数理解析研究所講究録
902巻 1995年 26-48

A Fully Abstract Denotational Model for Communicating Processes with Label-Passing

Eiichi Horita (堀田 英一), Fer-Jan de Vries (フェル-ヤン ドゥ フリース)
NTT Communication Science Laboratories
2-2, Hikaridai, Seika-Cho, Soraku-Gun, Kyoto 619-02, Japan
E-mail: horita@progn.kecl.ntt.jp, ferjan@progn.kecl.ntt.jp

Abstract: This paper investigates the full abstractness problem for a CCS-like language, CCS^{++}, with a *label-passing* mechanism. (Here *labels* are used as the names of communication channels through which interactions between processes take place.) This language is essentially the same as the one proposed by Astesiano et al.; it is an extension of full CCS of Milner and contains *label-passing*— an important feature of π-calculus of Milner et al.— in addition to the traditional CCS constructs: *inputs*, *outputs*, *parallel composition*, *nondeterministic choice*, *action restriction*, and *recursion*. First an operational model $\mathcal{O}$, called the weak maximal linear semantics, is defined in terms of a Plotkin-style transition system; $\mathcal{O}$ serves as a behavioral criterion for assessing semantic models for CCS^{++}. Then a denotational model $\mathcal{D}$ is constructed on the basis of a complete partial order (cpo) and it is shown that $\mathcal{D}$ is fully abstract with respect to $\mathcal{O}$ in the sense that $\mathcal{D}$ is the most abstract of those models which are less abstract than $\mathcal{O}$ and compositional. The full abstractness result means that $\mathcal{D}$ is the *most desirable* (or *optimal*) semantic model from the practical viewpoint embodied by $\mathcal{O}$.

1 Introduction

Full abstractness concerns the relationship between two semantic models, say $\mathcal{D}$ and $\mathcal{O}$, for a given language $\mathcal{L}$. We say that the model $\mathcal{D}$ is *fully abstract* with respect to the model $\mathcal{O}$ if the following holds for any two programs s_0, s_1 in $\mathcal{L}$:

$$\mathcal{D}[\![s_0]\!] = \mathcal{D}[\![s_1]\!] \text{ iff } \mathcal{O}[\![C[s_0]]\!] = \mathcal{O}[\![C[s_1]]\!] \text{ for all contexts } C[\cdot] \text{ of } \mathcal{L}. \quad (1)$$

Typically $\mathcal{O}$ is an operational semantics describing the behavior of processes in terms a Plotkin-style transition system.

The *full abstractness problem* for programming languages was first raised by Milner [26]. In general, a fully abstract model for a given language with respect to a given operational semantics $\mathcal{O}$ is the most abstract, and hence, the *most desirable* (or *optimal*) of those models which are less abstract than $\mathcal{O}$ and compositional. (Here we say a model $\mathcal{D}_1$ is *less abstract* than another model $\mathcal{D}_2$ iff every two programs having the same meaning under $\mathcal{D}_1$ have the same meaning under $\mathcal{D}_2$.) The model $\mathcal{D}$ can be considered the optimal model in the following sense: The operational semantics $\mathcal{O}$ may be considered to describe the most interesting characteristic of each process; such a model, however, need not be compositional (see Sect. 1.2 of [27]). (This fact is also exhibited in the setting of this paper; see Example 1.) Compositionality, in turn, is an essential property to fit the semantic model into an axiomatic framework based on equational logic; it is also necessary for the *modular* definition of program meanings (i.e., in order to define the meaning of a composite statement in terms of the meanings of its components). Thus some *extra* information needs to be involved to construct a compositional model; it is desirable, however, for the extra information to be *minimum* so as not to bring about inessential details. The fully abstract model $\mathcal{D}$ meets these requirements (see Fig. 1 for an illustration of the optimality of $\mathcal{D}$).

In this paper, we investigate the full abstractness problem for a CCS-like language, CCS^{++}, with a *label-passing* mechanism. (Here *labels* are used as the names of communication channels through which interactions between processes take place.) This language is essentially the same as the one proposed in [1]; it is an extension of full CCS [27] and contains *label-passing*—an important feature of π-calculus [29]— in addition to the traditional CCS constructs: *inputs*, *outputs*, *parallel composition*, *nondeterministic choice*, *action restriction*, and *recursion*.

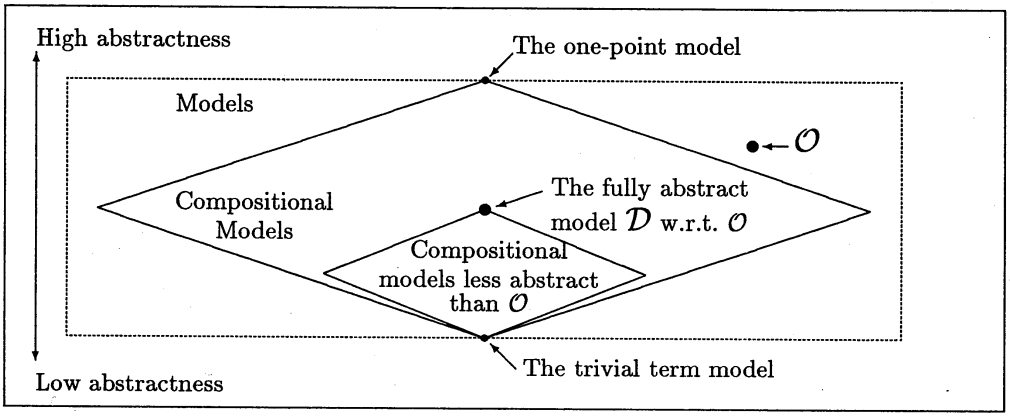


Figure 1: The hierarchy of semantic models

First an operational model $\mathcal{O}$, called the weak maximal linear semantics, is defined in terms of a Plotkin-style transition system. The model $\mathcal{O}$ is *linear* and *maximal* in that the meaning $\mathcal{O}[\![s]\!]$ of each program s is the set of *maximal sequences* of actions the program may perform, and it is *weak* in that the action sequences are obtained by abstracting from (finite sequences of) *internal moves*.[1] From a certain practical viewpoint, $\mathcal{O}$ can be considered to represent the most interesting characteristic of each (software or hardware) system; $\mathcal{O}$ serves as a behavioral criterion for assessing semantic models for CCS^{++}. Then a denotational model $\mathcal{D}$ is constructed on the basis of a complete partial order (cpo) and it is shown that $\mathcal{D}$ is fully abstract with respect to $\mathcal{O}$. The full abstractness result means that $\mathcal{D}$ is the *most desirable* (or *optimal*) semantic model from the practical viewpoint embodied by $\mathcal{O}$.

The cpo $\mathbf{P}$ underlying $\mathcal{D}$ is called the *Smyth powerdomain of failures*. This domain consists of *failure-divergence sets* first introduced in [7]; $\mathbf{P}$ is not only a cpo but also a *compete metric space* (*cms*) with an appropriate metric defined as in [3], and this twofold structure of $\mathbf{P}$ is conveniently used for establishing the full abstractness of $\mathcal{D}$ with respect to $\mathcal{O}$, as described in the next subsection. The model $\mathcal{D}$ is a natural extension of the *improved failures model* of [7, 15] to the language with label-passing (see the remark at the end of Sect. 1.3); thus the construction of $\mathcal{D}$ gives a counterexample to Hennessy's conjecture that the improved failures model is inadequate for modeling communicating processes with value-passing when the set of passed-values is infinite (see [14, pp. 234–235]).

1.1 Outline of the Full Abstractness Proof

There are two frameworks for denotational semantics: the *cpo framework* [37] and the *metric framework* [3], and each of the two has its own advantages and disadvantages (see Remark 1 below); the full abstractness result of this paper is established by combining the respective advantages of the two frameworks.

Remark 1 **(i)** The cpo framework provides a basis for a wide range of semantic models including weak and strong models, but it does not provide much support for establishing equivalence between a denotational and operational models.

(ii) The metric framework provides a powerful and uniform method (by means of Banach's fixed-point theorem) for establishing equivalence between a denotational and operational models (see [23] for a general account of this method, and see [3, 2, 8, 18, 20, 34] for applications of the method to various languages). Furthermore, a metric model can be automatically derived, under certain conditions, from the transition rules by which operational models are defined (see [17, 19, 35]). This method, however, is applicable only to strong models. ∎

[1] On the other hand, we call a semantic model *strong* when the model does abstract from internal moves.

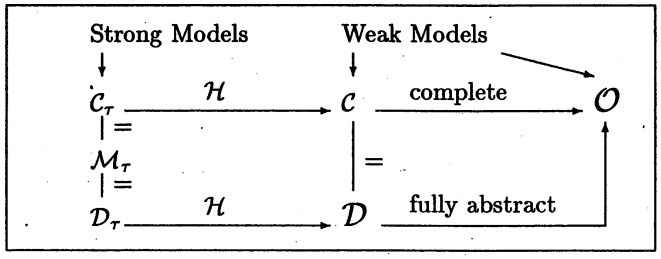


Figure 2: The scheme for establishing full abstractness results

By combining the respective advantages of the cpo and metric frameworks for denotational semantics, we obtain the following general scheme (i)–(iii) for establishing full abstractness results. (The scheme is illustrated by Fig. 2, where the expression $\mathcal{C}_\tau \xrightarrow{\mathcal{H}} \mathcal{C}$ denotes that $\mathcal{H} \circ \mathcal{C}_\tau = \mathcal{C}$.) This scheme has been applied in [18, Sect. 6] to a simple language for *pure* processes. (By "pure processes" we mean processes which can only communicate by simultaneously performing synchronization actions; see [14] for this terminology.) The full abstractness result reported in this paper is established by applying the scheme to a more complex language CCS^{++} with label-passing.

(i) Given a language $\mathcal{L}$ and an operational model $\mathcal{O}$ which serves as a behavioral criterion for assessing semantic models for $\mathcal{L}$, it may be possible to define a strong operational model $\mathcal{C}_\tau$ and a hiding function $\mathcal{H}$ and to show, by an operational analysis of program behaviors, that the model $\mathcal{C}$ defined by $\mathcal{C} = \mathcal{H} \circ \mathcal{C}_\tau$ is less abstract than $\mathcal{O}$ and *complete* with respect to $\mathcal{O}$ in the sense that the congruence induced by $\mathcal{O}$ is finer than the equivalence induced by $\mathcal{C}$. Thus it is sufficient to construct a denotational model $\mathcal{D}$ which is equivalent to $\mathcal{C}$, since the equivalence and the congruence induced by a denotational model coincide with each other. Such a denotational model $\mathcal{D}$ may be constructed as follows.

(ii) The metric framework can be conveniently used to construct a cms-based denotational model $\mathcal{M}_\tau$ and to establish the equivalence between $\mathcal{M}_\tau$ and $\mathcal{C}_\tau$, when these models are *strong* (see Remark 1(ii)).

(iii) When the semantic domain underlying $\mathcal{M}_\tau$ is both a cpo and a cms, we can construct a strong order-theoretic model $\mathcal{D}_\tau$ which is equivalent to $\mathcal{M}_\tau$; furthermore when $\mathcal{H}$ is a continuous homomorphism from the underlying algebra of $\mathcal{D}_\tau$ to the *weak* version of the algebra, we can also construct a weak order-theoretic model $\mathcal{D}$ such that $\mathcal{D} = \mathcal{H} \circ \mathcal{D}_\tau$; thus we can obtain the equivalence between $\mathcal{C}$ and $\mathcal{D}$, thereby establishing the *full abstractness* of $\mathcal{D}$ with respect to $\mathcal{O}$.

1.2 Comparison with Other Approaches to Full Abstractness Proofs

To establish the full abstractness of a denotational $\mathcal{D}$ with respect to a given operational model $\mathcal{O}$, we have to establish detailed connections between $\mathcal{D}$ and $\mathcal{O}$. However, it is in general very difficult to establish such connections directly, because the denotational method by which $\mathcal{D}$ is defined is very different from the operational (or transitional) method by which $\mathcal{O}$ is defined.

The following approach is taken to overcome this difficulty in this paper as well as in the other papers (such as [12, 14, 30]) treating the full abstractness problem in the context weak semantic models for concurrent languages. First, another operational model $\mathcal{C}$, called an *intermediate model*, is introduced, and it is shown to be equivalent to $\mathcal{D}$; then close connections between $\mathcal{C}$ and $\mathcal{O}$ are established by operational analysis of program behaviors, and thereby the desired connections between $\mathcal{D}$ and $\mathcal{O}$ are obtained.

Thus, the difficulty is reduced to the problem of establishing the equivalence between $\mathcal{C}$ and $\mathcal{D}$. Our approach to establishing the equivalence is different from those taken in the other papers (such as

[12, 14, 30]): We establish the equivalence by constructing an strong denotational $\mathcal{D}_\tau$ and operational one $\mathcal{C}_\tau$ such that $\mathcal{D} = \mathcal{H} \circ \mathcal{D}_\tau$ and $\mathcal{C} = \mathcal{H} \circ \mathcal{C}_\tau$ and by showning that both $\mathcal{D}_\tau$ and $\mathcal{C}_\tau$ are the fixed-point of the same higher-order contraction (which has a unique fixed-point by Banach's fixed-point theorem). In the other papers (such as [12, 14, 30]), on the other hand, this equivalence is established by a detailed operational analysis of the intermediate model $\mathcal{C}$, typically by showing that the preorder induced by $\mathcal{C}$ is *algebraic* (see [12, Sect. 4.5]). Roughly, our approach is denotational, whereas the other approaches are operational.

We believe our approach is more easily applicable to various languages, since there is a general method, which is insensitive to language constructs under a certain general condition, to construct the auxiliary models $\mathcal{D}_\tau$ and $\mathcal{C}_\tau$ and to establish their equivalence (see [17, 19]), whereas the operational analyses (of $\mathcal{C}$) needed in the other approaches are ad hoc and are sensitive to language constructs treated. Moreover, such operational analyses may involve crucial errors, which are difficult to detect in the absence of general methods (see Remark 3 for such an error found in [30]).

1.3 Connections with Other Full Abstractness Results

A similar full abstractness result has been established by Bergstra, Klop, and Olderog for a language without recursion or internal moves [5]. Moreover, Rutten investigated the semantics of a concurrent language for pure processes in the framework of cms's, and showed that the failures model is fully abstract with respect to a *strong linear semantics* $\mathcal{O}_{\mathrm{L}}$ [33]. The full abstractness of $\mathcal{D}$ with respect to $\mathcal{O}$, expressed by (1) above, is an extension of the result of Bergstra et al. to a language with recursion, internal moves, and label-passing. Also it is an extension of Rutten's result to the case of *weak semantics* (from strong semantics) and to a language with label-passing. The result reported in this paper is an extension of the result of [18, Chap. 6], where a similar full abstractness result has been given for a concurrent language for *pure* processes. (Similar full abstractness results have also been given in [16], but there all the models are constructed in a purely operational framework.)

