数理解析研究所講究録
996巻 1997年 67-81

A Metric Semantics for the π-Calculus Extended with External Events

(Extended Abstract)

Eiichi Horita and Ken Mano
NTT Communication Science Laboratories
E-mail: horita@progn.kecl.ntt.jp, mano@progn.kecl.ntt.jp

Abstract. We investigate the semantics for a concurrent language $\mathcal{L}$ which is based on the π-calculus and extended with *external events* (such as inputs from the environment and outputs to it). The operational criterion we use for $\mathcal{L}$ is the *strong bisimilarity* based on the *reduction system* augmented with external events. We construct a metric semantics for $\mathcal{L}$ and establish its correctness with respect to the operational criterion.

Contents

1 Introduction

We investigate the semantics for a concurrent language $\mathcal{L}$ which is based on the π-calculus extended with *external events* (such as inputs from the environment and outputs to it). This language $\mathcal{L}$ is a simplified version of our language *Nepi*, which we designed on the basis of the π-calculus for network/distributed programming, giving an experimental implementation on networks [11, 12, 13].

The operational criterion we use for $\mathcal{L}$ is the *strong bisimilarity* based on the *reduction system* augmented with external events. (We base our operational criterion on the reduction system of [16, 17], rather than the original labeled transition system of [18], because we feel the former represents communication within the system more compactly than the

latter. We augmented the the reduction system with external events, because we feel this is more natural than encoding external events—such as outputting data through an I/O port, drawing graphics on a display, etc.—into the original syntax of the π-calculus.) We construct a metric semantics for $\mathcal{L}$ by using the standard methodology of metric semantics [1, 2] and by applying the methods for (automatically) deriving denotational models from transition rules [8, 9, 10]. After this, we establish the *correctness* of $\mathcal{M}$ with respect to the operational criterion based on the reduction system. This is done by using an intermediate operational semantics $\mathcal{O}_\nu$, which is defined on the basis of a labeled transition system (LTS) featuring an explicit naming facility for restricted (or freshly generated) gate names.[1] (The idea underlying this LTS is essentially the same as that underlying the $\nu\pi$-calculus, an implementation-oriented calculus introduced in [8, 9, 10], where it is shown that the $\nu\pi$-calculus is equivalent to the π-calculus at a certain level of abstraction.)

The rest of this paper is organized as follows. In Sect. 2, the language $\mathcal{L}$ is introduced. In Sect. 3, the reduction system for $\mathcal{L}$ is given. In Sect. 4, we define the metric semantics $\mathcal{M}$ for $\mathcal{L}$. In Sect. 5, we establish the correctness of $\mathcal{M}$ with respect to strong bisimilarity based on the reduction system of Sect. 2. Finally, in Sect. 6, several remarks are given concerning related work and directions for further study.

2 Language $\mathcal{L}$ Based on the π-Calculus with Events

In this section, we introduce the concurrent language $\mathcal{L}$ which is based on the π-calculus and extended with *external events* (such as inputs from the environment and outputs to it). This language $\mathcal{L}$ is a simplified version of our language *Nepi*, which we designed on the basis of the π-calculus for network/distributed programming, giving an experimental implementation on networks [11, 12, 13]. We omitted several features of Nepi from $\mathcal{L}$ to make the semantic discussion simple, preserving the essence of the ideas underlying the design of Nepi. More specifically, we replaced parameterized external events by a simple set of (atomic) external events, and omitted several features such as built-in data-types, passing of data of those types, conditionals, and alternative composition. The language $\mathcal{L}$ is very similar to the version of the π-calculus in [17], except that $\mathcal{L}$ features external events and employ recursion instead of the replication operator in [17].

2.1 Notational Preliminaries

The phrase "let $(x \in)\ X$ be ..." introduces a set X with variable x ranging over X. For a set X, the powerset of X is denoted by $\wp(X)$, and the set of finite subsets of X is denoted by $\wp_{\mathrm{fin}}(X)$. We use the standard λ-notation $(\lambda x \in X.\ E(x))$ to denote the mapping which maps $x \in X$ to $E(x)$. We sometimes write $\langle E_x \rangle_{x \in X}$ or $\langle E_x |\ x \in X \rangle$ for $(\lambda x \in X.\ E(x))$. For two sets X and Y, the function space from X to Y is denoted by $(X \to Y)$. The set of natural numbers $0, 1, \ldots$ is denoted by ω. The reflexive transitive closure of a binary relation R on a set X is denoted by R^*. We use the two notations $(a_1, \ldots, a_n)$ and $\langle a_1, \ldots, a_n \rangle$ interchangeably to denote the n-tuple consisting of the components $a_1, \ldots, a_n$.

For a metric space (X, d) and $Y \subseteq X$, the (topological) *closure* of Y in (X, d) is denoted by Y^{cls}. For a contraction Φ from a complete metric space X to itself, the unique fixed-point of Φ is denoted by $\mathit{fix}(\Phi)$, where the existence of $\mathit{fix}(\Phi)$ is guaranteed by Banach's fixed-point theorem (see [1, 2]). The set of all closed subsets of a metric space (X, d) is denoted by $\wp_{\mathrm{cl}}(X)$.

