepsilon + \varepsilon = \{\varepsilon\}$,
$\upsilon w + \varepsilon = \upsilon \bullet \{w\} = \varepsilon + \upsilon w$,
$\upsilon w + \upsilon' w' = [\upsilon = \sigma, \sigma \bullet (w + \upsilon' w'),\ \upsilon \bullet w] \cup [\upsilon' = \sigma,\ \sigma \bullet (\upsilon w + w'),\ \upsilon' \bullet w']$

D"5 $\varepsilon \oplus \varepsilon = \{\varepsilon, \sigma\}$,
$\upsilon w \oplus \varepsilon = \sigma \bullet \{\upsilon w\} = \varepsilon \oplus \upsilon w$,
$\upsilon w \oplus \upsilon' w' = \sigma \bullet \{\upsilon w\} \cup \sigma \bullet \{\upsilon' w'\}$
$\cup\ [\upsilon = \sigma,\ \sigma \bullet (w \oplus \upsilon' w'),\ \emptyset]$
$\cup\ [\upsilon' = \sigma,\ \sigma \bullet (\upsilon\ w \oplus w'),\ \emptyset]$

D"6 $\varepsilon \mid \varepsilon = \{\varepsilon\}$,
$\upsilon w \mid \varepsilon = \upsilon \bullet w = \varepsilon \mid \upsilon w$,
$\upsilon w \mid \upsilon' w' = \upsilon \bullet (w \mid \upsilon' w') \cup \upsilon' \bullet (\upsilon w \mid w')$
$\cup\ [\overline{\upsilon} = \upsilon' \in \Lambda,\ \tau \bullet (w \mid w'),\ \emptyset\ ]$

Σ-trees and substitutions

We introduce a countable set of variables I, with typical element i. We denote respectively by Σ^∞ the set of finite and infinite Σ-trees, and by $\Sigma^\infty(I)$ the set of finite and infinite trees on $\Sigma \cup I$. A tree is called a good tree if it has an infinite number of operators σ on each infinite branch. We use $T <_\Omega T'$ to indicate that T is a finite approximation of T', which means that T is finite and is less than T' in the usual syntactic order, where Ω is identified with NIL. A tree T in Σ^∞ may be interpreted by a corresponding set of words $[|T|]$ in $\mathcal{N}^\infty$. The interpretation of finite trees is defined as follows :

$[\![\text{NIL}]\!] = \{\varepsilon\}$,

$[\![\upsilon T]\!] = \upsilon \bullet [\![T]\!]$,

$[\![op_k(T_1, \ldots T_k)]\!] = \underline{op}_k([\![T_1]\!], \ldots [\![T_k]\!])$.

The interpretation of infinite trees is defined by continuous extension of the finitary interpretation : $[\![T]\!] = CL(\cup [\![T']\!] \ / \ T' <_\Omega T)$.

We call substitution a mapping S from I to $\mathcal{N}^\infty$. For T in $\Sigma^\infty(I)$, Sub(T,S) denotes the result of applying substitution S to tree T.

Towards a proof of MON-GFP

The following property of the projection operator $\Pi_\Lambda : \mathcal{P}(\mathcal{N}^\infty) \to \mathcal{P}(\Lambda^\infty)$ is useful:

proposition 1 For any finite tree T and for any substitution S,
$\Pi_\Lambda[\![Sub(T,S)]\!] = \Pi_\Lambda[\![Sub(T, \Pi_\Lambda \circ S)]\!]$.

We state afterwards two other propositions which entail MON-GFP.

proposition 2 Let T be a good tree, and let S and S' be substitutions satisfying $\Pi_\Lambda \circ S(i) = S'(i)$ for all $i \in I$. Then for any word $w \in [\![Sub(T,S)]\!]$, either $w \in [\![Sub(T',S)]\!]$ for some finite approximation T' of T, or $\Pi_\Lambda(w) = \Pi_\Lambda(w')$ for some infinite word w' in $[\![Sub(T,S')]\!]$.

proposition 3 Let T be a good tree, and let S and S' be substitutions satisfying $S(i) = \Pi_\Lambda \circ S'(i)$ for all $i \in I$. Then for any infinite word w in $[\![Sub(T,S)]\!]$, $\Pi_\Lambda(w) = \Pi_\Lambda(w')$ for some infinite word w' in $[\![Sub(T,S')]\!]$.