In [14], Hennessy and Ingófsdóttir proposed a denotational model for a minor variant of full CCS, which is slightly simpler than CCS^{++} because full CCS features value-passing but does not feature label-passing. Their model is closely related to our model $\mathcal{D}$, but the machinery used for constructing their model is quite different from that used in this paper: The semantic domain in [14] is abstractly constructed, along the lines of [32], as a solution of a system of reflexive domain equations (which are rather complex with a powerdomain constructor appearing in a recursive way), while we use the simple failures domain defined in elementary set theory. And the behavioral criterion employed in [14] is based on the concept of *testing* and different from the one employed in this paper—we use the weak linear semantics as the behavioral criterion.

The model $\mathcal{D}$ is a natural extension of the *improved failures model* of [7, 15] to the language with label-passing, although we use the Smyth order [36] instead of the reverse set inclusion used in [7, 15], for the convenience in relating operational and denotational semantics.[2] (Note that in [7, 15], the improved failure model is given in a purely denotational framework, with no comparisons with operational models.)

2 Language CCS^{++}

In this section, we define the language a CCS-like language, CCS^{++}, with a *label-passing* mechanism. (Here *labels* are used as the names of communication channels through which interactions between processes take place.) This language is essentially the same as the one proposed in [1]; it is an extension of full CCS [27] and contains *label-passing*—an important feature of π-calculus [29]— in addition to the CCS constructs: *inputs*, *outputs*, *parallel composition with value-passing*, *nondeterministic choice*, *action restriction*, and *recursion*.

We define the language in the framework of typed λ-calculus with μ-notation as the version of PCL given in [11]; we take the types $\mathcal{V}$ (of values), $\mathcal{B}$ (of Booleans), and $\mathcal{P}$ (of process) as ground

[2] The original failures model [6] also use the reverse set inclusion as the ordering between failure sets.

types, and we restrict the composite types to $\mathcal{B}^{(1)} = (\mathcal{V} \to \mathcal{B})$ (the type of *predicates* or *parameterized Booleans* with one parameter of type $\mathcal{V}$) and $\mathcal{P}^{(1)} = (\mathcal{V} \to \mathcal{B})$ (the type of *parameterized processes* with one parameter of type $\mathcal{V}$).[3]

As a preliminary to the definition of the language CCS^{++}, we define several basic sets.

Definition 1 **(1)** It is assumed that a common domain $\mathbf{V}$ of *values* and *labels* is given. (Here *labels* serve as the names of communication channels through which *values* are passed.) Let $(b \in) \mathbf{B} = \{\mathbf{true}, \mathbf{false}\}$ be the domain of *Booleans*.

(2) We put $\mathbf{V}! = \{v! |\ v \in \mathbf{V}\}$ and $\mathbf{V}? = \{v? |\ v \in \mathbf{V}\}$. The set $(c \in)$ **Out** of *output actions* is defined by $\mathbf{Out} = \mathbf{V}! \times \mathbf{V} = \{(v!, v') |\ v, v' \in \mathbf{V}\}$. Likewise, we define the set $(c \in)$ **In** of *input actions* by $\mathbf{In} = \mathbf{V}? \times \mathbf{V} = \{(v?, v') |\ v, v' \in \mathbf{V}\}$. From these, the set $(c \in)$ $\mathbf{C}$ of *communication actions* is defined by $\mathbf{C} = \mathbf{Out} \cup \mathbf{In}$. We define the set $(a \in)$ $\mathbf{C}_\tau$ of *actions* by $\mathbf{C}_\tau = \mathbf{C} \cup \{\tau\}$.

(3) Let $\mathbf{Var}_\mathcal{V}$ be the set of variables of type $\mathcal{V}$, $(X \in)$ $\mathbf{Var}_\mathcal{P}$ the set of variables of type $\mathcal{P}$, and $(\xi \in)$ $\mathbf{Var}_\mathcal{P}^{(1)}$ the set of variables of type $\mathcal{P}^{(1)}$. We put $\mathbf{Var}_\mathcal{P}^* = \mathbf{Var}_\mathcal{P} \cup \mathbf{Var}_\mathcal{P}$, and define the set $(Z \in)$ **Var** of variables by $\mathbf{Var} = \mathbf{Var}_\mathcal{V} \cup \mathbf{Var}_\mathcal{P}^*$.

(4) We assume that a set $(E \in)$ $\mathcal{E}$ of *values expressions* and a set $(G \in)$ $\mathcal{G}$ of *Boolean expressions* are given. Here it is not necessary to specify the syntax for $\mathcal{E}$ and $\mathcal{G}$; we only postulate the following conditions (i)–(iii) concerning $\mathcal{E}$ and $\mathcal{G}$, for convenience in defining semantic models in later sections: **(i)** $\mathbf{Var}_\mathcal{V} \subseteq \mathcal{E}$, $\mathbf{V} \subseteq \mathcal{E}$, and $\mathbf{B} \subseteq \mathcal{G}$. **(ii)** For $E_0, E_1 \in \mathcal{E}$, "$(E_0 = E_1)$" $\in \mathcal{G}$. **(iii)** For $G_0, G_1 \in \mathcal{G}$, "$\neg(G_0)$", "$(G_0 \wedge G_1)$" $\in \mathcal{G}$.

We define the set $\mathcal{G}^{(1)}$ of *parameterized Boolean expressions* by

$$(H \in)\ \mathcal{G}^{(1)} = \{\text{"}(\lambda x.\ G)\text{"} |\ x \in \mathbf{Var}_\mathcal{V} \wedge G \in \mathcal{G}\}. \blacksquare$$

In term of the basic sets given above, the language CCS^{++} is defined by:

Definition 2 (Language CCS^{++})

(1) First, we define a set $(S \in)$ $\tilde{\mathcal{L}}$ of terms of type $\mathcal{P}$, simultaneously with the set of terms $(T \in)$ $\tilde{\mathcal{L}}^{(1)}$ of type $\mathcal{P}^{(1)}$, by the following BNF grammar:

$$\begin{cases} S ::= \mathbf{0} \mid \mathbf{out}(E_0, E_1, S) \mid \mathbf{in}(E, H, T) \mid (S_0 \parallel S_1) \mid (S_0 + S_1) \mid \\ \qquad \partial_C(S) \mid \mathbf{if}(G, S_0, S_1) \mid \mathbf{ap}(T, E) \mid X \mid (\mu X.\ S), \\ T ::= (\lambda x.\ S) \mid \xi \mid (\mu \xi.\ T), \end{cases}$$

where X (resp. ξ) ranges over $\mathbf{Var}_\mathcal{P}$ (resp. $\mathbf{Var}_\mathcal{P}^{(1)}$), E ranges over $\mathcal{E}$, G ranges over $\mathcal{G}$, H ranges over $\mathcal{G}^{(1)}$, and C ranges over $\wp(\mathbf{C})$.

Below we give a brief description of each of the constructs:

(i) The constant $\mathbf{0}$ resprests *inaction* in the sense of CCS.

(ii) The construct $\mathbf{out}(E_0, E_1, S)$. represents *output* the value E_1 through the channel E_0; this is a natural extension of the CCS construct "$\bar{c}E_1.\ S$" of CCS with the label c corresponding to E_0.

(iii) The construct $\mathbf{in}(E_0, (\lambda x.\ G), T)$ represents *selective input* through the channel E_0 of those values satisfying the predicate $(\lambda x.\ G)$. This is an analogue of the LOTOS construct for *value-acceptance with a selection predicate* (see [10] or [38, Sect. 3.5.2]). For a motivation for this combinator, see Remark 2(2) below.

[3] We can easily extend the language so as to include such processes as take several values as their parameters, without any nontrivial changes of the definition of semantics in later sections; it is only for simplicity that we treat only processes with zero or one parameter. However, it will cause a fundamental change in the definition of the semantics, if we include higher-order processes, such processes as take processes as their parameters.

(iv) The combinator $\|$ represents is *parallel composition* of CCS.

(v) The combinator $+$ represents *external choice* (or *general choice*) of CSP (see Sect. 3.3 of [15]). The transition rules for this combinator is slightly different from those for the choice combinator of CCS (see rule (iv) in Definition 9(1)). For a motivation for this combinator, see Remark 2(1) below.

(vi) The construct $\partial_C(S)$ represents the restriction of those actions of S which are contained in C; The transition rule for this construct is slightly different from those for the restriction construct "$S \setminus C$" of CCS (see rule (vii) in Definition 9(1)). For a motivation for this construct, see Remark 2(3) below.

(vii) The construct $\mathbf{if}(B, S_0, S_1)$ is the same as the conditional construct of CCS.

(viii) The construct $\mathbf{ap}(T, E)$ is the *application* of the *parameterized process* T to the actual parameter E. We sometimes write $T(E)$ instead of $\mathbf{ap}(T, E)$.

(ix) The μ-notation $(\mu X.\ S)$ represents a recursively defined process whose body is S.

(x) Likewise, the μ-notation $(\mu\xi.\ T)$ represents a recursively defined parameterized process whose body is T.

We will say that a variable $X \in \mathbf{Var}_{\mathcal{P}}$ is *guarded* in an expression $S' \in \tilde{\mathcal{L}}$, when each occurrence of X occurs in a subexpression of the form $\mathbf{out}(E, E', S'')$ or $\mathbf{in}(E, (\lambda x.\ G), T)$. Likewise we say that a variable $\xi \in \mathbf{Var}_{\mathcal{P}}^{(1)}$ is *guarded* in an expression $T' \in \tilde{\mathcal{L}}^{(1)}$, when each occurrence of ξ occurs in a subexpression of the form $\mathbf{out}(E, E', S'')$ or $\mathbf{in}(E, (\lambda x.\ G), T)$.

The set $(S \in)\ \mathcal{L}$ of statements of the language CCS^{++} is defined to be the set of $S \in \tilde{L}$ satisfying the following *guardedness condition*:

For each subexpression $(\mu X.\ S')$ (resp. $(\mu\xi.\ T)$) of S, the variable X (resp. ξ) is guarded in S' (resp. in T).	(2)

Likewise we define the set $(T \in)\ \mathcal{L}^{(1)}$ of parameterized statements of the language CCS^{++} to be the set of $T \in \tilde{L}^{(1)}$ satisfying the above guardedness condition (2).

Let us put $(U \in)\ \mathcal{L}^* = \mathcal{L} \cup \mathcal{L}^{(1)}$.

(2) The constructs "$(\lambda x.\ \cdots)$", "$(\mu X.\ \cdots)$", and "$(\mu\xi.\ \cdots)$" have the usual binding property. For $U \in \mathcal{L}^*$, let $\mathrm{FV}(U)$ be the set of elements of $\mathbf{Var}$ which have a free occurrence in U.

For $\mathcal{Y} \subset \wp(\mathbf{Var})$, let us define $\mathcal{L}[\mathcal{Y}]$ and $\mathcal{L}^{(1)}[\mathcal{Y}]$ by $\mathcal{L}[\mathcal{Y}] = \{S \in \mathcal{L} |\ \mathrm{FV}(S) \subseteq \mathcal{Y}\}$ and by $\mathcal{L}^{(1)}[\mathcal{Y}] = \{T \in \mathcal{L}^{(1)} |\ \mathrm{FV}(T) \subseteq \mathcal{Y}\}$., respectively. Then $\mathcal{L}[\emptyset]$ (resp. $\mathcal{L}^{(1)}[\emptyset]$) is the set of *closed* statements (resp. the set of of *closed* parameterized statements). Let us use s (resp. t) as a variable ranging over $\mathcal{L}[\emptyset]$ (resp. $\mathcal{L}^{(1)}[\emptyset]$). Let us put $(u \in)\ \mathcal{L}^*[\emptyset] = \mathcal{L} \cup \mathcal{L}^{(1)}$. For $Z \in \mathbf{Var}$, we write $\mathcal{L}[Z]$ instead of $\mathcal{L}[\{Z\}]$.

Further we put $(e \in)\ \mathcal{E}[\emptyset] = \{E \in \mathcal{E} |\ \mathrm{FV}(E) = \emptyset\}$, $(g \in)\ \mathcal{G}[\emptyset] = \{G \in \mathcal{G} |\ \mathrm{FV}(G) = \emptyset\}$, and $(h \in)\ \mathcal{G}^{(1)}[\emptyset] = \{H \in \mathcal{G}^{(1)} |\ \mathrm{FV}(H) = \emptyset\}$. ∎

Remark 2 (1) We can adopt the choice combinator of CCC instead of "+" (namely we can remove "+" and introduce another binary combinator with the same transition rules as those for the choice combinator of CCC) and preserve the full abstractness result, by slightly modifying the definition of the semantic domain along the lines of [4, 16], [16, Chapter 6]; it is only for simplicity that we adopt the combinator "+" for external choice instead of the choice combinator of CCC.

(2) Clearly we obtain richer expressive power by adopting the construct $\mathbf{in}(E_0, (\lambda x.\ G), T)$ for selective input instead of a natural extension (which would be represented by $\mathbf{in}(E_0, T)$ without the selection predicate) the CCS construct "$cx.S$" for unselective input. Such richer expressive power is convenient for describing communication systems (see [38, Sect. 3.5.2]). By using this

rich expressive power, we can establish the full abstractness of the denotational model $\mathcal{D}$ given in Sect. 5; it is not known whether or not the full abstractness can be established if we replace the construct $\mathbf{in}(E_0, (\lambda x.\ G), T)$ for selective input by the simper construct $\mathbf{in}(E_0, T)$ for unselective input. (Note that there is no difficulty in constructing a denotational model by slightly simplifying $\mathcal{D}$, for a language which has $\mathbf{in}(E_0, T)$ instead of $\mathbf{in}(E_0, (\lambda x.\ G), T)$.)

(3) The restriction construct $S \setminus C$ of CCS represents the restriction of those actions of S which are contained either in C or its complement $\overline{C}$; In this sense the restriction construct of CCS is more unselective than our restriction construct. By using this type restriction, we can establish the full abstractness of the denotational model $\mathcal{D}$; again it is not known whether or not the full abstractness can be established if we replace the construct $\partial_C(S)$ by the more unselective restriction construct of CCS. (Note that there is no difficulty in constructing a denotational model by slightly modifying $\mathcal{D}$, for a language which has the CCS construct $S \setminus C$ instead of $\partial_C(S)$.)

(4) The grammar given in Definition 2 is sufficient to construct a parser of the language, except for the construct $\partial_C(S)$; in order to construct a parser, it is necessary to introduce some syntax for specifying the suffix C in $\partial_C(S)$. A possible way is to introduce two constructs "$\partial!(E, (\lambda x.\ B), S)$" and "$\partial?(E, (\lambda x.\ B), S)$" which correspond to "$\partial_{\{([\![E]\!]!, v) \mid [\![B[v/x]]\!] = \mathbf{true}\}}(S)$" and "$\partial_{\{([\![E]\!]?, v) \mid [\![B[v/x]]\!] = \mathbf{true}\}}(S)$", respectively, where $[\![E]\!]$ (resp. $[\![B[v/x]]\!]$) is the evaluation of the expression $[\![E]\!]$ (resp. $[\![B[v/x]]\!]$) (see the beginning of Sect. 4 for more explanation of the evaluation mechanism $[\![\cdot]\!]$). ∎

We can characterize $\mathcal{L}^*$ as a subset of the set of terms generated by a many-sorted signature **Fun** (given in Definition 3 below) together with λ-notation and μ-notation. Thus, along the usual lines of denotational semantics, we can define a denotational model by giving a cpo as an underling domain and by giving interpretations of the elements of **Fun** as continuous functions on the cpo (see [11]); in Sect. 5 we will define the denotational model $\mathcal{D}$ for CCS^{++} in this way.