[1] We use the term *gate* borrowed from LOTOS [14]. In some references, other terms such as *channel* and *port* are used for the same concept.

2.2 Definition of the Language $\mathcal{L}$

Let $\mathbf{E}$ be the set of *external events* such as inputs from the environment and outputs to it. And let $(a \in)\ \mathbf{A} = \mathbf{E} \cup \{\tau\}$, where τ is a symbol representing an *internal* (or *unobservable*) action. We use three sets $\mathcal{V}_{\text{gate}}$, $\mathcal{V}_{\text{proc}}$, and $\mathcal{V}^{(1)}_{\text{proc}}$ of variables. $\mathcal{V}_{\text{gate}}$ is the set of variables for *gates*, and $\mathcal{V}_{\text{proc}}$ (resp. $\mathcal{V}^{(1)}_{\text{proc}}$) is the set of variables for processes (resp. parameterized processes with one parameter).[2] We put $\mathcal{V}^{*}_{\text{proc}} = \mathcal{V}_{\text{proc}} \cup \mathcal{V}^{(1)}_{\text{proc}}$.

Definition 1 (Process Expressions and Programs) Let $(v \in)\ \mathcal{L}_{\text{gate}}$ be the set of expressions representing gates. In the setting of this section, we have $\mathcal{L}_{\text{gate}} = \mathcal{V}_{\text{gate}}$. (Later, in Sect. 4, we introduce a set $\mathbf{G}$ of constant symbols representing gates. In that setting, we have $\mathcal{L}_{\text{gate}} = \mathcal{V}_{\text{gate}} \cup \mathbf{G}$.)

(1) First, we simultaneously define the set $(S \in)\ \mathcal{L}$ of processes by the following grammar:

$$S ::= \delta \mid (\mathbf{e}\ S) \mid (!\ v_1\ v_2\ S) \mid (?\ v\ x\ S) \mid (\nu\ x\ S) \mid (\|\ S_1\ S_2) \mid X \mid (\mathcal{X}\ v),$$

where $\mathbf{e} \in \mathbf{E}$, $x \in \mathcal{V}_{\text{gate}}$, $X \in \mathcal{V}_{\text{proc}}$ and $\mathcal{X} \in \mathcal{V}^{(1)}_{\text{proc}}$.[3] We write $(\nu\ \langle x, x' \rangle\ S)$ as an abbreviation of $(\nu\ x\ (\nu\ x'\ S))$.

(2) An occurrence of a gate variable $x \in \mathcal{V}_{\text{gate}}$ is said to be *bound* in a statement $S \in \mathcal{L}$ iff it appears in a part of the form $(?\ v\ x\ \cdots)$ or $(\nu\ x\ \cdots)$; occurrences which are not bound are said to be *free*. For $S \in \mathcal{L}$, we define $fn(S)$ to be the set of gate variables x which has a free occurrence in S. For $\mathcal{V} \subseteq \mathcal{V}_{\text{gate}}$, let $\mathcal{L}_{\text{gate}}[\mathcal{V}] = \{v \in \mathcal{L}_{\text{gate}} \mid fn(v) \subseteq \mathcal{V}\}$ and $\mathcal{L}[\mathcal{V}] = \{S \in \mathcal{L} \mid fn(S) \subseteq \mathcal{V}\}$. Elements of $\mathcal{L}[\emptyset]$ are called *closed statements*.

(3) For $Z \in \mathcal{V}^{*}_{\text{proc}}$ and $S \in \mathcal{L}$, we say S satisfies the *guardedness condition* with respect to Z iff every free occurrence of Z in S appears in part of the form $(e\ \cdots)$, $(!\ v_1\ v_2\ \cdots)$, or $(?\ v\ y\ \cdots)$. Note that if Z does not occur in S at all, then Z is guarded in S by definition.

(4) We define the set $(\Delta \in)\ \mathcal{DC}$ of *declaration components* by

$$\begin{aligned}\mathcal{DC} = \{\,&(X, S) \mid X \in \mathcal{V}_{\text{proc}} \wedge S \in \mathcal{L}[\emptyset] \wedge \\ &S \text{ satisfies the guardedness condition w.r.t. every } Z \in \mathcal{V}^{*}_{\text{proc}}\} \\ \cup \{\,&(\mathcal{X}, (x, S)) \mid \mathcal{X} \in \mathcal{V}^{(1)}_{\text{proc}} \wedge x \in \mathcal{V}_{\text{gate}} \wedge S \in \mathcal{L}[\{x\}] \wedge \\ &S \text{ satisfies the guardedness condition w.r.t. every } Z \in \mathcal{V}^{*}_{\text{proc}}\}.\end{aligned}$$

From $\mathcal{DC}$, we define the set $(D \in)\ \mathcal{D}$ of *declarations* to be the set of mappings

$$D : \mathcal{V}^{*}_{\text{proc}} \to \mathcal{L} \cup \{(x, S) \mid x \in \mathcal{V}_{\text{gate}} \wedge S \in \mathcal{L}[\{x\}]\}$$

such that $\forall Z \in \mathcal{V}^{*}_{\text{proc}}[\ (Z, D(Z)) \in \mathcal{DC}\ ]$.