6. THE FULLY OBSERVATIONAL MODEL

The declared objective of the paper was to supply CCS with a fully abstract model, where the gauge for the comparison of programs is the largest possible family of CCS experiments with infinitary results. The full abstractness of function $[\![.]\!]_{obs}$: PROG $\to \mathcal{D}$, stated in section 5, allows us to assert that the continuous algebra $(\mathcal{D}, \sqsubseteq, \{op_k\})$ is the interpretation of such a model. Further, the relation $[\![t]\!]_{obs} = \hat{\phi}[\![t]\!]^\infty_{op}$ suggests us a method for the explicit construction of the abstract model. The method amounts to a transposition of the operational model through the abstraction morphism $\hat{\phi}$. The result of the transposition is a meaning function ν_{obs} : TERM $\to$ (ENV $\to \mathcal{D}$), where ENV = $(\mathcal{X} \to \mathcal{D})$. A simple optimization of the transposition, namely the uniform replacement of $\nu_{obs}[\![t \ \underline{wh} \ \emptyset]\!](e)$ by $\nu_{obs}[\![t]\!](e)$ yields the following definition.

THE SPECIFICATION OF ν_{obs}

1. $\nu_{obs} [\![x]\!](e) = e(x)$,
2. $\nu_{obs} [\![op_k(t_1, \ldots t_k)]\!](e) = op_k(\nu_{obs}[\![t_1]\!](e), \ldots \nu_{obs}[\![t_k]\!](e))$,

3. If D = $(x_1 = v_1, \ldots x_n = v_n)$,
Let $\mathbb{x} = (x_1, \ldots x_n)$, $\mathbb{X} = (X_1, \ldots X_n)$, $\mathbb{F} = (F_1, \ldots F_n)$
For $X_i \in \mathcal{D}$ and F_i : ENV $\to \mathcal{D}$ given by
$F_i(e) = \mathcal{V}_{obs}[\![v_i]\!](e)$, then $\mathcal{V}_{obs}[\![t \text{ wh } D]\!](e) =$
$\mathcal{V}_{obs}[\![t]\!]\ (e[fix_D.\mathbb{F}(e[\mathbb{X}/\mathbb{x}])/\mathbb{x}\ ])$.

Remark: The specification remains correct if $\langle w, \perp \rangle$ is converted into $\emptyset$ everywhere in the definition of $\mathcal{D}$ and the related operations.

There remains to give the right definition for the fixed point combinator fix_D , where the notation indicates a possible dependence on declaration D. The results presented in section 5 lead us to proceed as follows.

Let roof : $\mathcal{P}(\mathcal{A}) \to \mathcal{D}$ and omega : $\mathcal{P}(\mathcal{N}^\infty) \to \mathcal{D}$ be the functions s.t.:
roof(A) = {a ∈ A / ∀a'∈ A, a ⊑a' ⇒ a' = a} and
omega(B) = {<w,ω> / w= Π_Λ(w') for some w' ∈ B∩ $\mathcal{N}^\omega$}.

- For open declarations D, we state :
$fix_D.\mathbb{F}(\mathbb{X}) = l.f.p.\mathbb{F}(\mathbb{X})$.
- For closed declarations D, we state :
$fix_D.\mathbb{F}(\mathbb{X}) = \mathbb{ROOF}(l.f.p.\ \mathbb{F}(\mathbb{X}) \cup \mathbb{OMEGA}\ (g.f.p.\ \widetilde{\mathbb{F}}(\mathbb{X})))$,