Definition 3 Let **Fun** be the set of constants and combinators of $\mathcal{L}$, and for $(i, j, k, \ell, m) \in \omega^5$, let $\mathbf{Fun}_{(i,j,k,\ell,m)}$ be the set of elements with arity $\mathcal{V}^i \cdot \mathcal{B}^j \cdot (\mathcal{B}^{(1)})^k \cdot \mathcal{P}^\ell \cdot (\mathcal{P}^1)^m$. Namely, we define $\mathbf{Fun}_{(i,j,k,\ell,m)}$ as follows:

(i) $\mathbf{Fun}_{(0,0,0,0,0)} = \{\text{"}\mathbf{0}\text{"}\}$, **(ii)** $\mathbf{Fun}_{(2,0,0,1,0)} = \{\text{"}\mathbf{out}\text{"}\}$, **(iii)** $\mathbf{Fun}_{(1,0,1,0,1)} = \{\text{"}\mathbf{in}\text{"}\}$,

(iv) $\mathbf{Fun}_{(0,0,0,1,0)} = \{\text{"}\partial_C\text{"} \mid C \in \mathbf{C}\}$, **(v)** $\mathbf{Fun}_{(0,0,0,1,0)} = \{\text{"}\|\text{"}, \text{"}+\text{"}\}$,

(vi) $\mathbf{Fun}_{(0,1,0,2,0)} = \{\text{"}\mathbf{if}\text{"}\}$, **(vii)** $\mathbf{Fun}_{(1,0,0,0,1)} = \{\text{"}\mathbf{ap}\text{"}\}$,

(viii) $\mathbf{Fun}_{(i,j,k,\ell,m)} = \emptyset$ for the other indexes (i, j, k, ℓ, m). ∎

3 Preliminary Definitions

In this section we fix basic mathematical notation and introduce several notions concerning domains of sequences.

Definition 4 Let A and B be sets.

(1) The set of natural numbers is denoted by ω, and each number $n \in \omega$ is identified with the set $\{i \in \omega \mid 0 \leq i < n\}$. The *powerset* of A (resp. the collection of finite subsets of A) is denoted by $\wp(A)$ (resp. by $\wp_{\text{fin}}(A)$).

(2) The set of functions from A to B is denoted by $(A \to B)$ or by B^A. The domain and range of a function f are denoted by $\text{dom}(f)$ and $\text{ran}(f)$, respectively. For a variable x and an expression $E(x)$, the λ-expression $(\lambda x \in A.\ E(x))$ is used to denote the function which maps $x \in A$ to $E(x)$. We sometimes write $\langle E(x) \rangle_{x \in A}$ or $\langle E(x) \mid x \in A \rangle$ for $(\lambda x \in A.\ E(x))$.

(3) The empty sequence is denoted by ϵ. The sequence consisting of $a_0, \cdots, a_{n-1} \in A$ is denoted by $\langle a_0, \cdots, a_{n-1} \rangle$.[4] The set of finite sequences of elements of A is denoted by $A^{<\omega}$, and $A^+ = A^{<\omega} \setminus \{\epsilon\}$. The set of finite or infinite sequences of elements of A is denoted by $A^{\leq\omega}$. Each sequence $w \in A^{\leq\omega}$ is regarded as a *function* whose domain is a member of $\omega \cup \{\omega\}$. Thus, the *length* of w is $\mathrm{dom}(w)$, which we refer to as $\mathrm{lgt}(w)$. (We have $w = \langle w(i) \rangle_{i \in \mathrm{lgt}(w)}$ by definition.)

(4) For $w_1 \in A^{<\omega}$, $w_2 \in A^{\leq\omega}$, let $w_1 \cdot w_2$ denote the *concatenation* of w_1 and w_2. And for $p_1 \subseteq A^{<\omega}$ and $p_2 \subseteq A^{\leq\omega}$, let $p_1 \cdot p_2 = \{w_1 \cdot w_2 |\ w_1 \in p_1 \wedge w_2 \in p_2\}$. For $p \subseteq A^{\leq\omega}$ and $w \in A^{<\omega}$, let $p[w] = \{\tilde{w} \in A^{\leq\omega} |\ w \cdot \tilde{w} \in p\}$. The *prefix order* $\preceq$ and the *strict prefix order* $\prec$ are defined as usual: For $w_1, w_2 \in A^{\leq\omega}$, $w_1 \preceq w_2$ iff $w_1 = w_2 \restriction \mathrm{dom}(w_1)$, and $w_1 \prec w_2$ iff $w_1 \preceq w_2 \wedge w_1 \neq w_2$.

(5) We define the set $A \triangleright B$ of finite or infinite sequences of elements of A ending with an element of B (when finite) by $A \triangleright B = (A^{<\omega} \cdot B) \cup A^{\omega}$.

(6) Let X be a topological space. The *closure* of a subset $Y \subseteq X$ is denoted by Y^{cl}. The collection of *closed* subsets of X is denoted by $\wp_{\mathrm{cl}}(X)$, and the the set of *nonempty* subsets of X is denoted by $\wp_{\mathrm{n}}(X)$.

Hereafter, we use the convention that $\wp_{\cdots}(X)$ denotes the set of all subsets (of X) having the property (or properties) indicated by the suffix $\cdots$. Thus, for example, $\wp_{\mathrm{ncl}}(X)$ denotes the set of all *nonempty* and *closed* subsets of X. ∎

Definition 5 Let A and B disjoint nonempty sets not containing $\bot$. We put $B_\bot = B \cup \{\bot\}$.

(1) A function $\mathrm{strip}(\cdot) : (A \triangleright B_\bot) \to A^{\leq\omega}$ is defined as follows: For $q \in (A \triangleright B_\bot)$, let $\mathrm{strip}(q) = q \restriction (\mathrm{lgt}(q) - 1)$ if $\mathrm{lgt}(q) < \omega$, and otherwise let $\mathrm{strip}(q) = q$. The *stream order* $\sqsubseteq$ on $(A \triangleright B_\bot)$ is defined as follows: For $q_1, q_2 \in (A \triangleright B_\bot)$, let $q_1 \sqsubseteq q_2$ if either $q_1 = q_2$ or $(q_1 \in A^{<\omega} \cdot \{\langle\bot\rangle\}$ and $\mathrm{strip}(q_1) \preceq \mathrm{strip}(q_2))$. For each $n \in \omega$, the *n-th projection function* $(\cdot)^{[n]} : (A \triangleright B_\bot) \to A^{<\omega} \cdot B_\bot$ is defined as follows: For $q \in (A \triangleright B_\bot)$, let $q^{[n]} = q$ if $\mathrm{lgt}(q) \leq n$, and otherwise let $q^{[n]} = (q \restriction n) \cdot \langle\bot\rangle$. And for $p \in \wp(A \triangleright B_\bot)$, let $p^{[n]} = \{q^{[n]} |\ q \in p\}$.

(2) A distance function d on $(A \triangleright B_\bot)$ is defined in terms of the projection functions as follows: For $q_1, q_2 \in (A \triangleright B_\bot)$, let $d(q_1, q_2) = \inf\{(1/2)^n |\ n \in \omega \wedge (q_1)^{[n]} = (q_2)^{[n]}\}$.

Likewise, a distance function $\tilde{d}$ on $\wp(A \triangleright B_\bot)$ is defined as follows:[5] For $p_1, p_2 \in \wp(A \triangleright B_\bot)$, let $\tilde{d}(p_1, p_2) = \inf\{(1/2)^n |\ n \in \omega \wedge (p_1)^{[n]} = (p_2)^{[n]}\}$. ∎

Definition 6 Let A and B be disjoint nonempty sets not containing $\bot$, and let d and $\tilde{d}$ be defined as in Definition 5.

(1) A binary relation $\sqsubseteq_{\mathrm{s}}$, the *Smyth order* on $\wp(A \triangleright B_\bot)$, is defined as follows: For $p_1, p_2 \in \wp(A \triangleright B_\bot)$, let $p_1 \sqsubseteq_{\mathrm{s}} p_2$ if $\forall q_2 \in p_2, \exists q_1 \in p_1 [\ q_1 \sqsubseteq q_2\ ]$.

(2) We will say that $p \in \wp(A \triangleright B_\bot)$ is *flat* if $\forall q, q' \in p [\ q \sqsubseteq q' \Rightarrow q = q'\ ]$. We put

$$\wp_{\mathrm{f}}(A \triangleright B_\bot) = \{p \in \wp(A \triangleright B_\bot) |\ p \text{ is flat}\}.$$

(3) We define $\wp_{\mathrm{fcl}}(A \triangleright B_\bot)$ and $\wp_{\mathrm{nfcl}}(A \triangleright B_\bot)$, according to the convention in Definition 4(6).

(4) Let $\wp_{\mathrm{co}}(A \triangleright B_\bot) = \{p \in \wp(A \triangleright B_\bot) |\ p \text{ is compact}\}$, where the notion of *compactness* is induced by the metric d. We define $\wp_{\mathrm{fco}}(A \triangleright B_\bot)$, $\wp_{\mathrm{nfco}}(A \triangleright B_\bot)$, and $\wp_{\mathrm{nfco}}(A \triangleright B_\bot)$ as in part (3).

(5) Let $X \in \wp(A \triangleright B_\bot)$ and $q \in X$. We will say that q is *minimal* in X when $\neg\exists q' \in X [\ q' \sqsubseteq q \wedge q' \neq q\ ]$. Let $\mathrm{Min}(X) = \{q \in X |\ q \text{ is minimal in } X\}$.

---

[4] We do not abuse notation by writing a_0 to refer to $\langle a_0 \rangle$, because it will be confusing when we treat sequences consisting of sequences in later sections (cf. [15, Sect. 1.5] for similar strict notation).

[5] It is easily checked that $\tilde{d}$ coincides with the *Hausdorff distance* induced by d (for the definition of Hausdorff distance see [9]).

(6) The *limit* of a converging sequence $\langle p_n\rangle_{n\in\omega}$ in $(\wp_{\mathrm{cl}}(A \triangleright B_\perp), \tilde{d})$ is denoted by $\lim_n p_n$. ∎

Notation 1 Let $(X, \sqsubseteq, \perp)$, $(Y, \sqsubseteq', \perp')$ be cpo's.

(1) For a chain $\langle x_n\rangle_{n\in\omega}$ in X, the *lub* of $\{x_n\}_{n\in\omega}$ in X is denoted by $\bigsqcup_n x_n$. Let $[X \to Y]$ denote the set of *continuous* functions from X to Y.

(2) Let $(X, \sqsubseteq, \perp)$ be a cpo, $m \in \omega$, and $\langle x_n\rangle_{n\geq m} \in (\{n \in \omega |\ n \geq m\} \to X)$. We say that $\langle x_n\rangle_{n\geq m}$ *is a chain* to mean that $\langle x_{m+n}\rangle_{n\in\omega}$ is a chain; when $\langle x_n\rangle_{n\geq m}$ is a chain, we write $\bigsqcup_{n\geq m} p_n$ to denote $\bigsqcup_n p_{m+n}$. ∎

Definition 7 For $p \in \wp(\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp)$ and $w \in (\mathbf{C}_\tau)^{<\omega}$, let $p[w] = \{q \in (\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp) |\ w \cdot q \in p\}$. We put $\mathrm{act}(p) = \{a \in \mathbf{C}_\tau |\ p[\langle a\rangle] \neq \emptyset\}$. ∎

Definition 8 Let (X, d) and (Y, d') be metric spaces and ε be a nonnegative real number. We write $(X \to^\varepsilon Y)$ to denote the set $\{f \in (X \to Y) |\ \forall x, x' \in X[\ d'(f(x), f(x')) \leq \varepsilon \cdot d(x, x')\ ]\}$. In particular, we write $(X \to^1 Y)$ to denote the set of *nonexpansive* mappings from X to Y. ∎

4 The Operational Model $\mathcal{O}$

In this section, the operational model $\mathcal{O}$ is defined in terms of a transition system in the style of Plotkin [31].

To define the transition system, we assume that some *evaluation mechanism* $[\![\cdot]\!]$ mapping $e \in \mathcal{E}[\emptyset]$ (resp. $g \in \mathcal{B}[\emptyset]$) to $[\![e]\!] \in \mathbf{V}$ (resp. to $[\![g]\!] \in \mathbf{B}$) is given. For $h \equiv (\lambda x.\ G) \in \mathcal{G}^{(1)}[\emptyset]$, we define the evaluation $[\![h]\!]$ by $[\![h]\!] = (\lambda v \in \mathbf{V}.\ [\![G[v/x]]\!])$.

4.1 Transition System

Definition 9 (Transition Relations)

(1) The transition relations $\xrightarrow{a} \subseteq (\mathcal{L}[\emptyset])^2$ $(a \in \mathbf{C}_\tau)$ are defined as the smallest relations satisfying the following rules:

(i) $\mathbf{out}(e_0, e_1, s) \xrightarrow{([\![e_0]\!]!, [\![e_1]\!])} s$.

(ii) $\mathbf{in}(e, (\lambda x.\ G), t) \xrightarrow{([\![e]\!]?, v)} \mathbf{ap}(t, v)$, if $v \in \mathbf{V} \wedge [\![G[v/x]]\!] = \mathbf{true}$.

(iii) $\dfrac{s_1 \xrightarrow{c} s_1'}{\begin{array}{l}(s_1 + s_2) \xrightarrow{c} s_1' \\ (s_2 + s_1) \xrightarrow{c} s_1'\end{array}}$ $(c \in \mathbf{C})$. **(iv)** $\dfrac{s_1 \xrightarrow{\tau} s_1'}{\begin{array}{l}(s_1 + s_2) \xrightarrow{\tau} (s_1' + s_2) \\ (s_2 + s_1) \xrightarrow{\tau} (s_2 + s_1')\end{array}}$.

(v) $\dfrac{s_1 \xrightarrow{a} s_1'}{\begin{array}{l}(s_1 \parallel s_2) \xrightarrow{a} (s_1' \parallel s_2) \\ (s_2 \parallel s_1) \xrightarrow{a} (s_2 \parallel s_1')\end{array}}$ $(a \in \mathbf{C}_\tau)$. **(vi)** $\dfrac{s_1 \xrightarrow{(v!, v')} s_1',\ s_2 \xrightarrow{(v?, v')} s_2'}{\begin{array}{l}(s_1 \parallel s_2) \xrightarrow{\tau} (s_1' \parallel s_2') \\ (s_2 \parallel s_1) \xrightarrow{\tau} (s_2' \parallel s_1')\end{array}}$ $(v, v' \in \mathbf{V})$.

(vii) $\dfrac{s \xrightarrow{a} s'}{\partial_C(s) \xrightarrow{a} \partial_C(s')}$ $(a \notin C)$.

(viii) $\dfrac{s_1 \xrightarrow{a} (s)}{\mathbf{if}(g, s_1, s_2) \xrightarrow{a} (s)}$, if $[\![g]\!] = \mathbf{true}$. **(ix)** $\dfrac{s_2 \xrightarrow{a} (s)}{\mathbf{if}(g, s_1, s_2) \xrightarrow{a} (s)}$, if $[\![g]\!] = \mathbf{false}$.