(5) For $\mathcal{V} \subseteq \mathcal{V}_{\text{gate}}$, we define $\mathcal{L}_{\text{prog}}[\mathcal{V}] = \mathcal{D} \times \mathcal{L}[\mathcal{V}]$. Pairs $(D, s) \in \mathcal{L}_{\text{prog}}[\emptyset] = \mathcal{D} \times \mathcal{L}[\emptyset]$ are called *programs*.[4] ∎

---

[2] For simplicity, we only treat parameterized processes with one parameter; processes with several parameters can be treated similarly, with no essential difficulty.

[3] We define syntactic entities in $\mathcal{L}$ as S-expressions used in Lisp, for convenience in avoiding ambiguity in the syntax and in distinguishing syntactic descriptions from semantic ones.

[4] Here we require, for simplicity of semantic discussion, that every variable in $\mathcal{V}_{\text{proc}} \cup \mathcal{V}^{(1)}_{\text{proc}}$ be declared to have its body. In actual programming, however, we can only specify a finite set of declaration components (see [11, 12, 13]). We treat a finite set $\{(Z_1, B_1), \cdots, (Z_n, B_n)\}$ of declaration components as specifying a declaration D such that $D(Z_i) = B_i$ for $i \in \{1, \ldots, n\}$ and for $Z \notin \{Z_1, \ldots, Z_n\}$, $D(Z)$ is a certain default value (such as δ).

3 Reduction System for $\mathcal{L}$ with External Events

In this section, we give a reduction system (augmented with external events) for $\mathcal{L}$. We define the *strong bisimilarity* based on the *reduction system* (augmented with external events), as the operational criterion for $\mathcal{L}$.

We use (as the basis of our operational criterion) the reduction system of [16, 17], rather than the original labeled transition system of [18], because we feel the former represent communications within the system in question more compactly than the latter. We augmented the the reduction system with external events, because we feel this is more natural than encoding external events—such as outputting data through an I/O port, drawing graphics on a display, etc.—into the original syntax of the π-calculus. It seems that the programming system Pict based on the π-calculus treats external I/O in a similar fashion, as additional elements extraneous to the π-calculus [19].

3.1 Structural Congruence

Definition 2 Let $\equiv_\alpha$ denote α-congruence on $\mathcal{L}$. Following [16, Sect. 2.3], we define the *structural congruence* $\hat{\equiv}$ over $\mathcal{L}_{\text{prog}}$ as the smallest congruence relation satisfying the following eight laws SC1–SC8.

(1) Two process expressions that are α-convertible with each other are congruent:

SC1: $S_1 \equiv_\alpha S_2 \Rightarrow (D, S_1) \hat{\equiv} (D, S_2)$.

(2) We have the Abelian semigroup laws SC2 and SC3 below for parallel composition:

SC2: $(D, (\| \; S_1 \; (\| \; S_2 \; S_3))) \hat{\equiv} (D, (\| \; (\| \; S_1 \; S_2) \; S_3))$.

SC3: $(D, (\| \; S_1 \; S_2)) \hat{\equiv} (D, (\| \; S_2 \; S_1))$.

(3) For process expressions of the form $(\nu \cdots)$, we have the following three laws:

SC4: $(D, (\nu \; \langle x, x' \rangle \; S)) \hat{\equiv} (D, (\nu \; \langle x', x \rangle \; S))$.

SC5: $x \notin fn(S) \Rightarrow (D, (\nu \; x \; S)) \hat{\equiv} (D, S)$.

SC6: $x \notin fn(S') \Rightarrow (D, (\| \; (\nu \; x \; S_1) \; S_2)) \hat{\equiv} (D, (\nu \; x \; (\| \; S_1 \; S_2)))$.

(4) Each process variable $Z \in \mathcal{V}^*_{\text{proc}}$ is congruent to its body:

SC7: $(X, s_X) \in D \Rightarrow (D, X) \hat{\equiv} (D, s_X)$.

SC8: $(\mathcal{X}, (x, S_{\mathcal{X}})) \in D \Rightarrow (D, (\mathcal{X} \; v)) \hat{\equiv} (D, S_{\mathcal{X}}[v/x])$. ∎

3.2 Reduction System with External Events

Definition 3 For each $a \in \mathbf{A}$, a transition relation $\xrightarrow{a}$ is defined between elements of $\mathcal{L}_{\text{prog}}[\mathcal{V}_{\text{gate}}] = \mathcal{D} \times \mathcal{L}$. As usual, we write

$$(D_1, S_1) \xrightarrow{a} (D_2, S_2) \tag{1}$$

to mean that $\langle (D_1, S_1), (D_2, S_2) \rangle \in \xrightarrow{a}$. We shall always be concerned with (1) only in cases where $D_1 = D_2$. For $D \in \mathcal{D}$ and $S_1, S_2 \in \mathcal{L}$, we often write $S_1 \xrightarrow{a}_D S_2$ to mean that $(D, S_1) \xrightarrow{a} (D, S_2)$ (cf. [2, Definition 1.7]). We sometimes simply write $S_1 \xrightarrow{a} S_2$ for $S_1 \xrightarrow{a}_D S_2$, when it is clear from the context what declaration D is in question.