letting $\widetilde{\mathbb{F}} = (\widetilde{F_1}, \ldots \widetilde{F_n})$ and $\widetilde{F_i}$: $(\mathcal{X} \to \mathcal{P}(\mathcal{N}^\infty)) \to \mathcal{P}(\mathcal{N}^\infty)$: $\widetilde{F_i}(e) = \mathcal{V}_{tr}[\![v_i]\!](e)$
for $\mathcal{V}_{tr}$: TERM $\to$ $((\mathcal{X} \to \mathcal{P}(\mathcal{N}^\infty)) \to \mathcal{P}(\mathcal{N}^\infty))$ the auxiliary meaning function defined afterwards.
Let Π_Λ: $\mathcal{D} \to \mathcal{P}(\mathcal{N}^\infty)$ be the 'projection' $\Pi_\Lambda(A) = \{\Pi_\Lambda(w) / w \leq w'$ for some $\langle w', R' \rangle \in A\}$.
The function $\mathcal{V}_{tr}$ is obtained by a transposition of the operational meaning function through the abstraction morphism $\psi: ({}^\infty\mathcal{T}) \to \mathcal{N}^\infty$: $\psi(t_i - \upsilon_i \to t_{i+1})_{i<\gamma} = (\upsilon_i)_{i<\gamma}$.
However, in the specification of $\mathcal{V}_{tr}(t)$, we let $\Pi_\Lambda(\mathcal{V}_{obs}(u))$ play the role of $\psi([\![u]\!]^\infty_{op})$ for all proper subprograms u of t. That replacement agrees with propositions 5.2 and 5.3. Furthermore, it affords a compositional expression of $\mathcal{V}_{obs}$ at the level of programs and closed declarations, even though an auxiliary meaning function is called for. We also proceed to the optimization of the transposition by reducing uniformly $\mathcal{V}_{tr}[\![t \text{ wh } \emptyset]\!](e)$ to $\mathcal{V}_{tr}[\![t]\!](e)$.

THE SPECIFICATION OF $\mathcal{V}_{tr}$

0. If t∈ PROG then $\mathcal{V}_{tr}[\![t]\!](e) = \Pi_\Lambda(\mathcal{V}_{obs}[\![t]\!](e'))$ for arbitrary e' ∈ $(\mathcal{X} \to \mathcal{D})$,
otherwise $\mathcal{V}_{tr}[\![t]\!](e)$ is defined by relations 1 to 3.

1. $\mathcal{V}_{tr}[\![x]\!](e) = e(x)$,

2. $\mathcal{V}_{tr}[\![op_k(t_1, \ldots t_k)]\!](e) = op_k(\mathcal{V}_{tr}[\![t_1]\!](e), \ldots \mathcal{V}_{tr}[\![t_k]\!](e))$,

3'. If D is a closed declaration $(x_1 = v_1, \ldots x_n = v_n)$,
let $[\![D]\!]_{obs} = (\mathcal{V}_{obs}[\![x_1 \text{ wh } D]\!](e'), \ldots, \mathcal{V}_{obs}[\![x_n \text{ wh } D]\!](e'))$,
for arbitrary e'∈ $(\mathcal{X} \to \mathcal{D})$, and let $\mathbb{x} = (x_1 \ldots x_n)$, then
$\mathcal{V}_{tr}[\![t \text{ wh } D]\!](e) = \mathcal{V}_{tr}[\![t]\!](e[\Pi_\Lambda([\![D]\!]_{obs})/\mathbb{x}])$,

3". If D is an open declaration $(x_1 = v_1, \ldots x_n = v_n)$,
let $\mathbb{x} = (x_1, \ldots x_n)$, $\mathbb{X} = (X_1, \ldots X_n)$, $\mathbb{F} = (F_1, \ldots F_n)$

for $X_i \in \mathcal{P}(\mathbb{N}^\infty)$ and $F_i : (\mathcal{X} \to \mathcal{P}(\mathbb{N}^\infty)) \to \mathcal{P}(\mathbb{N}^\infty)$
given by $F_i(e) = \sigma \bullet \mathcal{V}_{tr}[\![v_i]\!](e)$, then

$$\mathcal{V}_{tr}[\![t \ \underline{wh}\ D\ ]\!](e) = \mathcal{V}_{tr}[\![t]\!](e[g.f.p.\ \mathbb{F}\ (e[\mathbb{X} / \mathbb{X}]) / \mathbb{X}])$$

The outcome of the definitions is the following result.

<u>theorem</u>. $\forall t \in$ PROG, $\mathcal{V}_{obs}[\![t]\!](e) = \hat{\phi}[\![t]\!]^\infty{}_{op}$ for arbitrary environments $e : \mathcal{X} \to \mathcal{D}$, and thus $\mathcal{V}_{obs}$ is a fully abstract model for CCS.