(x) The following rule is called the *pre-evaluation rule*:

$$\frac{\mathbf{ap}(u, [\![e]\!]) \xrightarrow{a} s'}{\mathbf{ap}(u, e) \xrightarrow{a} s'} \quad (e \in \mathcal{E}[\emptyset]).$$

(xi) The following rule is called the λ*-rule*:

$$\frac{S[v/x] \xrightarrow{a} s'}{\mathbf{ap}((\lambda x.\ S), v) \xrightarrow{a} s'} \quad (v \in \mathbf{V}).$$

(xii) The following rule is called the *recursion rule*:

$$\frac{S[(\mu X.\ S)/X] \xrightarrow{a} s'}{(\mu X.\ S) \xrightarrow{a} s'}.$$

(xiii) The following rule is called the *parameterized recursion rule*:

$$\frac{\mathbf{ap}(T[(\mu\xi.\ T)/\xi], v) \xrightarrow{a} s'}{\mathbf{ap}((\mu\xi.\ T), v) \xrightarrow{a} s'} \quad (v \in \mathbf{V}).$$

(2) For $s \in \mathcal{L}[\emptyset]$, we put $\mathrm{act}(s) = \{a \in \mathbf{C}_\tau |\ \exists s' \in \mathcal{L}[\emptyset][\ s \xrightarrow{a} s'\ ]\}$. ∎

The transition system is *image-finite* in the following sense:

Proposition 1 *For every $s \in \mathcal{L}[\emptyset]$ and $a \in \mathbf{C}_\tau$, the* image-set $\{s' \in \mathcal{L}[\emptyset] |\ s \xrightarrow{a} s'\}$ *is finite.* ∎

Proof. By induction on the structure of s using the guardedness condition (2). ■

For $a \in \mathbf{C}_\tau$, let us define the *channel* of a, written $\mathrm{chan}(a)$, by

$$\mathrm{chan}(a) = \begin{cases} v! & \text{if } a = (v!, v'), \\ v? & \text{if } a = (v?, v'), \\ \tau & \text{otherwise.} \end{cases} \tag{3}$$

Then, the transition system is *bounded with respect to channels* in the following sense:

Proposition 2 *For every $s \in \mathcal{L}[\emptyset]$, the set* $\mathrm{chan}[\mathrm{act}(s)]$ *of channels is finite.* ∎

4.2 The Operational Model $\mathcal{O}$

In this subsection, we define the operational model $\mathcal{O}$ in terms of the transition relations $\langle \xrightarrow{a} |\ a \in \mathbf{C}_\tau \rangle$. As a preliminary to the definition, we introduce three other notions of transitions (as in [28]):

Definition 10 (1) For $w = (a_0, \ldots, a_{n-1}) \in (\mathbf{C}_\tau)^{<\omega}$, and $s, s' \mathcal{L}[\emptyset]$, we write $s \xrightarrow{w} s'$ to mean that there exist $s_0, \ldots, s_n \in \mathcal{L}[\emptyset]$ such that

$$s \equiv s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots s_{n-1} \xrightarrow{a_{n-1}} s_n \equiv s'.$$

(2) For $w \in \mathbf{C}^{<\omega}$ and $s, s' \in \mathcal{L}[\emptyset]$, we write $s \xRightarrow{w} s'$ to mean that

$$\exists \tilde{w} \in (\mathbf{C}_\tau)^{<\omega}[\ s \xrightarrow{\tilde{w}} s' \wedge w = (\tilde{w} \setminus \tau)\ ],$$

where $(w \setminus \tau)$ denote the result of erasing τ's in w.

(3) For $w \in (\mathbf{C}_\tau)^\omega$ and $s \in \mathcal{L}[\emptyset]$, we write $s \xrightarrow{w}$ to mean that

$$\exists \langle s_n \rangle_{n \in \omega} \in (\mathcal{L}[\emptyset])^\omega[\ s_0 = s \wedge \forall n \in \omega[\ s_n \xrightarrow{w(n)} s_{n+1}\ ]\ ].$$ ∎

By using the above definition, the operational model $\mathcal{O}$ is defined by:

Definition 11 (The Operational Model $\mathcal{O}$)

(1) The flattening function $\mathrm{Min}(\cdot)$ on $\wp(\mathbf{C}_\tau \triangleright \{\delta, \bot\})$ is defined as in Definition 6(6).

(2) First, we define an auxiliary model $\tilde{\mathcal{O}} : \mathcal{L}[\emptyset] \to \wp(\mathbf{C}_\tau \triangleright \{\delta, \bot\})$ as follows: For $s \in \mathcal{L}[\emptyset]$ and $\rho \in (\mathbf{C}_\tau \triangleright \{\delta, \bot\})$, we put $\rho \in \tilde{\mathcal{O}}[\![s]\!]$, if one of the following three conditions (i)–(iii) is satisfied:

Lemma 4.1 *Let φ, ψ be knowledge propositions including only disjunction and conjunction symbols. Then*

$$[\varphi \vee \psi] \div \rho^\sharp = [\varphi] \div \rho^\sharp \sqcap [\psi] \div \rho^\sharp$$

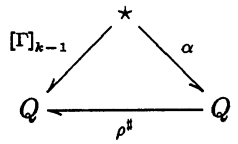
$$[\varphi \wedge \psi] \div \rho^\sharp = [\varphi] \div \rho^\sharp \sqcup [\psi] \div \rho^\sharp$$

Each agent infers from previous message and determines its next action and its next message to send. We provide a property which guarantees that agents reasonably infer proposition from some messages.

Definition 4.1 *Let $s \in Q$ and let φ be a knowledge proposition and $\Gamma \equiv \psi_1 \wedge \cdots \wedge \psi_m$ be a conjunctive knowledge propositions. We say that* agent i knows φ from the condition set Γ at s *if and only if*

$$s\delta_i \sqcap [\Gamma] \div \rho^\sharp \sqsubseteq [\varphi]$$

As mentioned in the previous section, the quotient relation is the greatest relation α satisfing the commutative diagram:



Lemma 4.2 *Let $s \in Q$. Assume that i knows φ from Γ at s. For every state t such that $(t, s) \in \rho$, if $t \models K_i\varphi$ then $s \models K_i\varphi$.*

Proof : Suppose that for every state $t \in Q$ such that $t \sqsubseteq s\rho^\sharp$, $t \sqsubseteq [\varphi] \div \delta_i$, that is

$$s\rho^\sharp \sqsubseteq [\varphi]_k \div \delta_i$$

As ρ and δ_i are commutative,

$$\begin{aligned} s &\sqsubseteq ([\varphi] \div \delta_i) \div \rho^\sharp \\ s &\sqsubseteq [\varphi] \div \rho^\sharp \delta_i \\ s &\sqsubseteq [\varphi] \div \delta_i \rho^\sharp \\ s &\sqsubseteq ([\varphi] \div \rho^\sharp) \div \delta_i \\ s\delta_i &\sqsubseteq [\varphi] \div \rho^\sharp \end{aligned}$$

From assumption it holds that

$$s\delta_i \sqcap [\varphi] \div \rho^\sharp \sqsubseteq [\psi]$$

Then we have $s\delta_i \sqsubseteq [\psi]$. □

Remark 4.1: If the transition relation ρ is reflexive, then it holds that $[\varphi] \div \rho \sqsubseteq [\varphi]$ for any knowledge proposition φ. □

Theorem 4.1 *Let the transition relation ρ be reflexive, and $[\Gamma] \sqsubseteq [\varphi]$ where Γ is a conjunctive knowledge proposition and φ is a knowledge proposition. For any transition $(t, s) \in \rho$, if $t \models K_i\Gamma$ then $s \models K_i\varphi$.*

Proof : Assume that $t \sqsubseteq [\Gamma] \div \delta_i$ for every $t \sqsubseteq s\rho^\sharp$, that is,

$$s\rho^\sharp \sqsubseteq [\Gamma] \div \delta_i.$$

As ρ and δ_i are commutative, we have

$$s\delta_i\rho^\sharp = s\rho^\sharp\delta_i \sqsubseteq [\Gamma]$$

so that $s\delta_i \sqsubseteq [\Gamma] \div \rho^\sharp$. From assumption

$$\begin{aligned} s\delta_i &= s\delta_i \sqcap [\Gamma] \div \rho^\sharp \\ &\sqsubseteq s\delta_i \sqcap [\varphi] \div \rho^\sharp \\ &\sqsubseteq s\delta_i \sqcap [\varphi] \end{aligned}$$

therefore $s\delta_i \sqsubseteq [\varphi]$, hence $s \sqsubseteq [\varphi] \div \delta_i$. □

Corollary 4.1 *(In the same condition of theorem.) For every transition $(t,s) \in \rho$, if $t \models K_i\varphi$ then $s \models K_i\varphi$ then $s \models K_i\varphi$.*

Proof : By theorem in the case of $\Gamma \equiv \varphi$. □

Proposition 4.1 *Let ρ be reflexive. For every transition $(t,s) \in \rho$, if $t \models K_i\varphi$ and $t \models K_i\psi$ then i knows $\varphi \vee \psi$ from $\varphi \wedge \psi$.*

Proof : By assumption we have

$$\begin{aligned} s\rho^\sharp &\sqsubseteq [K_i\varphi] \sqcap [K_i\psi] \\ &= ([\varphi] \div \delta_i) \sqcap ([\psi] \div \delta_i) \\ &= ([\varphi] \sqcap [\psi]) \div \delta_i. \end{aligned}$$

As ρ and δ_i commutes

$$\begin{aligned} s\delta_i &\sqsubseteq ([\varphi] \sqcap [\psi]) \div \rho^\sharp \\ &= [\varphi \wedge \psi] \div \rho^\sharp \\ &\sqsubseteq [\varphi \vee \psi] \div \rho^\sharp \\ &\sqsubseteq [\varphi \vee \psi] \end{aligned}$$

from assumption. Hence we have $s\delta_i \sqcap ([\varphi \wedge \psi] \div \rho^\sharp) \sqsubseteq [\varphi \vee \psi]$. □

参考文献

[CM86] K. M. Chandy and J. Misra. How processes learn. *Distributed Computing*, 1:40–52, 1986.

[FH88] R. Fagin and J. Y. Halpern. I'm ok if you're ok: On the notion of trusting communication. *Journal of Philosophical Logic*, 17:329–354, 1988.

[HM89] J. Y. Halpern and Y. Moses. Modelling knowledge and action in distributed systems. *Distributed Computing*, 3:159–177, 1989.

[HM90] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the Association for Computing Machinery*, 37(3):549–587, 1990.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

参考文献

[HZ89] J. Y. Halpern and L. D. Zuck. A little knowledge goes a long way: Simple knowledge–based derivation and correctness proofs for a family for protocols. Technical Report RJ 5857, IBM Research Division, 1989.

[Kaw90] Y. Kawahara. Pushout–complements and basic concepts of grammers in toposes. *Theoretical Computer Science*, 77:267–289, 1990.

[Kaw94] Y. Kawahara. Relational formalization of knowledge dynamics. draft, 1994.

[KM92] Y. Kawahara and Y. Mizoguchi. Categorical assertion semantics in topoi. *Advances in Software and Technology*, 4:137–150, 1992.

[LL90] L. Lamport and N. Lynch. Distributed computing: Models and methods. In van Leeuwen J., editor, *Handbook of theoretical computer science*, pages 1157–1199. Elsevier Science Publishers B.V., 1990.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[SS85] G. Schmidt and T. Ströhlein. Relation algebras: Concept of points and represetntability. *Discrete Mathematics*, 54:83–92, 1985.

[Tar41] A. Tarski. On the calculus of relations. THE JOURNAL OF SYMBOLIC LOGIC, 6(3), 1941.

(1) *The function* $(\lambda p \in \mathbf{P}.\ \widetilde{\mathrm{out}}(v,v',p))$ *is a nonexpansive and continuous unary operation on* $\mathbf{P}$, *i.e., one has* $(\lambda p \in \mathbf{P}.\ \widetilde{\mathrm{out}}(v,v',p)) \in (\mathbf{P} \to^1 \mathbf{P}) \cap [\mathbf{P} \to \mathbf{P}]$.

(2) *The function* $(\lambda p \in \mathbf{P}.\ \widetilde{\mathrm{out}}(v,v',p))$ *is contractive in the sense that*

$$\forall p_0, p_1 \in \mathbf{P}[\ \tilde{d}(\mathrm{out}(v,v',p_0), \mathrm{out}(v,v',p_1)) \leq (1/2) \cdot \tilde{d}(p_0,p_1)\ ].\ \blacksquare \tag{6}$$

Definition 15 We define $\widetilde{in} : \mathbf{V} \times \mathbf{B}^{(1)} \times \mathbf{P} \to \wp_{\mathrm{fcl}}(\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp)$ as follows: For $v \in \mathbf{V}$, $\eta \in \mathbf{B}^{(1)}$, and $\pi \in \mathbf{P}^{(1)}$,

$$\widetilde{\mathrm{in}}(v,\eta,\pi) = \{\langle C\rangle |\ C \in \wp(\mathbf{C}) \wedge C \cap (\{v?\} \times I) = \emptyset\} \cup \bigcup_{v' \in I} (\ \langle (v!,v')\rangle \cdot \pi(v')\ ). \tag{7}$$

where $I = \{v' \in \mathbf{V} |\ \eta(v') = \mathbf{true}\}$. $\blacksquare$

As for $\widetilde{\mathrm{out}}(v,v',p)$, we have the following useful property of $\widetilde{\mathrm{in}}(v,\eta,h)$.

Proposition 6 *Let* $v\mathbf{V}$ *and* $\eta \in (\mathbf{V} \to \mathbf{B})$.

(1) *The function* $(\lambda\pi \in \mathbf{P}^{(1)}.\ \widetilde{\mathrm{in}}(v,\eta,\pi))$ *is a nonexpansive and continuous function from* $\mathbf{P}^{(1)}$ *to* $\mathbf{P}$, *i.e., one has* $(\lambda\pi \in \mathbf{P}^{(1)}.\ \widetilde{\mathrm{in}}(v,\eta,\pi)) \in (\mathbf{P}^{(1)} \to^1 \mathbf{P}) \cap [\mathbf{P}^{(1)} \to \mathbf{P}]$.

(2) *The function* $(\lambda\pi \in \mathbf{P}^{(1)}.\ \widetilde{\mathrm{in}}(v,\eta,\pi))$ *is contractive in the sense that*

$$\forall \pi_0, \pi_1 \in \mathbf{P}^{(1)}[\ \tilde{d}(\widetilde{\mathrm{in}}(v,\eta,\pi_0), \widetilde{\mathrm{in}}(v,\eta,\pi_1)) \leq (1/2) \cdot \tilde{d}_{\mathrm{f}}(\pi_0,\pi_1)\ ].\ \blacksquare \tag{8}$$

5.3.2 Parallel Composition

The definitions of the semantic operations corresponding to parallel composition, action restriction, and external choice is more involved than the definitions in Sect. 5.3.1. First we need a preliminary definition.