EVE:

$$(e\ S) \xrightarrow{e}_D S \quad (e \in \mathbf{E}).$$

RES:

$$\frac{S \xrightarrow{a}_D S'}{(\nu\ x\ S) \xrightarrow{a}_D (\nu\ x\ S')} \quad \text{for any } a \in \mathbf{A}.$$

PAR:

$$\frac{S_1 \xrightarrow{a}_D S_1'}{\begin{array}{l}(\|\ S_1\ S_2) \xrightarrow{a}_D (\|\ S_1'\ S_2) \\ (\|\ S_2\ S_1) \xrightarrow{a}_D (\|\ S_2\ S_1')\end{array}} \quad \text{for any } a \in \mathbf{A}.$$

COM:

$$(\|\ (!\ v_1\ v_1'\ S_1)\ (?\ v_2\ x\ S_2)) \xrightarrow{\tau}_D (\|\ S_1\ S_2[v_1'/x]), \quad \text{if } v_1 \equiv v_2.$$

STR:

$$\frac{(D, S_1') \tilde{\equiv} (D, S_1), S_1 \xrightarrow{a}_D S_2, (D, S_2) \tilde{\equiv} (D, S_2')}{S_1' \xrightarrow{a}_D S_2'}. \blacksquare$$

By the above definition, we have

$$(D_1, S_1) \xrightarrow{a} (D_2, S_2) \Rightarrow D_1 = D_2.$$

Thus, we may suppose that for each $D \in \mathcal{D}$ and $a \in \mathbf{A}$, a transition relation $\xrightarrow{a}_D \subseteq \mathcal{L} \times \mathcal{L}$ is defined.

4 Metric Semantics $\mathcal{M}$ for $\mathcal{L}$

In this section, we construct a metric semantics for $\mathcal{L}$ by using a standard methodology of metric semantics [1, 2].

To construct $\mathcal{M}$, we first a labeled transition system (LTS) featuring an explicit naming facility for restricted (or freshly generated) gate names. (The idea underlying this LTS is essentially the same as that underlying the $\nu\pi$-calculus, an implementation-oriented calculus introduced in [8, 9, 10], where it is shown that the $\nu\pi$-calculus, which is much more suited to distributed implementation, is equivalent to the π-calculus at a certain level of abstraction.) On the basis of the LTS, an intermediate operational semantics $\mathcal{O}_\nu$, which is used to show the correctness of $\mathcal{M}$ in Sect. 5. Then, we define semantic operations based the transition rules which define the LTS, by applying the methods for (automatically) deriving denotational models from transition rules [8, 9, 10]. Finally, the metric model $\mathcal{M}$ is constructed by using the semantic operations as interpretations of the syntactic operators of $\mathcal{L}$.

4.1 Labeled Transition System for $\mathcal{L}$ via Explicit Naming

Let $(g \in)\ \mathbf{G}$ be the infinite set *gate-names*; let Γ be a variable ranging over $\wp_{\text{fin}}(\mathbf{G})$. We define $\tilde{\mathcal{L}}$ in the same way as $\mathcal{L}$ except that each $g \in \mathbf{G}$ may appear in elements of $\tilde{\mathcal{L}}$ as a constant symbol representing a gate. Clearly $\mathcal{L} \subseteq \tilde{\mathcal{L}}$. For $\mathcal{V} \subseteq \mathcal{V}_{\text{gate}}$, we define $\tilde{\mathcal{L}}_{\text{prog}}[\mathcal{V}]$ as $\mathcal{L}_{\text{prog}}[\mathcal{V}]$, using $\tilde{\mathcal{L}}[\mathcal{V}]$ instead of $\mathcal{L}[\mathcal{V}]$.

Definition 4 We define the set $(k \in)\ \Lambda$ of *labels* by

$$\Lambda = \{(\Gamma, \alpha, \Gamma') |\ \Gamma, \Gamma' \in \wp_{\text{fin}}(\mathbf{G}) \wedge \Gamma \subseteq \Gamma' \wedge \alpha \in \tilde{\mathbf{A}}_\nu\}. \tag{2}$$

From Λ, the set $\mathbf{P}$ of *processes* is defined as the solution of the following domain equation:

$$\mathbf{P} \cong \wp_{\mathrm{cl}}(\Lambda \times \mathbf{P}), \tag{3}$$

where $\cong$ means that there exists an isometry Φ_{P} from $\mathbf{P}$ onto $\wp_{\mathrm{cl}}(\Lambda \times \mathbf{P})$. (For the definition of the metric on $\wp_{\mathrm{cl}}(\Lambda \times \mathbf{P})$ and for the proof of the unique existence of a solution to (3), see [1, 2].) Via this isometry Φ_{P}, we identify each set $X \in \wp_{\mathrm{cl}}(\Lambda \times \mathbf{P})$ with $\Phi_{\mathrm{P}}^{-1}(X) \in \mathbf{P}$; thus we sometimes write $\{(k,p) \in \Lambda \times \mathbf{P} | \cdots\}$ to refer to the element $\Phi_{\mathrm{P}}^{-1}(\{(k,p) \in \Lambda \times \mathbf{P} | \cdots\})$ of $\mathbf{P}$, when we know that $\{(k,p) \in \Lambda \times \mathbf{P} | \cdots\}$ is closed. ∎