BIBLIOGRAPHY

[1] Abramsky S. Experiments, Power domains, and Fully Abstract Models for Applicative Multiprogramming FCT 83, Borgholm Springer-Verlag, LNCS 158 (83) pp. 1-13

[2] Arnold A., Dicky A. An Algebraic Characterization of Transition System Equivalence, Report I-8603, Université de Bordeaux (86)

[3] Bekić H. Definable Operations in General Algebras, and the Theory of Automata and Flowcharts, IBM Laboratory Vienna (69) also in: Programming Languages and their Definition, selected papers of H. Bekić, Springer-Verlag LNCS 177(84)

[4] Bergstra J.A., Klop J.W., Olderog E.R. Readies and Failures in the Algebra of Communicating Processes, Report CS-R8523, CWI, Amsterdam (85)

[5] Boasson L., Nivat M. Adherences of Languages, JCSS 20 (80) pp. 285 -309

[6] Boudol G. Notes on Algebraic Calculi of Processes, Logics and Models of Concurrent Systems, K. Apt. ed, NATO - ASI -Series, Springer-Verlag (85) pp. 261 - 304

[7] Brookes S, Roscoe A. W. An Improved Failures Model for Communicating Processes Seminar on Concurrency, Brookes Roscoe Winskel eds. Springer-Verlag, LNCS 197(85) pp. 281-305

[8] Darondeau Ph. Kott L. On the Observational Semantics of Fair Parallelism, ICALP 83, Barcelona, Springer-Verlag LNCS 154 (83) pp. 147 - 159

[9] Darondeau Ph. About Fair Asynchrony, TCS 37 (85) pp. 305 - 336

[10] " Separating and Testing , STACS 86, Orsay, Springer-Verlag LNCS 210 (86) pp. 203 - 212

[11] de Bakker J.W., Meyer J.J.Ch., Olderog E.R. Infinite Streams and Finite Observations in the Semantics of Uniform Concurrency, ICALP 85, Naflion, Springer-Verlag LNCS 194 (85) pp. 149 - 157

[12] de Bakker J.W., Meyer J.J.Ch., Olderog E.R., Zucker J.I. Transition Systems, Infinitary Languages and the Semantics of Uniform Concurrency 17^{th} ACM STOC, Providence (85) pp. 252,262

[13] de Nicola R., Hennessy M. Testing Equivalences for Processes TCS 34 (84) pp. 83-133

[14] de Nicola R. Testing Equivalences and Fully Abstract Models for Communicating Processes. Ph. D Thesis, University of Edinburgh (85)

[15] Gamatie B.
Observational Congruences of Non Deterministic and Communicating Finite Processes RR - 254 - IRISA Rennes (85)

[16] "
Safe Implementation Equivalence for Asynchronous Non Deterministic Processes, MFCS 86, Bratislava, Springer-Verlag LNCS 233 (86)

[17] "
Towards Specification and Proof of Asynchronous Systems, STACS 86, Orsay, Springer-Verlag LNCS 210 (86) pp. 203 - 212

[18] Golson W., Rounds W. C. Connections Between two Theories of Concurrency : Metric Spaces and Synchronization Trees
Information and Control 57 (83) pp.102-124

[19] Hennessy M., Plotkin G. A Term Model for CCS
Springer-Verlag LNCS 88 (80) pp.261-274

[20] Hennessy M. Acceptance Trees, JACM 32 (85) pp. 896 - 928

[21] Jorrand Ph Specification of Communicating Processes and Process Implementation Correctness, 5th Int. Symposium on Programming, Turin
Springer Verlag LNCS 137(82) pp.242-256

[22] Milner R. Fully Abstract Models of Typed lambda-calculi , TCS 4 (77) pp. 1 - 23

[23] " A Calculus of Communicating Systems , Springer - Verlag LNCS 92 (80)

[24] Olderog E. R., Hoare C.A.R. Specification Oriented Semantics for Communicating Processes , Acta Informatica 23,1 (86) pp. 9-86

[25] Plotkin G. A Structural Approach to Operational Semantics,
Rept. DAIMI - FN - 19, Univ. of Aarhus, Computer Science Department (81)

[26] Pnueli A. Linear and Branching Structures in the Semantics and Logics of Reactive Systems, ICALP 85, Nafplion, Springer - Verlag LNCS 194 (85) pp. 15 - 32

[27] Rounds W. C. On the Relationships Between Scott Domains, Synchronization Trees and Metric Spaces, Information and Control 66 (85) pp. 6 - 28

[28] Tarski A. A Lattice Theoretic Fixpoint Theorem and its Applications,
Pacific Journal of Mathematics 5 (55) pp. 285 - 309.