Definition 16 For $p_0, p_1 \in \mathbf{P}$, let $\tilde{\|}_\perp(p_0,p_1) = \{\langle\perp\rangle\} \cap (p_0 \cup p_1)$, and

$$\begin{aligned}\tilde{\|}_\delta(p_0,p_1) = \{\ \langle C\rangle |\ C \in \wp(\mathbf{C}) \wedge \exists C_0, C_1 \in \wp(\mathbf{C})[\ \langle C_0\rangle \in p_0 \wedge \langle C_1\rangle \in p_1 \wedge \\ C \subseteq C_0 \cap C_1 \wedge \overline{(\mathbf{C} \setminus C_0)} \cap (\mathbf{C} \setminus C_1) = \emptyset.\}\ \blacksquare\end{aligned} \tag{9}$$

We will define a binary operation $\tilde{\|}$ on $\mathbf{P}$ so that the following proposition holds:

Proposition 7 *For* $p_0, p_1 \in \mathbf{P}$, *one has*

$$\tilde{\|}(p_0,p_1) = \tilde{\|}_\perp(p_0,p_1)\ \hat{\cup}\ \tilde{\|}_\delta(p_0,p_1)\ \hat{\cup}\ \tilde{\llbracket}(p_0,p_1)\ \hat{\cup}\ \tilde{\llbracket}(p_1,p_0)\ \hat{\cup}\ \tilde{\lfloor}(p_0,p_1)\ \hat{\cup}\ \tilde{\lfloor}(p_1,p_0), \tag{10}$$

where

$$\tilde{\llbracket}(p_0,p_1) = \bigcup_{a \in \mathrm{act}(p_0)} (\langle a\rangle \cdot \tilde{\|}(p_0[\langle a\rangle], p_1)), \quad \textit{and} \tag{11}$$

$$\begin{aligned}\tilde{\lfloor}(p_0,p_1) = \bigcup\{\ \langle\tau\rangle \cdot \tilde{\|}(p_0[\langle(v!,v')\rangle], p_1[\langle(v?,v')\rangle]) | \\ v,v' \in \mathbf{V} \wedge (v!,v') \in \mathrm{act}(p_0) \wedge (v?,v') \in \mathrm{act}(p_1)\}.\ \blacksquare\end{aligned} \tag{12}$$

Informally, we may suppose that $\tilde{\|}$ is defined by the above proposition, where the value $\tilde{\|}(p_0,p_1)$ is characterized in terms of $\|(p_0[w_0], p_1[w_1])$ with $w_0, w_1 \in (\mathbf{C}_\tau)^{<\omega}$. Formally, the operation $\tilde{\|}$ is defined as the unique fixed-point of a higher-order mapping $\Phi_\|$ defined by:

Definition 17 We define $\Phi_\| : (\mathbf{P}^2 \to \mathbf{P}) \to (\mathbf{P}^2 \to \wp(\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp))$ as follows: For $F \in (\mathbf{P}^2 \to \mathbf{P})$ and $(p_0,p_1) \in \mathbf{P}^2$, let

$$\begin{aligned}&\Phi_\|(F)(p_0,p_1) \\ &= \tilde{\|}_\perp(p_0,p_1)\ \hat{\cup}\ \tilde{\|}_\delta(p_0,p_1)\ \hat{\cup}\ \Phi_{\llbracket}(F)(p_0,p_1)\ \hat{\cup}\ \Phi_{\llbracket}(F)(p_1,p_0)\ \hat{\cup}\ \Phi_{\lfloor}(p_0,p_1)\ \hat{\cup}\ \Phi_{\lfloor}(p_1,p_0), \text{ where}\end{aligned} \tag{13}$$

$$\Phi_{\llbracket}(F)(p_0,p_1) = \bigcup_{a \in \mathrm{act}(p_0)} (\ \langle a\rangle \cdot F(p_0[\langle a\rangle], p_1)\ ), \text{ and} \tag{14}$$

$$\Phi_{\lfloor}(F)(p_0,p_1) = \bigcup\{\, \langle\tau\rangle \cdot F(p_0[\langle(v!,v')\rangle], p_1[\langle(v?,v')\rangle]) \mid v,v' \in \mathbf{V} \wedge (v!,v') \in \mathrm{act}(p_0) \wedge (v?,v') \in \mathrm{act}(p_1)\}. \blacksquare \tag{15}$$

The mapping $\Phi_{\|}$ is a monotonic contraction from $(\mathbf{P}^2 \to \mathbf{P})$ to itself, and preserves nonexpansiveness and (order-theoretical) continuity. (see Sect. 5.3.2 of [21] for the proof of this fact). We formally define $\tilde{\|}$ to be $\mathrm{fix}(\Phi_{\|})$, the unique fixed-point of $\Phi_{\|}$. By Corollary 1, we have $\tilde{\|} = \mathrm{fix}(\Phi_{\|}) = \lim_n((\Phi_{\|})^n(\tilde{\perp})) = \bigsqcup_n((\Phi_{\|})^n(\tilde{\perp}))$, where $\tilde{\perp} = (\lambda\vec{p} \in \mathbf{P}^2.\ \{\langle\perp\rangle\})$. Thus $\tilde{\|}$ is nonexpansive and (order-theoretically) continuous (i.e.,$\tilde{\|} \in (\mathbf{P}^2 \to^1 \mathbf{P}) \cap [\mathbf{P}^2 \to^1 \mathbf{P}]$), because the limit of nonexpansive functions is nonexpansive and the lub of continuous functions is continuous.

5.3.3 Action Restriction and External Choice

For every $C \in \wp(\mathbf{C})$, we define a unary operation $\tilde{\partial}_C \in (\mathbf{P} \to^1 \mathcal{P}) \cap [\mathbf{P} \to \mathcal{P}]$ corresponding to the combinator ∂_C; the formal definition of $\tilde{\partial}_C$ is similar to that of $\tilde{\|}$ but more simple (see Sect. 5.3.3 of [21] for the definition).

The binary operation $\tilde{+} :\in (\mathbf{P}^2 \to^1 \mathcal{P}) \cap [\mathbf{P}^2 \to \mathcal{P}]$ corresponding to the combinator '+' can be defined directly without using a higher-order mapping (see Sect. 5.3.4 of [21] for the definition).

5.3.4 Conditionals and Function Application

The definitions of the semantic operations corresponding to the construct $\mathbf{if}(\cdot,\cdot,\cdot)$ for conditionals and the construct $\mathbf{ap}(\cdot,\cdot)$ for function application are intuitively natural.

Definition 18 **(1)** We define $\widetilde{\mathrm{if}}(b) : \mathbf{B} \times \mathbf{P}^2 \to \mathbf{P}$ as follows: For $b \in \mathbf{B}$ and $p_0, p_1 \in \mathbf{P}$, let

$$\widetilde{\mathrm{if}}(b,p_0,p_1) = \begin{cases} p_0 & \text{if } b = \mathbf{true}, \\ p_1 & \text{if } b = \mathbf{false}. \end{cases} \tag{16}$$

(2) We define $\widetilde{\mathrm{apl}} : \mathbf{P}^{(1)} \times \mathbf{V} \to \mathbf{P}$ as follows: For $\pi \in \mathbf{P}^{(1)}$ and $v \in \mathbf{V}$, let $\widetilde{\mathrm{apl}}(\pi, v) = \pi(v)$. $\blacksquare$

5.4 The Denotational Model $\mathcal{D}$

On the basis of the cpo $(\mathbf{P}, \sqsubset_s)$, we define the denotational model $\mathcal{D}$ in terms of *weak* versions $\widehat{\mathrm{op}}$ of the semantic operations $\widetilde{\mathrm{op}}$ defined in the previous subsection. The weak operations $\widehat{\mathrm{op}}$ are defined as the compositions of the original operations $\widetilde{\mathrm{op}}$ and a hiding function $\mathcal{H}$. First we define $\mathcal{H}$.

5.4.1 The Hiding Function $\mathcal{H}$

The hiding function $\mathcal{H}$ is defined by:

Definition 19 **(1)** A function $h : (\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp) \to (\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp)$ is defined as follows: For $q = w \cdot \langle x\rangle$ with $(w,x) \in (\mathbf{C}_\tau)^{<\omega} \times \wp(\mathbf{C})_\perp$, let $h(q) = (w \setminus \tau) \cdot \langle x\rangle$, and for $q \in (\mathbf{C}_\tau)^\omega$, let

$$h(q) = \begin{cases} (\tilde{w} \setminus \tau) \cdot \langle\perp\rangle & \text{if } \exists\tilde{w}[\, w = \tilde{w} \cdot \langle\tau\rangle_{i\in\omega}\,], \\ h(q) = (w \setminus \tau) & \text{otherwise.} \end{cases}$$

(2) The hiding function $\mathcal{H} : \wp(\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp) \to \wp(\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp)$ is defined as follows: For $p \in \wp(\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp)$, let $\mathcal{H}(p) = \mathrm{Min}(h[p])$. $\blacksquare$

The monotonicity of $\mathcal{H}$ immediately follows from this definition:

Proposition 8 *The hiding function $\mathcal{H}$ is monotonic with respect to $\sqsubset_s$.* $\blacksquare$

Moreover, we have the following lemma:

Lemma 2 *The hiding function $\mathcal{H}$ is a continuous function from $\wp_{\mathrm{fcl}}(\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp)$ to itself.* $\blacksquare$

This lemma follows from the continuity of h on $\wp(\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp)$ by applying (a localized version of) the *lifting method* proposed by Meyer and de Vink (see Theorem 4.6 of [24]). See Lemma 3 of [21] for the proof.

5.4.2 Weak Versions of Semantic Operations

Definition 20 For $(i,j,k,\ell,m) \in \omega^5$ and $\mathrm{op} \in \mathbf{Fun}_{(k,\ell,m)}$, let $\widehat{\mathrm{op}} : \mathbf{V}^i \cdot \mathbf{B}^j \cdot (\mathbf{B}^{(1)})^k \cdot \mathbf{P}^\ell \cdot (\mathbf{P}^{(1)})^m \to \mathbf{P}$ be defined as follows: For $\vec{v} \in \mathbf{V}^i$, $\vec{b} \in \mathbf{B}^j$, $\vec{\beta} \in (\mathbf{B}^{(1)})^k$ $\vec{p} \in \mathbf{P}^\ell$, and $\vec{\pi} \in (\mathbf{P}^{(1)})^m$, let

$$\widehat{\mathrm{op}}(\vec{v} \cdot \vec{b} \cdot \vec{\beta} \cdot \vec{p} \cdot \vec{\pi}) = \mathcal{H}(\widetilde{\mathrm{op}}(\vec{v} \cdot \vec{b} \cdot \vec{\beta} \cdot \vec{p} \cdot \vec{\pi})). \blacksquare$$

5.4.3 Definition of the Denotational Model $\mathcal{D}$

Having defined the semantic operations $\widehat{\mathrm{op}}$, the definition of the denotational model $\mathcal{D}$ is straightforward. In order to formulate the definition, it is convenient to introduce the set **Valt** of *valuations*.

Definition 21 Let $(\zeta \in)$ **Valt** be the set of those elements of $(\mathbf{Var} \to (\mathbf{V} \cup \mathbf{P}^*))$ which preserve types in the following sense:

$$(Z \in \mathbf{Var}_\mathcal{V} \Rightarrow \zeta(Z) \in \mathbf{V}) \wedge (Z \in \mathbf{Var}_\mathcal{P} \Rightarrow \zeta(Z) \in \mathbf{P}) \wedge (Z \in \mathbf{Var}_\mathcal{P}^{(1)} \Rightarrow \zeta(Z) \in \mathbf{P}^{(1)}).$$

Elements of **Valt** are called *valuations*.

We fix a valuation $\bar{\zeta} \in$ **Valt** in the rest of this paper for notational convenience in defining denotational models. ∎

We define the model $\mathcal{D}$ as the least fixed-point model based on the domain $\mathbf{P}$ and the semantic operations $\widehat{\mathrm{op}}$. That is, we define $\mathcal{D}$ as the unique function $\mathcal{D} \in \mathcal{L}^* \to ([\mathbf{Valt} \to \mathbf{P}] \cup [\mathbf{Valt} \to \mathbf{P}^{(1)}])$ satisfying conditions (17)–(22) listed in the next definition.

Definition 22 The mapping $\mathcal{D} : \mathcal{L}^* \to ([\mathbf{Valt} \to \mathbf{P}] \cup [\mathbf{Valt} \to \mathbf{P}^{(1)}])$ preserves types in the sense that

$$\forall S \in \mathcal{L}[\, \mathcal{D}[\![S]\!] \in [\mathbf{Valt} \to \mathbf{P}]\,] \wedge \forall T \in \mathcal{L}^{(1)}[\, \mathcal{D}[\![S]\!] \in [\mathbf{Valt} \to \mathbf{P}^{(1)}]\,], \tag{17}$$

and satisfies the following conditions:

$$\textbf{(i)}\ \forall Z \in \mathbf{Var}, \forall \zeta \in \mathbf{Valt}[\, \mathcal{D}[\![Z]\!](\zeta) = \zeta(Z)\,]. \tag{18}$$

$$\textbf{(ii)}\ \forall (i,j,k,\ell,m) \in \omega^5, \forall \mathrm{op} \in \mathbf{Fun}_{(i,j,k,\ell,m)}, \tag{19}$$
$$\forall \vec{E} \in \mathcal{E}^i, \forall \vec{G} \in \mathcal{G}^j, \forall \vec{H} \in (\mathcal{G}^{(1)})^k, \forall \vec{S} \in \mathcal{L}^\ell, \forall \vec{T} \in (\mathcal{L}^{(1)})^m, \forall \zeta \in \mathbf{Valt}[$$
$$\mathcal{D}[\![\mathrm{op}(\vec{E} \cdot \vec{G} \cdot \vec{H} \cdot \vec{S} \cdot \vec{T})]\!](\zeta) = \widehat{\mathrm{op}}([\![\vec{E}]\!] \cdot [\![\vec{G}]\!] \cdot [\![\vec{H}]\!] \cdot \mathcal{D}[\![\vec{S}]\!](\zeta) \cdot \mathcal{D}[\![\vec{T}]\!](\zeta))\,].$$

$$\textbf{(iii)}\ \forall x \in \mathbf{Var}_\mathcal{V}, \forall S \in \mathcal{L}, \forall \zeta \in \mathbf{Valt}[\, \mathcal{D}[\![(\lambda x.\ S)]\!](\zeta) = (\lambda v \in \mathbf{V}.\ \mathcal{D}[\![S]\!](\zeta[v/x]))\,]. \tag{20}$$

$$\textbf{(iv)}\ \forall X \in \mathbf{Var}_\mathcal{P}, \forall S \in \mathcal{L}[\, (\mu X.\ S) \in \mathcal{L} \Rightarrow \tag{21}$$
$$\forall \zeta \in \mathbf{Valt}[\, (\lambda p \in \mathbf{P}.\ \mathcal{D}[\![S]\!](\zeta[p/X])) \in [\mathbf{P} \to \mathbf{P}]$$
$$\wedge \mathcal{D}[\![(\mu X.\ S)]\!](\zeta) = \mathbf{Y}(\lambda p \in \mathbf{P}.\ \mathcal{D}[\![S]\!](\zeta[p/X]))\,]].$$

$$\textbf{(v)}\ \forall \xi \in \mathbf{Var}_\mathcal{P}^{(1)}, \forall T \in \mathcal{L}^{(1)}[\, (\mu \xi.\ T) \in \mathcal{L}^{(1)} \Rightarrow \tag{22}$$
$$\forall \zeta \in \mathbf{Valt}[\, (\lambda \pi \in \mathbf{P}^{(1)}.\ \mathcal{D}[\![T]\!](\zeta[\pi/\xi])) \in [\mathbf{P}^{(1)} \to \mathbf{P}^{(1)}]$$
$$\wedge \mathcal{D}[\![(\mu \xi.\ T)]\!](\zeta) = \mathbf{Y}(\lambda \pi \in \mathbf{P}^{(1)}.\ \mathcal{D}[\![T]\!](\zeta[\pi/\xi]))\,]]. \blacksquare$$

For every closed statement $s \in \mathcal{L}[\emptyset]$, the value $\mathcal{D}[\![s]\!](\zeta)$ does not depend on ζ.