Definition 5 We define

$$(a \in)\ \mathbf{A}_\nu = \{(g!, g') |\ g, g' \in \mathbf{G}\} \cup \{(g?, g') |\ g, g' \in \mathbf{G}\} \cup \mathbf{A}.$$

Let ι be a symbol representing the *generation of a fresh gate-name*, with $\iota \notin \mathbf{A}$. We put $(\alpha \in)\ \tilde{\mathbf{A}}_\nu = \mathbf{A}_\nu \cup \{\iota\}$. For $v \in \mathcal{L}_{\mathrm{gate}}[\emptyset]$, let $[\![v]\!]$ be the evaluation of v.

For each $\alpha \in \tilde{\mathbf{A}}_\nu$, a transition relation $\overset{a}{\rightarrowtail}$ is defined between elements of $\tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset] \times \wp_{\mathrm{fin}}(\mathbf{G}) = (\mathcal{D} \times \tilde{\mathcal{L}}[\emptyset]) \times \wp_{\mathrm{fin}}(\mathbf{G})$. For $D \in \mathcal{D}$, $s_1, s_2 \in \tilde{\mathcal{L}}[\emptyset]$ and $\Gamma_1, \Gamma_2 \in \wp_{\mathrm{fin}}(\mathbf{G})$, we often write $\langle s_1, \Gamma_1 \rangle \overset{\alpha}{\rightarrowtail}_D \langle s_2, \Gamma_2 \rangle$ to mean that $\langle (D, s_1), \Gamma_1 \rangle \overset{\alpha}{\rightarrowtail} \langle (D, S_2), \Gamma_2 \rangle$. We sometimes simply write $\langle s_1, \Gamma_1 \rangle \overset{\alpha}{\rightarrowtail} \langle s_2, \Gamma_2 \rangle$ for $\langle s_1, \Gamma_1 \rangle \overset{\alpha}{\rightarrowtail}_D \langle s_2, \Gamma_2 \rangle$, when it is clear from the context what declaration D is in question.

EVE$_\nu$:

$$\langle (e\ s), \Gamma \rangle \overset{e}{\rightarrowtail}_D \langle s, \Gamma \rangle \quad (e \in \mathbf{E}).$$

RES$_\nu$:

$$\langle (\nu\ x\ S), \Gamma \rangle \overset{\iota}{\rightarrowtail}_D \langle S[g/x], \Gamma \cup \{g\} \rangle, \quad \text{for any } g \in \mathbf{G} \setminus \Gamma.$$

OUT$_\nu$

$$\langle (!\ v_1\ v_2\ s), \Gamma \rangle \overset{([\![v_1]\!]!, [\![v_2]\!])}{\rightarrowtail}_D \langle s, \Gamma \rangle.$$

IN$_\nu$ $\quad \langle (?\ v\ x\ S), \Gamma \rangle \overset{([\![v]\!]?, g)}{\rightarrowtail}_D \langle S[g/x], \Gamma \rangle \quad (g \in \mathbf{G}).$

PAR$_\nu^\iota$:

$$\frac{\langle s_1, \Gamma \rangle \overset{\iota}{\rightarrowtail}_D \langle s_1', \Gamma' \rangle}{\begin{array}{l}\langle (\|\ s_1\ s_2), \Gamma \rangle \overset{\iota}{\rightarrowtail}_D \langle (\|\ s_1'\ s_2), \Gamma' \rangle \\ \langle (\|\ s_2\ s_1), \Gamma \rangle \overset{\iota}{\rightarrowtail}_D \langle (\|\ s_2\ s_1'), \Gamma' \rangle\end{array}}.$$

PAR$_\nu$: We stipulate that the priority of ι is higher than that of $a \in \mathbf{A}_\nu$ (cf. [4]). This is formally stated by[5]

$$\frac{\neg(\langle s_2, \Gamma \rangle \overset{\iota}{\rightarrowtail}_D), \langle s_1, \Gamma \rangle \overset{a}{\rightarrowtail}_D \langle s_1', \Gamma' \rangle}{\begin{array}{l}\langle (\|\ s_1\ s_2), \Gamma \rangle \overset{a}{\rightarrowtail}_D \langle (\|\ s_1'\ s_2), \Gamma' \rangle \\ \langle (\|\ s_2\ s_1), \Gamma \rangle \overset{a}{\rightarrowtail}_D \langle (\|\ s_2\ s_1'), \Gamma' \rangle\end{array}} \quad (a \in \mathbf{A}_\nu).$$

[5] By the rules stated in this definition, we have

$$\langle s_1, \Gamma \rangle \overset{a}{\rightarrowtail}_D \langle s_1', \Gamma' \rangle \Rightarrow \neg(\langle s_1, \Gamma \rangle \overset{\iota}{\rightarrowtail}_D) \quad (a \in \mathbf{A}_\nu).$$

Thus, adding the condition that $\neg(\langle s_1, \Gamma \rangle \overset{\iota}{\rightarrowtail}_D)$ in the antecedent of rule PAR$_\nu$ makes no change in the resulting transition system. Likewise, adding the condition that $\neg(\langle s_i, \Gamma \rangle \overset{\iota}{\rightarrowtail}_D)$ $(i = 1, 2)$ in the antecedent of rule COM $_\nu$ below makes no change.

COM$_\nu$:

$$\frac{\langle s_1,\Gamma\rangle \overset{(g!,g')}{\rightarrowtail}_D \langle s_1',\Gamma\rangle, \langle s_2,\Gamma\rangle \overset{(g?,g')}{\rightarrowtail}_D \langle s_2',\Gamma\rangle}{\begin{array}{c}\langle(\!\| s_1\ s_2),\Gamma\rangle \overset{\tau}{\rightarrowtail}_D \langle(\!\| s_1'\ s_2'),\Gamma\rangle \\ \langle(\!\| s_2\ s_1),\Gamma\rangle \overset{\tau}{\rightarrowtail}_D \langle(\!\| s_2'\ s_1'),\Gamma\rangle\end{array}}.$$

REC$_\nu$: If $(X, s_X) \in D$, then $\dfrac{\langle s_X,\Gamma\rangle \overset{\alpha}{\rightarrowtail}_D \langle s',\Gamma'\rangle}{\langle X,\Gamma\rangle \overset{\alpha}{\rightarrowtail}_D \langle s',\Gamma'\rangle}$.

REC$_\nu^{(1)}$: If $(\mathcal{X}, (x, S_\mathcal{X})) \in D$, then $\dfrac{\langle S_\mathcal{X}[v/x],\Gamma\rangle \overset{\alpha}{\rightarrowtail}_D \langle s,\Gamma\rangle}{\langle(\mathcal{X}\ v),\Gamma\rangle \overset{\alpha}{\rightarrowtail}_D \langle s,\Gamma\rangle}$. ∎

In some rules in the above definition, we use negation in the premise-part. The well-definedness of a transition system based on transition rules with negation is not always obvious. In the above, however, we can define the transition system in two stages: the first sage for $\overset{\iota}{\rightarrowtail}$, and the second stage for $\overset{a}{\rightarrowtail}$ $(a \in \mathbf{A}_\nu)$. Then, the well-definedness of the transition system will be obvious (cf. [4]).

Remark 1 In the $\nu\pi$-calculus in [11, 12, 13], the three rules OUT$_\nu$, IN$_\nu$ and COM$_\nu$ are replaced by the following two rules

COM$'_\nu$:

$$(\!\| (!\ v_1\ v_2\ s)\ (?\ v_1'\ x\ s')) \overset{\tau}{\rightarrowtail}_D (\!\| s\ s'[v_2/x]), \quad \text{if } [\![v_1]\!] = [\![v_1']\!].$$

STR$_\nu$

$$\frac{(D, s_1') \cong (D, s_1), \langle s_1,\Gamma\rangle \overset{\alpha}{\rightarrowtail}_D \langle s_2,\Gamma'\rangle, (D, s_2) \cong (D, s_2')}{\langle s_1',\Gamma\rangle \overset{\alpha}{\rightarrowtail}_D \langle s_2',\Gamma'\rangle} \quad (\alpha \in \mathbf{A} \cup \{\iota\}),$$

where $\cong$ is the smallest congruence relation on $\tilde{\mathcal{L}}_{\text{prog}}$ satisfying the associativity and commutativity laws for $\|$ (i.e., equations SC2 and SC3 in Definition 2 with $\tilde{=}$ replaced by $\cong$) and equations SC7 and SC8 in Definition 2 with $\tilde{=}$ replaced by $\cong$.

Even if we adopt the rule COM$'_\nu$ instead of the three rules, we can obtain Lemma 2 below, which is the key lemma for the proof of Theorem 1 (the main result of this paper). However, using the three rules is more convenient for constructing the metric model $\mathcal{M}$, along the lines of [8, 9, 10]. ∎

From the transition system $\langle \overset{\alpha}{\rightarrowtail} \mid \alpha \in \tilde{\mathbf{A}}_\nu \rangle$, which we call the *lower-level* transition system for $\tilde{\mathcal{L}}$, we define a *higher-level* transition system $\langle \overset{a}{\rightarrow}_\nu \mid \alpha \in \tilde{\mathbf{A}} \rangle$ for $\tilde{\mathcal{L}}$ as follows (for the concepts of *lower-level* and *higher-level* transition systems, cf. [6]).