Proposition 9 $\forall s \in \mathcal{L}[\emptyset], \forall \zeta_0, \zeta_1 \in \mathbf{Valt}[\, \mathcal{D}[\![s]\!](\zeta_0) = \mathcal{D}[\![s]\!](\zeta_1)\,]$. ∎

Notation 2 In the rest of this paper, we simply write $\mathcal{D}[\![s]\!]$ for $s \in \mathcal{L}[\emptyset]$ to denote the value $\mathcal{D}[\![s]\!](\bar{\zeta}) \in \mathbf{P}$; thus the notation $\mathcal{D}[\![s]\!]$ denote either an element of $(\mathbf{Valt} \to \mathbf{P})$ or an element of $\mathbf{P}$ depending on the context. We will write $\mathcal{D}[\![s]\!](\cdot)$ to explicitly denote an element of $(\mathbf{Valt} \to \mathbf{P})$ when it is necessary. ∎

The model $\mathcal{D}$ is *compositional* in the sense that for any two statements s_0 and s_0 with the same meaning in $\mathcal{D}$, s_0 in an arbitrary context can be replaced by s_1 without changing the overall meaning. Namely, we have the following proposition.

Proposition 10

$$\forall X \in \mathbf{Var}_\mathcal{P}, \forall X \in \mathcal{L}[X], \forall s_0, s_1 \in \mathcal{L}[\emptyset][\, \mathcal{D}[\![s_0]\!] = \mathcal{D}[\![s_1]\!] \Rightarrow \mathcal{D}[\![S[s_0/X]]\!] = \mathcal{D}[\![S[s_1/X]]\!]\,]. \tag{23}$$

Proof. We can prove that the following holds for every $S \in \mathcal{L}[X]$ by induction on the structure of S:

$$\forall \zeta \in \mathbf{Valt}, \forall s \in \mathcal{L}[\emptyset][\, \mathcal{D}[\![S[s/X]]\!] = \mathcal{D}[\![S]\!](\zeta[\mathcal{D}[\![s]\!]/X])\,]. \tag{24}$$

From this the claim (23) immediately follows. ■

5.5 Auxiliary Denotational Models $\mathcal{D}_\tau$ and $\mathcal{M}_\tau$

In order to relate the denotational model $\mathcal{D}$ and the operational model $\mathcal{O}$, we introduce two auxiliary denotational models $\mathcal{D}_\tau$ and $\mathcal{M}_\tau$. The models $\mathcal{D}_\tau$ and $\mathcal{M}_\tau$ are *strong* denotational models based on the cpo $(\mathbf{P}, \sqsubseteq_s)$ and the cms $(\mathbf{P}, \tilde{d})$, respectively.

5.5.1 The Strong Order-Theoretic Model $\mathcal{D}_\tau$

The strong order-theoretic model $\mathcal{D}_\tau$ is defined just as $\mathcal{D}$ on the basis of the cpo $(\mathbf{P}, \sqsubseteq_s)$, but in terms of the original semantic operations $\widetilde{\mathrm{op}}$ instead of the weak versions. That is, $\mathcal{D}_\tau$ is defined as the unique function $\mathcal{D}_\tau \in \mathcal{L}^* \to ([\mathbf{Valt} \to \mathbf{P}] \cup [\mathbf{Valt} \to \mathbf{P}^{(1)}])$ satisfying conditions (17)—(22) with $\mathcal{D}$ and $\widetilde{\mathrm{op}}$ replaced by $\mathcal{D}_\tau$ $\widetilde{\mathrm{op}}$, respectively.

The model $\mathcal{D}_\tau$ has the following property corresponding to Proposition 9 for $\mathcal{D}$.

Proposition 11 $\forall s \in \mathcal{L}[\emptyset], \forall \zeta_0, \zeta_1 \in \mathbf{Valt}[\ \mathcal{D}_\tau[\![s]\!](\zeta_0) = \mathcal{D}_\tau[\![s]\!](\zeta_1)\ ].$ ∎

We will use the notational convention in Notation 2 for $\mathcal{D}_\tau$. as well as for $\mathcal{D}$.

5.5.2 The Metric Model $\mathcal{M}_\tau$

We define the metric model $\mathcal{M}_\tau$ on the basis of the cms $(\mathbf{P}, \tilde{d})$ and in terms of the semantic operations $\widetilde{\mathrm{op}}$ ($\mathrm{op} \in \mathbf{Fun}$). More precisely, $\mathcal{M}_\tau$ is defined as the unique function

$$\mathcal{M}_\tau : \mathcal{L}^* \to ((\mathbf{Valt} \to^1 \mathbf{P}) \cup (\mathbf{Valt} \to^1 \mathbf{P}^{(1)}))$$

satisfying conditions (25)–(30) listed in the next definition.

Definition 23 The mapping $\mathcal{M}_\tau : \mathcal{L}^* \to ((\mathbf{Valt} \to^1 \mathbf{P}) \cup (\mathbf{Valt} \to^1 \mathbf{P}^{(1)}))$ preserves types in the sense that

$$\forall S \in \mathcal{L}[\ M[\![S]\!] \in (\mathbf{Valt} \to^1 \mathbf{P})\ ] \wedge \forall T \in \mathcal{L}^{(1)}[\ M[\![S]\!] \in (\mathbf{Valt} \to^{(1)} \mathbf{P}^{(1)})\ ], \tag{25}$$

and satisfies the following conditions:

(i) $\forall Z \in \mathbf{Var}, \forall \zeta \in \mathbf{Valt}[\ \mathcal{M}_\tau[\![Z]\!](\zeta) = \zeta(Z)\ ].$ (26)

(ii) $\forall (i,j,k,\ell,m) \in \omega^5, \forall \mathrm{op} \in \mathbf{Fun}_{(i,j,k,\ell,m)},$ (27)
$\forall \vec{E} \in \mathcal{E}^i, \forall \vec{G} \in \mathcal{G}^j, \forall \vec{H} \in (\mathcal{G}^{(1)})^k, \forall \vec{S} \in \mathcal{L}^\ell, \forall \vec{T} \in (\mathcal{L}^{(1)})^m, \forall \zeta \in \mathbf{Valt}[$
$\mathcal{M}_\tau[\![\mathrm{op}(\vec{E} \cdot \vec{G} \cdot \vec{H} \cdot \vec{S} \cdot \vec{T})]\!](\zeta) = \widetilde{\mathrm{op}}([\![\vec{E}]\!] \cdot [\![\vec{G}]\!] \cdot [\![\vec{H}]\!] \cdot \mathcal{M}_\tau[\![\vec{S}]\!](\zeta) \cdot \mathcal{M}_\tau[\![T]\!](\zeta))\ ].$

(iii) $\forall x \in \mathbf{Var}_\mathcal{V}, \forall S \in \mathcal{L}, \forall \zeta \in \mathbf{Valt}[$ (28)
$\mathcal{M}_\tau[\![(\lambda x.\ S)]\!](\zeta) = (\lambda v \in \mathbf{V}.\ \mathcal{M}_\tau[\![S]\!](\zeta[v/x]))\ ].$

(iv) $\forall X \in \mathbf{Var}_\mathcal{P}, \forall S \in \mathcal{L}[\ (\mu X.\ S) \in \mathcal{L} \Rightarrow$ (29)
$\forall \zeta \in \mathbf{Valt}[\ (\lambda p \in \mathbf{P}.\ \mathcal{M}_\tau[\![S]\!](\zeta[p/X])) \in (\mathbf{P} \to^{1/2} \mathbf{P})$
$\wedge \mathcal{M}_\tau[\![(\mu X.\ S)]\!](\zeta) = \mathrm{fix}(\lambda p \in \mathbf{P}.\ \mathcal{M}_\tau[\![S]\!](\zeta[p/X]))\ ]].$

(v) $\forall \xi \in \mathbf{Var}^{(1)}_\mathcal{P}, \forall T \in \mathcal{L}^{(1)}[\ (\mu \xi.\ T) \in \mathcal{L}^{(1)} \Rightarrow$ (30)
$\forall \zeta \in \mathbf{Valt}[\ (\lambda \pi \in \mathbf{P}^{(1)}.\ \mathcal{M}_\tau[\![T]\!](\zeta[\pi/\xi])) \in (\mathbf{P}^{(1)} \to^{1/2} \mathbf{P}^{(1)})$
$\wedge \mathcal{M}_\tau[\![(\mu \xi.\ T)]\!](\zeta) = \mathrm{fix}(\lambda \pi \in \mathbf{P}^{(1)}.\ \mathcal{M}_\tau[\![T]\!](\zeta[\pi/\xi]))\ ]].$ ∎

The model $\mathcal{M}_\tau$ has the following property corresponding to Proposition 9 for $\mathcal{D}$.

Proposition 12 $\forall s \in \mathcal{L}[\emptyset], \forall \zeta_0, \zeta_1 \in \mathbf{Valt}[\ \mathcal{M}_\tau[\![s]\!](\zeta_0) = \mathcal{M}_\tau[\![s]\!](\zeta_1)\ ].$ ∎

In the rest of this paper, we simply write $\mathcal{M}_\tau[\![s]\!]$ for $s \in \mathcal{L}[\emptyset]$ to denote the value $\mathcal{M}_\tau[\![s]\!](\tilde{\zeta}) \in \mathbf{P}$; We will use the notational convention in Notation 2 for $\mathcal{M}_\tau$. as well as for $\mathcal{D}$ and $\mathcal{D}_\tau$.

6 Full Abstractness of $\mathcal{D}$ with respect to $\mathcal{O}$

In this section, we investigate the relationship between the denotational model $\mathcal{D}$ and the operational model $\mathcal{O}$, and thereby establish the full abstractness of $\mathcal{D}$ with respect to $\mathcal{O}$. For relating the two models $\mathcal{D}$ and $\mathcal{O}$, we introduce two auxiliary models $\mathcal{C}_\tau$ and $\mathcal{C}$; these models are defined operationally and called the *intermediate models*.

6.1 Intermediate Models $\mathcal{C}_\tau$ and $\mathcal{C}$

In this subsection, we define two intermediate models $\mathcal{C}_\tau$ and $\mathcal{C}$; the former is strong and the latter is weak. First, we define the strong intermediate model $\mathcal{C}_\tau$ by:

Definition 24 (Strong Operational Model $\mathcal{C}_\tau$) We define $\mathcal{C}_\tau : \mathcal{L}[\emptyset] \to \wp(\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp)$ as follows: For $s \in \mathcal{L}[\emptyset]$ and $q \in (\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp)$, we put $q \in \mathcal{C}_\tau[\![s]\!]$, if either of the following two conditions (i) and (ii) is satisfied:
(i) $\exists w \in (\mathbf{C}_\tau)^{<\omega}, \exists C \in \wp(\mathbf{C})[\ q = w \cdot \langle C\rangle \wedge \exists s' \in \mathcal{L}[\emptyset][\ s \stackrel{w}{\Longrightarrow} s' \wedge \mathrm{act}(s') \cap (C \cup \{\tau\}) = \emptyset\ ]]$.
(ii) $q \in (\mathbf{C}_\tau)^\omega \wedge q \stackrel{w}{\longrightarrow} .$ ∎

For convenience in relating $\mathcal{C}_\tau$ with $\mathcal{M}_\tau$ in Sect. 6.3, we give an alternative characterization of $\mathcal{C}_\tau$ as the unique fixed-point of a higher-order contraction Ψ defined by:

Definition 25 Let $\mathbf{M} = (\mathcal{L}[\emptyset] \to \mathbf{P})$. Clearly $\mathbf{M}$ is a cms with the pointwise metric induced by $\tilde{d}$. The mapping $\Psi : \mathbf{M} \to \mathbf{M}$ is defined as follows: For every $F \in \mathbf{M}$, $\Psi(F)$ is the mapping defined by

$$\begin{aligned}\Psi(F)(s) = {} & \{\langle C\rangle |\ C \in \wp(\mathbf{C}) \wedge (C \cup \{\tau\}) \cap \mathrm{act}(s) = \emptyset\} \\ & \cup \bigcup\{\langle a\rangle \cdot F(s') |\ a \in \mathbf{C}_\tau \wedge s' \in \mathcal{L}[\emptyset] \wedge s \stackrel{a}{\longrightarrow} s'\},\end{aligned} \tag{31}$$

where s ranges over $\mathcal{L}[\emptyset]$. ∎

It is easy to check that Ψ is a contraction from $\mathbf{M}$ to $\mathbf{M}$. From the definition of $\mathcal{C}_\tau$, it follows that $\mathcal{C}_\tau$ is a fixed-point of Ψ, i.e., that $\forall u \in \mathcal{L}^*[\emptyset][\ \Psi(\mathcal{C}_\tau)(u) = \mathcal{C}_\tau[\![u]\!]\ ]$. Thus we have:

Proposition 13 (Alternative Characterization of $\mathcal{C}_\tau$) *One has $\mathcal{C}_\tau = \mathrm{fix}(\Psi)$. That is, for every $s \in \mathcal{L}[\emptyset]$,*

$$\begin{aligned}\mathcal{C}_\tau[\![s]\!] = {} & \{\langle C\rangle |\ C \in \wp(\mathbf{C}) \wedge (C \cup \{\tau\}) \cap \mathrm{act}(s) = \emptyset\} \\ & \cup \bigcup\{\langle a\rangle \cdot \mathcal{C}_\tau[\![s']\!] |\ a \in \mathbf{C}_\tau \wedge s' \in \mathcal{L}[\emptyset] \wedge s \stackrel{a}{\longrightarrow} s'\}.\end{aligned} \tag{32}$$

∎

From $\mathcal{C}_\tau$, the weak intermediate model $\mathcal{C}$ is defined by applying the hiding function $\mathcal{H}$:

Definition 26 We define $\mathcal{C} : \mathcal{L}[\emptyset] \to \mathbf{P}$ as follows: For $s \in \mathcal{L}[\emptyset]$, let $\mathcal{C}[\![s]\!] = \mathcal{H}(\mathcal{C}_\tau[\![s]\!])$. ∎

6.2 Completeness of $\mathcal{C}$ with respect to $\mathcal{O}$

In this subsection, we investigate the relationship between the intermediate model $\mathcal{C}$ and the operational model $\mathcal{O}$, and thereby establish a relationship called the *completeness* of $\mathcal{C}$ with respect to $\mathcal{O}$.

First, we need a few preliminary definitions.

Definition 27 For $p \in \wp(\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp)$, let $\tilde{\mathcal{T}}_\perp(p) = p \cap ((\mathbf{C}_\tau)^{<\omega} \cdot \{\langle\perp\rangle\})$, $\tilde{\mathcal{T}}_\omega(p) = p \cap (\mathbf{C}_\tau)^\omega$, and $\tilde{\mathcal{F}}(p) = p \cap ((\mathbf{C}_\tau)^{<\omega} \cdot \wp(\mathbf{C}))$. ∎

Definition 28 We define an abstraction function $\mathcal{A} : \wp(\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp) \to \wp(\mathbf{C}_\tau \triangleright \{\delta, \perp\})$ as follows: For $p \in \wp(\mathbf{C}_\tau \triangleright \wp(\mathbf{C})_\perp)$, let $\mathcal{A}(p) = \tilde{\mathcal{T}}_\perp(p) \cup \tilde{\mathcal{T}}_\omega(p) \cup \{w \cdot \langle\delta\rangle |\ w \cdot \langle\mathbf{C}\rangle \in p\}$. ∎

We immediately obtain the following connection between $\mathcal{O}$ and $\mathcal{C}$, from the definitions of $\mathcal{O}$, $\mathcal{C}$, and $\mathcal{A}$:

Proposition 14 $\forall s \in \mathcal{L}[\emptyset][\ \mathcal{O}[\![s]\!] = \mathcal{A}(\mathcal{C}[\![s]\!])\ ]$. ∎

Moreover, the model $\mathcal{C}$ is *complete* with respect to $\mathcal{O}$ in the following sense:

Lemma 3 *Let $\mathbf{V}$ is infinite. Then for every $s_0, s_1 \in \mathcal{L}[\emptyset]$, one has*

$$\mathcal{C}[\![s_0]\!] \neq \mathcal{C}[\![s_1]\!] \Rightarrow \exists \hat{s} \in \mathcal{L}[\emptyset][\ \mathcal{O}[\![s_0 \| \hat{s}]\!] \neq \mathcal{O}[\![s_1 \| \hat{s}]\!]\ ]. \text{ ∎} \tag{33}$$

Proof. Suppose $\mathbf{V}$ is infinite. And let s_0 and s_1 be programs such that $\mathcal{C}[\![s_0]\!] \neq \mathcal{C}[\![s_1]\!]$. We fix $X \in \mathbf{Var}_\mathcal{D}$. It suffices to show that

$$\exists S \in \mathcal{L}[X][\ \mathcal{O}[\![S[s_0/X]]\!] \neq \mathcal{O}[\![S[s_0/X]]\!]\ ]. \tag{34}$$

Since $\mathcal{C}[\![s_0]\!] \neq \mathcal{C}[\![s_1]\!]$, one of the three conditions (35)(i), (35)(ii), or (35)(iii) holds:

$$\text{(i) } \tilde{\mathcal{T}}_\perp(\mathcal{C}[\![s_0]\!]) \neq \tilde{\mathcal{T}}_\perp(\mathcal{C}[\![s_1]\!]),\ \text{(ii) } \tilde{\mathcal{T}}_\omega(\mathcal{C}[\![s_0]\!]) \neq \tilde{\mathcal{T}}_\omega(\mathcal{C}[\![s_1]\!]),\ \text{(iii) } \tilde{\mathcal{F}}(\mathcal{C}[\![s_0]\!]) \neq \tilde{\mathcal{F}}(\mathcal{C}[\![s_1]\!]). \tag{35}$$

When (35)(i) or (35)(ii) holds, we immediately obtains (34) by putting $S \equiv X$. Let us consider the remaining case that (35)(iii) holds.

We can assume without loss of generality that there exist $w \in \mathbf{C}^{<\omega}$ and $C \in \wp(\mathbf{C})$ such that

$$\text{(i) } w \cdot \langle C \rangle \in \mathcal{C}[\![s_0]\!],\ \text{(ii) } w \cdot \langle C \rangle \notin \mathcal{C}[\![s_1]\!]. \tag{36}$$

Let us put $w = \langle c_0, \ldots, c_k \rangle$.

If there exists $w' \preceq w$ such that $w' \cdot \langle \perp \rangle \in \tilde{T}(\mathcal{C}[\![s_1]\!])$, then we have $w' \cdot \langle \perp \rangle \notin \tilde{\mathcal{T}}_\perp(\mathcal{C}[\![s_0]\!]) \wedge w' \cdot \langle \perp \rangle \in \tilde{\mathcal{T}}_\perp(\mathcal{C}[\![s_0]\!])$, and therefore this case is reduced to the case that (35)(i) holds. Thus it suffices to consider the case that

$$\neg \exists w' \preceq w [\ w' \cdot \langle \perp \rangle \in \tilde{\mathcal{T}}_\perp(\mathcal{C}[\![s_1]\!])\ ]. \tag{37}$$

Then, we can construct a set $\tilde{C} \in \wp(\mathbf{C})$ and a program $\widehat{s}$ such that

$$\text{(i) } \langle \delta \rangle \in \mathcal{O}[\![\partial_{\tilde{C}}(s_0 \| \widehat{s})]\!],\ \text{(ii) } \langle \delta \rangle \in \mathcal{O}[\![\partial_{\tilde{C}}(s_1 \| \widehat{s})]\!] \tag{38}$$

as follows. First, we put $\tilde{C} = \bigcup_{w' \preceq w} (\mathcal{A}(s_0, w') \cup \mathcal{A}(s_1, w')) \setminus C$, where we define $\mathcal{S}(\tilde{s}, \tilde{w})$, for $\tilde{s} \in \mathcal{L}[\emptyset]$ and $\tilde{w} \in \mathbf{C}^{<\omega}$, by

$$\mathcal{S}(\tilde{s}, \tilde{w}) = \{\mathrm{chan}(c) |\ c \in \mathbf{C} \wedge \exists s' [\ \tilde{s} \stackrel{\tilde{w}}{\Longrightarrow} s' \wedge c \in \mathrm{act}(s')\ ]\}.$$

We fix an element $\bar{v} \in \mathbf{V}$ arbitrarily, and let $\bar{v}_1$ be an element of $\mathbf{V}$ such that

$$\bar{v}_1! \in (\mathbf{V}! \setminus \bigcup_{w' \preceq w} \bigcup_{i \in 2} (\mathcal{S}(s_i, w') \cup \overline{\mathcal{S}(s_i, w')})).$$

(The nonemptiness of the right-hand set of the above formula follows from (36)(i) and (37); see Proposition 22 of [21] for the proof.) We inductively define $\widehat{s}_0, \ldots, \widehat{s}_k$ by the following clauses:

$$\text{(i) } \widehat{s}_0 \equiv \mathbf{0};\ \text{(ii) for } i \in \{1, \ldots, k\}, \text{ let } \widehat{s}_i \equiv \mathbf{out}(\bar{v}_1, \bar{v}, \mathbf{0}) + f(\overline{c_{k-i}}, \widehat{s}_{i-1}),$$

where we define $f(c, \tilde{s})$, for $c \in \mathbf{C}$ and $\tilde{s} \in \mathcal{L}[\emptyset]$, by

$$f(c, \tilde{s}) = \begin{cases} \mathbf{out}(v, v', \tilde{s}) & \text{if } c = (v!, v'), \\ \mathbf{in}(v, (\lambda x.\ x = v'), (\lambda x.\ \tilde{s})) & \text{if } c = (v?, v'). \end{cases}$$

And we put $\widehat{s} \equiv \widehat{s}_k$. Then, we can prove (38) by an operational analysis of the behaviors of the two programs $\partial_{\tilde{C}}(s_0 \| \widehat{s})$ and $\partial_{\tilde{C}}(s_1 \| \widehat{s})$. See Lemma 7 of [21] for details. ■

6.3 Equivalence of $\mathcal{D}_\tau$, $\mathcal{M}_\tau$, and $\mathcal{C}_\tau$

For $\vec{Z} \in (\mathbf{Var}^*_{\mathcal{P}})^{<\omega}$ and $\vec{U} \in (\mathcal{L}^*)^{<\omega}$, we say $\vec{Z}$ *and* $\vec{U}$ *agree in types* if $\mathrm{lgt}(\vec{Z}) = \mathrm{lgt}(\vec{U})$ and

$$\forall i \in \mathrm{lgt}(\vec{Z})[\ (\vec{Z}(i) \in \mathbf{Var}_{\mathcal{P}} \Rightarrow \vec{u}(i) \in \mathcal{L}) \wedge (\vec{Z}(i) \in \mathbf{Var}^{(1)}_{\mathcal{P}} \Rightarrow \vec{u}(i) \in \mathcal{L}^{(1)})\ ].$$

In order to relate $\mathcal{C}_\tau$ and $\mathcal{M}_\tau$, we extend the domain of $\mathcal{C}_\tau$ from $\mathcal{L}[\emptyset]$ to $\mathcal{L}^*[\emptyset]$ as follows.

Definition 29 For $t \in \mathcal{L}^{(1)}[\emptyset]$, we define $\mathcal{C}_\tau[\![t]\!]$ by $\mathcal{C}_\tau[\![t]\!] = (\lambda v \in \mathbf{V}.\ \mathcal{C}_\tau[\![\mathbf{ap}(t, v)]\!])$. ▮

The model $\mathcal{C}$ is a *homomorphism* from $\mathcal{L}^*[\emptyset]$ to $\mathbf{P}^*$ in the following sense:

Lemma 4 *For* $(i, j, k, \ell, m) \in \omega^5$ *and* $\mathrm{op} \in \mathbf{Fun}_{(i,j,k,\ell,m)}$, *one has*

$$\begin{aligned} &\forall \vec{e} \in (\mathcal{E}[\emptyset])^i, \forall \vec{g} \in (\mathcal{G}[\emptyset])^j, \forall \vec{h} \in (\mathcal{G}^{(1)}[\emptyset])^k, \forall \vec{s} \in (\mathcal{L}[\emptyset])^\ell, \forall \vec{t} \in (\mathcal{L}^{(1)}[\emptyset])^m[\\ &\quad \mathcal{C}_\tau[\![\mathrm{op}(\vec{e} \cdot \vec{g} \cdot \vec{h} \cdot \vec{s} \cdot \vec{t})]\!] = \widetilde{\mathrm{op}}([\![\vec{e} \cdot \vec{g} \cdot \vec{h}]\!] \cdot \mathcal{C}_\tau[\![\vec{s} \cdot \vec{t}]\!])\ ]. \blacksquare \end{aligned} \tag{39}$$

This lemma follows from the definition of the operations $\widetilde{\mathrm{op}}$ by applying the general method described in [17, 19]. See Lemma 8 of [21] for the proof.

The two models $\mathcal{C}_\tau$ and $\mathcal{M}_\tau$ are equivalent in the following sense:

Lemma 5 *For every* $U \in \mathcal{L}^*$ *with* $\mathrm{FV}(U) \subseteq \mathbf{Var}^*_{\mathcal{P}}$, *one has*

$$\forall \vec{Z} \in (\mathbf{Var}^*_{\mathcal{P}})^{<\omega}, \forall \vec{u} \in (\mathcal{L}^*[\emptyset])^{<\omega}[\ \mathrm{FV}(U) \subseteq \mathrm{ran}(\vec{Z}) \wedge \vec{Z} \textit{ and } \vec{u} \textit{ agree in types} \Rightarrow \forall \zeta \in \mathbf{Valt}[\ \mathcal{C}_\tau[\![U[\vec{u}/\vec{Z}]]\!] = \mathcal{M}_\tau[\![U]\!](\zeta[\mathcal{C}_\tau[\![\vec{u}]\!]/\vec{Z}])\ ]]. \blacksquare \tag{40}$$

Proof. The claim (40) can be proved by induction on the structure of $U \in \mathcal{L}^*$, using Lemma 4 and properties (25)–(30) of $\mathcal{M}_\tau$. See Lemma 9 of [21] for the proof. ■

From Lemma 5, we immediately obtain the next corollary.

Corollary 2 $\forall s \in \mathcal{L}[\emptyset][\ \mathcal{C}_\tau[\![s]\!] = \mathcal{M}_\tau[\![s]\!]\ ]. \blacksquare$

Although the respective underlying structures of $\mathcal{M}_\tau$ and $\mathcal{D}_\tau$ are different, the meaning of each program under $\mathcal{M}_\tau$ is equal to its meaning under $\mathcal{D}_\tau$.

Lemma 6 $\forall U \in \mathcal{L}^*, \forall \zeta \in \mathbf{Valt}[\ \mathcal{M}_\tau[\![U]\!](\zeta) = \mathcal{D}_\tau[\![U]\!](\zeta)\ ]. \blacksquare$

Proof. This can be proved by induction on the structure of $U \in \mathcal{L}^*$, using properties (25)–(30) of $\mathcal{M}_\tau$ and the corresponding properties of $\mathcal{D}_\tau$. ■

From Lemma 6, we immediately obtain the following corollary.

Corollary 3 $\forall s \in \mathcal{L}[\emptyset][\ \mathcal{M}_\tau[\![s]\!] = \mathcal{D}_\tau[\![s]\!]\ ]. \blacksquare$

From Corollary 4, Lemma 6, and Corollary 3, we obtain the following proposition.

Proposition 15 $\forall s \in \mathcal{L}[\emptyset][\ \mathcal{C}[\![s]\!] = \mathcal{D}[\![s]\!]\ ]. \blacksquare$

6.4 Relationship between $\mathcal{D}$ and $\mathcal{D}_\tau$

The denotational model $\mathcal{D}$ can be represented as the composition of $\mathcal{D}_\tau$ and the hiding function $\mathcal{H}$; this is the claim of Lemma 7 below. Since the two models $\mathcal{D}$ and $\mathcal{D}_\tau$ are defined independently, this relation does not follow immediately, and we need a few preliminaries.

6.4.1 Hiding as a Homomorphism

We extend the domain of $\mathcal{H}$ to $(\mathbf{V} \cup \mathbf{B}^{(*)} \cup \mathbf{P}^{(*)})$ as follows:

Definition 30 For $z \in (\mathbf{V} \cup \mathbf{B}^* \cup \mathbf{P}^{(*)})$, let

$$\mathcal{H}(z) = \begin{cases} z & \text{if } z \in \mathbf{V} \cup \mathbf{B}^{(*)}, \\ \mathcal{H}(z) & \text{if } z \in \mathbf{P}, \\ \mathcal{H} \circ z = (\lambda v \in \mathbf{V}.\ \mathcal{H}(z(v))) & \text{if } z \in \mathbf{P}^{(1)}. \end{cases}$$

For $n \in \omega$ and $\vec{z} \in (\mathbf{V} \cup \mathbf{B}^{(*)} \cup \mathbf{P}^{(*)})^n$, we write $\mathcal{H}(\vec{z})$ to denote $\langle \mathcal{H}(\vec{z}(i)) \rangle_{i \in n}$. ■

The hiding function $\mathcal{H}$ is a homomorphism in the following sense:

Lemma 7 *For* $(i,j,k,\ell,m) \in \omega^5$ *and* $\mathrm{op} \in \mathbf{Fun}_{(i,j,k,\ell,m)}$, *one has*

$$\forall \vec{v} \in \mathbf{V}^i, \forall \vec{b} \in \mathbf{B}^j, \forall \vec{\beta} \in (\mathbf{B}^{(1)})^k, \forall \vec{p} \in \mathbf{P}^\ell, \forall \vec{\pi} \in (\mathbf{P}^{(1)})^m[\mathcal{H}(\widetilde{\mathrm{op}}(\vec{v} \cdot \vec{b} \cdot \vec{\beta} \cdot \vec{p} \cdot \vec{\pi})) = \widehat{\mathrm{op}}(\mathcal{H}(\vec{v} \cdot \vec{b} \cdot \vec{\beta} \cdot \vec{p} \cdot \vec{\pi}))\ ]. \blacksquare \tag{41}$$

Proof. It is easy to check that (41) holds, except for the recursively defined operations $\tilde{\|}$ and $\tilde{\partial}_C$. For these operations, we first prove (41) for finite processes by induction on their length, and thereby prove (41) for infinite processes (see Propositions 6.16 and 6.19 of [18, Chapter 6]). ■

6.4.2 Relationship between $\mathcal{D}_\tau$ and $\mathcal{D}$

The two models $\mathcal{D}_\tau$ and $\mathcal{D}$ are related by means of the hiding function $\mathcal{H}$ as follows.