Definition 6 For each $a \in \mathbf{A}_\nu$, we define a transition relation $\overset{a}{\rightarrow}_\nu$ on $\mathcal{L}_{\text{prog}}[\emptyset] \times \wp_{\text{fin}}(\mathbf{G})$ as follows. For $D \in \mathcal{D}$, $s, s' \in \mathcal{L}[\emptyset]$ and $\Gamma, \Gamma' \in \wp_{\text{fin}}(\mathbf{G})$,

$$\langle (D, s),\Gamma\rangle \overset{a}{\rightarrow}_\nu \langle (D, s'),\Gamma'\rangle \Leftrightarrow \langle (D, s),\Gamma\rangle (\overset{\iota}{\rightarrowtail})^* \overset{a}{\rightarrowtail} \langle (D, s'),\Gamma'\rangle. \tag{4}$$

As in Definition 4, we write $\langle s_1,\Gamma_1\rangle \overset{a}{\rightarrow}_{\nu,D} \langle s_2,\Gamma_2\rangle$ to mean that $\langle (D, s_1),\Gamma_1\rangle \overset{a}{\rightarrow}_\nu \langle (D, s_2),\Gamma_2\rangle$. Also we sometimes simply write $\langle s_1,\Gamma_1\rangle \overset{a}{\rightarrow}_\nu \langle s_2,\Gamma_2\rangle$ for $\langle s_1,\Gamma_1\rangle \overset{a}{\rightarrow}_{\nu,D} \langle s_2,\Gamma_2\rangle$, when it is clear from the context what declaration D is in question. ∎

4.2 Intermediate Operational Semantics $\mathcal{O}_\nu$

From the transition relations $\overset{\alpha}{\rightarrowtail}$ $(\alpha \in \tilde{\mathbf{A}}_\nu)$ we define the intermediate operational model $\mathcal{O}_\nu$ as follows.

Definition 7 (Intermediate Operational Semantics $\mathcal{D}_\nu$)
We define $\mathcal{O}_\nu : \tilde{\mathcal{L}}_{\text{prog}}[\emptyset] \to \mathbf{P}$ so that the following holds for every $(D, s) \in \tilde{\mathcal{L}}_{\text{prog}}[\emptyset]$.

$$\mathcal{O}_\nu[\![(D,s)]\!] = \{\, \langle (\Gamma, \alpha, \Gamma'), \mathcal{O}_\nu[\![(D,s')]\!]\rangle | \quad (\Gamma,\alpha,\Gamma') \in \Lambda \wedge s' \in \tilde{\mathcal{L}}[\emptyset] \wedge \langle (D,s),\Gamma\rangle \overset{\alpha}{\rightarrowtail} \langle (D,s'),\Gamma'\rangle \,\}. \tag{5}$$

The closedness of the right-hand side of (5) follows from the image-finiteness of the LTS $(\tilde{\mathcal{L}}[\emptyset] \times \wp_{\text{fin}}(\mathbf{G}), \langle \overset{\alpha}{\rightarrowtail} \mid \alpha \in \tilde{\mathbf{A}}_\nu\rangle)$. ∎

Formally the mapping $\mathcal{O}_\nu$ is defined as the unique fixed-point of a higher-order contractive mapping (on $(\tilde{\mathcal{L}}_{\text{prog}}[\emptyset] \to \mathbf{P})$) whose definition can be inferred from (5).

4.3 Metric Semantics $\mathcal{M}$ for $\mathcal{L}$

First, we define the denotational interpretations of the operator of $\mathcal{L}$.

Definition 8

(1) For $e \in \mathbf{E}$ and $p \in \mathbf{P}$, we define $\tilde{e}(p) \in \mathbf{P}$ by

$$\tilde{e}(p) = \{\langle (\Lambda, e, \Lambda), p\rangle | \ \Lambda \in \wp_{\text{fin}}(\mathbf{G})\}. \tag{6}$$

(2) For $g, g' \in \mathbf{G}$ and $p \in \mathbf{P}$, we define $\tilde{!}(g, g', p) \in \mathbf{P}$ by

$$\tilde{!}(g,g',p) = \{\langle (\Gamma, (g!, g'), \Gamma), p\rangle | \ \Gamma \in \wp_{\text{fin}}(\mathbf{G})\}. \tag{7}$$

(3) For $\pi \in (\mathbf{G} \to \mathbf{P})$ and $g \in \mathbf{G}$, we define $\tilde{?}(g, \pi) \in \mathbf{P}$ by

$$\tilde{?}(g,\pi) = \{\langle (\Gamma, (g?, g'), \Gamma), \pi(g')\rangle | \ \Gamma \in \wp_{\text{fin}}(\mathbf{G}) \wedge g' \in \mathbf{G}\}. \tag{8}$$

(4) For $\pi \in (\mathbf{G} \to \mathbf{P})$, we define $\tilde{\nu}(\pi) \in \mathbf{P}$ by

$$\tilde{\nu}(\pi) = \{\langle (\Gamma, \iota, \Gamma \cup \{g\}), \pi(g)\rangle | \ \Gamma \in \wp_{\text{fin}}(\mathbf{G}) \wedge g \in \mathbf{G} \setminus \Gamma\}. \tag{9}$$

where g is an arbitrary element of $\mathbf{G}$.