Lemma 8 *For every* $U \in \mathcal{L}^*$, *one has* $\forall \zeta \in \mathbf{Valt}[\ \mathcal{H}(\mathcal{D}_\tau[\![U]\!](\zeta)) = \mathcal{D}[\![U]\!](\mathcal{H} \circ \zeta)\ ]. \blacksquare$

Proof. This lemma can be established by induction on the structure of $U \in \mathcal{L}^*$, using the continuity of $\mathcal{H}$ and the fact that $\mathcal{H}$ is a homomorphism (Lemma 7). See Lemma 6 of [21] for details. ■

The next corollary immediately follows from Lemma 8:

Corollary 4 $\forall s \in \mathcal{L}[\ \mathcal{D}[\![s]\!] = \mathcal{H}(\mathcal{D}_\tau[\![s]\!])\ ]. \blacksquare$

6.5 Full Abstractness of $\mathcal{D}$ with respect to $\mathcal{O}$

By combining the result established in the previous section, we can now prove the main result of our paper, the full abstractness of $\mathcal{D}$ with respect to $\mathcal{O}$.

Theorem 1 *Let* $\mathbf{V}$ *is infinite. Then for every* $s_0, s_1 \in \mathcal{L}[\emptyset]$, *one has*

$$\mathcal{D}[\![s_0]\!] = \mathcal{D}[\![s_1]\!] \Leftrightarrow \forall X \in \mathbf{Var}_{\mathcal{P}}, \forall S \in \mathcal{L}[X][\ \mathcal{O}[\![S[s_0/X]]\!] = \mathcal{O}[\![S[s_1/X]]\!]\ ]. \blacksquare \tag{42}$$

Proof. Let $s_0, s_1 \in \mathcal{L}[\emptyset]$. We will prove (42).

($\Rightarrow$) Suppose that (*): $\mathcal{D}[\![s_0]\!] = \mathcal{D}[\![s_0]\!]$. Let $X \in \mathbf{Var}_{\mathcal{P}}$ and $S \in \mathcal{L}[X]$. We have

$\mathcal{O}[\![S[s_0/S]]\!] = \mathcal{A}(\mathcal{C}[\![S[s_0/S]]\!])$ (by Proposition 14) $= \mathcal{A}(\mathcal{D}[\![S[s_0/S]]\!])$ (by Proposition 15)
$= \mathcal{A}(\mathcal{D}[\![S[s_1/S]]\!])$ (by (*) and Proposition 10) $= \mathcal{A}(\mathcal{C}[\![S[s_1/S]]\!]) = \mathcal{O}[\![S[s_1/S]]\!]$.

Thus we obtain the $\Rightarrow$-part of (42).

($\Leftarrow$) To prove the $\Leftarrow$-part of (42), it suffices to prove its contrapositive:

$$\mathcal{D}[\![s_0]\!] \neq \mathcal{D}[\![s_1]\!] \Rightarrow \exists X \in \mathbf{Var}_{\mathcal{P}}, \exists S \in \mathcal{L}[X][\ \mathcal{O}[\![S[s_0/X]]\!] \neq \mathcal{O}[\![S[s_1/X]]\!]\ ]. \tag{43}$$

Suppose that $\mathcal{D}[\![s_0]\!] \neq \mathcal{D}[\![s_1]\!]$. Then, by Proposition 15, we have $\mathcal{C}[\![s_0]\!] \neq \mathcal{C}[\![s_1]\!]$. Thus, by Lemma 3, there exists $\widehat{s} \in \mathcal{L}[\emptyset]$ such that $\mathcal{O}[\![s_0 \| \widehat{s}]\!] \neq \mathcal{O}[\![s_1 \| \widehat{s}]\!]$. Therefore, by putting $S \equiv (X \| \widehat{s})$ with X being an arbitrary variable, we have $\mathcal{O}[\![S[s_0/X]]\!] \neq \mathcal{O}[\![S[s_1/X]]\!]$. Thus we obtain (43), and therefore the $\Leftarrow$-part of (42). $\blacksquare$

7 Concluding Remarks

There are two directions for future research.

First, it remains for future study to investigate what we can do with CCS^{++}, e.g., to investigate the following questions:

- Is it possible to code in CCS^{++} a concurrent object-oriented model? (Example 3 of [21] suggests that such a coding will be possible to some extent, by giving a description of a typical client-server system in CCS^{++}.)
- Are any (efficient) implementations of CCS^{++} possible? (Note that, unlike LOTOS [22], CCS^{++} is a programming language in nature with all the features intended to be implementable in some sense; the former is a specification language with several features not intended to be implemented.)

Another direction for future research is to apply the general scheme described in Sect. 1.1 in order to establish full abstractness results for more complex languages such as the ones treated in [2, 8, 20, 34] (these languages include Ada-style rendezvous, real-time features, shared variables, and/or object-orientation). Particularly interesting target is π-calculus [29]. In [13], Hennessy provided a kind of term model for a version of π-calculus, which model is shown to be fully abstract with respect to a certain behavioral criterion based on testing. But the behavioral criterion in [13] is different from ours based on the weak linear semantics, and domain-theoretic construction of good denotational models for π-calculus remains for future research.

A Appendix

Remark 3 In the proof of a semantic equivalence result, Theorem 13.3, of [30], the following proposition (44) is claimed to hold and is used as a crucial intermediate step of the proof (see the end of the second paragraph of [30, page 61]).

$$\forall \ell \in \omega, \forall n \geq \ell + 1, \forall R_1, R_2[\ R_1, R_2 \text{ are closed statements} \Rightarrow P^n(R_1) \cong_\ell P^n(R_2)\ ], \tag{44}$$

where P is a mapping which maps each statement R to $\tilde{P}[R/\xi]$ (with $\tilde{P}$ being an arbitrary statement such that $\mathrm{FV}(\tilde{P}) \subseteq \{\xi\}$), and $\cong_\ell$ is such a modification of Milner's ℓ-nested observation equivalence [27] as takes the notion of divergence into account (see [30, Sect. 12]). (We have $R_1 \cong_0 R_2$ iff $(R_1 \uparrow \Leftrightarrow R_2 \uparrow)$, where $\uparrow$ denotes the possibility of divergence at ϵ. And $\cong_{\ell+1}$ is defined from $\cong_\ell$, just as Milner's $(\ell+1)$-nested observation equivalence is defined from the ℓ-nested observation equivalence.) However, (44) does not hold as is shown below. Let $\tilde{P} \equiv c \cdot (\xi \setminus c)$, and P be the syntactic mapping which maps each statement R to $\tilde{P}[R/\xi]$. Then $P(\mathrm{div}) \equiv c \cdot (\mathrm{div} \setminus c) \xrightarrow{c} (\mathrm{div} \setminus c) \uparrow$, and therefore, $P^2(\mathrm{div}) \equiv c \cdot (P(\mathrm{div}) \setminus c) \xrightarrow{c} (P(\mathrm{div}) \setminus c) \xrightarrow{\tau} (\mathrm{div} \setminus c) \uparrow$. Thus, we have $P^2(\mathrm{div}) \stackrel{c}{\Rightarrow} (\mathrm{div} \setminus c) \uparrow$. If (44) holds for $\ell = 1$, $n = 2$, $R_1 \equiv \mathrm{div}$, and $R_2 \equiv \mathrm{stop}$, then there must exist a statement R_2' such that $P^2(\mathrm{stop}) \stackrel{c}{\Rightarrow} R_2' \uparrow$, but this is clearly impossible. ∎

References

[1] E. Astesiano and E. Zucca (1984), Parametric channels via label expressions in CCS, in *Theoretical Computer Science, Vol. 33*, pp. 45–63.

[2] J.W. de Bakker and E.P. de Vink (1993), Rendez-vous with metric semantics, *New Generation Computing, Vol. 12, No. 1*, pp. 53–90.

[3] J.W. de Bakker and J.I. Zucker (1982), Processes and the denotational semantics of concurrency, *Information and Control, Vol. 54*, pp. 70–120.

[4] J.A. Bergstra, J.W. Klop, and E.-R. Olderog (1987), Failures without chaos: a new process process semantics for fair abstraction, in *Proceedings of IFIP Conference on Formal Description of Programming Concepts III*, Ebberup 1986 (M. Wirsing, ed.), North-Holland, pp. 1134–1177.

[5] J.A. Bergstra, J.W. Klop, and E.-R. Olderog (1988), Readies and failures in the algebra of communicating processes, *SIAM J. of Computing Vol. 17, No. 6*, pp. 1134–1177.

[6] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe (1985), A theory of communicating sequential processes, *Journal of the Association for Computing Machinery, Vol. 31*, pp. 560–599.

[7] S.D. Brookes and A.W. Roscoe (1984), An improved failures model for communicating processes, in *Lecture Notes in Computer Science, Vol. 197*, pp. 281–305, Springer, Berlin.

[8] F. van Breugel (1991), Comparative semantics for a real-time programming languages with integration, In *Proceeding of TAPSOF/CAAP'91, Lecture Notes in Computer Science, Vol. 493* (S. Abramsky and T.S.E. Maibaum, eds.), pp. 397-411, Springer, Berlin.

[9] J. Dugundji (1966), *Topology*, Allyn and Bacon, Boston.

[10] M.H. Goldsmith, A.W. Roscoe, and B.G.O. Scott (1993), *Denotational Semantics for occam 2*, Technical Monograph PRG-108, Oxford University Computing Laboratory.

[11] C.A. Gunter (1993), *Semantics of Programming Languages: Structures and Techniques*, MIT Press.

[12] M. Hennessy (1988), *Algebraic Theory of Processes*, MIT Press.

[13] M. Hennessy (1992), *A Model for the π-Calculus*, University of Sussex (obtained by anonymous ftp from `<ftp.cogs.sussex.ac.uk>`).

[14] M. Hennessy and A. Ingófsdóttir (1993), A theory of communicating processes with value passing, *Information and Computation Vol 107*, pp. 202–236.

[15] C.A.R. Hoare (1985), *Communicating Sequential Processes*, Prentice Hall.

[16] E. Horita (1992), Fully abstract models for communicating processes with respect to weak linear semantics with divergence, *IEICE Transactions on Information and Systems, Vol. E75-D, No. 1*, pp. 64–77.

[17] E. Horita (1992), Deriving denotational models fro nonuniform concurrency from structured operational semantics, *IPSJ Technical Report, Vol. 92, No. 67, 92-PRG-8*, pp. 1–8.

[18] E. Horita (1993), *Fully Abstract Models for Concurrent Languages*, Ph.D. thesis, the Free University of Amsterdam (also appeared as *ECL Technical Report, No. 8939*).

[19] E. Horita (1994), *Deriving failures models for nonuniform concurrency from Structured Operational Semantics*, to appear in *New Generation Computing*.

[20] E. Horita, J.W. de Bakker, and J.J.M.M. Rutten (1990), *Fully Abstract Denotational Models for Nonuniform Concurrent Languages*, *CWI Report CS-R9027*, Amsterdam. (To appear in *Information and Computation.*)

[21] E. Horita and F.-J. de Vries (1994), *A fully abstract denotational model for communicating processes with label-passing*, to appear as an ECL Technical Report (a preliminary version was distributed at Workshop on Concurrency Theory and Its Application, Research Institute of Mathematical Science, Kyoto, July 25–27, 1994).

[22] ISO (1989), *Information Processing Systems – Open Systems Interconnection – LOTOS – A Formal Description Technique based on the Temporal Ordering of Observational Behaviour*, International Standard ISO 8807.

[23] J.N. Kok and J.J.M.M. Rutten (1990), Contractions in comparing concurrency semantics, in *Theoretical Computer Science, Vol. 76*, pp. 179–222.

[24] J.-J.Ch. Meyer and E.P. de Vink (1990), Application of compactness in the Smyth powerdomain of streams, *Theoretical Computer Science, Vol. 57*, pp. 251–282.

[25] J.-J.Ch. Meyer and E.-R. Olderog (1990), Hiding in stream semantics of uniform concurrency, *Acta Informatica, Vol. 27*, pp. 381–397.

[26] R. Milner (1975), Processes: a mathematical model of computing agents, in *Proceedings of Logic Colloquium 73* (H.E. Rose, J.C. Shepherdson, eds.), pp. 157–173, North-Holland, Amsterdam.

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

[28] R. Milner (1989), *Communication and Concurrency*, Prentice Hall International.

[29] R. Milner, J. Parrow, and D. Walker (1992), A calculus of mobile processes, I and II, *Information and Computation, Vol. 100*, pp. 1–40 and pp. 41–77.

[30] E.-R. Olderog and C.A.R. Hoare (1986), Specification-oriented semantics for communicating processes, *Acta Informatica, Vol. 23*, pp. 9–66.

[31] Plotkin, G.D. (1981), *A Structural Approach to Operational Semantics*, Report DAIMI FN-19, Comp. Sci. Dept., Aarhus Univ.

[32] Plotkin, G.D. (1981), *Post-Graduate Lecture Notes in Advanced Domain Theory*, Department of Computer Science, University of Edinburgh.

[33] J.J.M.M. Rutten (1989), Correctness and full abstraction of metric semantics for concurrency, in *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency* (J.W. de Bakker, W.P. de Roever, G. Rozenberg, eds.), *Lecture Notes in Computer Science Vol. 354*, pp. 628–658, Springer, Berlin.

[34] J.J.M.M. Rutten (1990), Semantic correctness for a parallel object-oriented language, *SIAM J. of Computing Vol. 19, No. 2*, pp. 341–383.

[35] Rutten, J.J.M.M. (1990), Deriving denotational models for bisimulation from Structured Operational Semantics, in *Proceedings of IFIP TC2 Working Conference on Programming Concepts and Methods* (M. Broy, C.B. Jones, eds.).

[36] M.B. Smyth (1987), Powerdomains, *Journal of Computer and System Sciences, Vol. 16*, pp. 23–26.

[37] J. Stoy (1977), *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*, MIT Press.

[38] K.J. Turner (1993), *Using Formal Description Techniques: An Introduction to ESTELLE, LOTOS and SDL*, John Wiley & Sons.