(5) For $p \in \mathbf{P}$, $\Gamma \in \wp_{\text{fin}}(\mathbf{G})$ and $\alpha \in \tilde{\mathbf{A}}_\nu$, let

$$p[\Gamma, \alpha] = \{\langle (\Gamma, \alpha, \Gamma'), p'\rangle | \ \Gamma' \in \wp_{\text{fin}}(\mathbf{G}) \wedge p' \in \mathbf{P} \wedge \langle (\Gamma,\alpha,\Gamma'), p'\rangle \in p\}.$$

For $p_1, p_2 \in \mathbf{P}$, we define $\tilde{\|}(p_1, p_2) \in \mathbf{P}$ by

$$\begin{aligned}
&\tilde{\|}(p_1,p_2) = \\
&\{\langle (\Gamma,\iota,\Gamma'), \tilde{\|}(p_1', p_2)\rangle | \ \langle (\Gamma,\iota,\Gamma'), p_1'\rangle \in p_1\}^{\text{cls}} \\
&\cup \{\langle (\Gamma,\iota,\Gamma'), \tilde{\|}(p_1, p_2')\rangle | \ \langle (\Gamma,\iota,\Gamma'), p_2'\rangle \in p_2\}^{\text{cls}} \\
&\cup \{\langle (\Gamma,a,\Gamma'), \tilde{\|}(p_1', p_2)\rangle | \ p_1[\Gamma,\iota] = p_2[\Gamma,\iota] = \emptyset \wedge \langle (\Gamma,a,\Gamma'), p_1'\rangle \in p_1\}^{\text{cls}} \\
&\cup \{\langle (\Gamma,a,\Gamma'), \tilde{\|}(p_1, p_2')\rangle | \ p_1[\Gamma,\iota] = p_2[\Gamma,\iota] = \emptyset \wedge \langle (\Gamma,a,\Gamma'), p_2'\rangle \in p_2\}^{\text{cls}} \\
&\cup \{\langle (\Gamma,\tau,\Gamma'), \tilde{\|}(p_1', p_2')\rangle | \ p_1[\Gamma,\iota] = p_2[\Gamma,\iota] = \emptyset \wedge \\
&\quad \langle (\Gamma,(g!,g'),\Gamma'), p_2'\rangle \in p_2 \wedge \langle (\Gamma,(g?,g'),\Gamma'), p_2'\rangle \in p_2\}^{\text{cls}} \\
&\cup \{\langle (\Gamma,\tau,\Gamma'), \tilde{\|}(p_1', p_2')\rangle | \ p_1[\Gamma,\iota] = p_2[\Gamma,\iota] = \emptyset \wedge \\
&\quad \langle (\Gamma,(g?,g'),\Gamma'), p_2'\rangle \in p_2 \wedge \langle (\Gamma,(g!,g'),\Gamma'), p_2'\rangle \in p_2\}^{\text{cls}}.
\end{aligned} \tag{10}$$

Formally, the operation $\tilde{\|} : \mathbf{P}^2 \to \mathbf{P}$ is defined as the unique fixed-point of a higher-order contractive mapping $\Phi_{\|} : (\mathbf{P}^2 \to \mathbf{P}) \to (\mathbf{P}^2 \to \mathbf{P})$. The definition of $\Phi_{\|}$ can be derived from (10) by applying the methods for (automatically) deriving denotational models from transition rules [8, 9, 10]. ∎

We can show that each of the semantic operations $\tilde{e}(\cdot)$, $\tilde{!}(g, g', \cdot)$, $\tilde{?}(g, \cdot)$, $\tilde{\nu}(\cdot)$, or $\tilde{\|}(\cdot, \cdot)$ is a nonexpansive mapping from an appropriate domain to a codomain. Moreover, the three operations $\tilde{e}(\cdot)$, $\tilde{!}(g, g', \cdot)$, and $\tilde{?}(g, \cdot)$ are contractions. We use these facts to define the metric model $\mathcal{M}$ below.

From the denotational interpretations of the operators, we define the metric model $\mathcal{M}$ as follows.

Definition 9 (Metric Semantics $\mathcal{M}$)

(1) We define $\mathbf{GEnv} = (\mathcal{V}_{\text{gate}} \to \mathbf{G})$. Let ρ be a variable ranging over **GEnv**. Also we define

$$\mathbf{PEnv} = \{\Pi \in (\mathcal{V}^*_{\text{proc}} \to (\mathbf{P} \cup (\mathbf{G} \to \mathbf{P})))| \\ \forall X \in \mathcal{V}_{\text{proc}}[\ \Pi(X) \in \mathbf{P}\ ] \wedge \forall \mathcal{X} \in \mathcal{V}^{(1)}_{\text{proc}}[\ \Pi(X) \in (\mathbf{G} \to \mathbf{P})\ ]\}.$$

Let Π be a variable ranging over **PEnv**.

(2) For $v \in \mathcal{L}_{\text{gate}}$ and $\rho \in \mathbf{GEnv}$, let $[\![v]\!](\rho)$ be the evaluation of v w.r.t. ρ.

For $S \in \tilde{\mathcal{L}}$, $\Pi \in \mathbf{PEnv}$ and $\rho \in \mathbf{GEnv}$, we define $\tilde{\mathcal{M}}[\![S]\!](\Pi)(\rho)$ by induction on the structure of S using the following rules (11)–(18).

$$\tilde{\mathcal{M}}[\![X]\!](\Pi)(\rho) = \Pi(X) \tag{11}$$

$$\tilde{\mathcal{M}}[\![(\mathcal{X}\ v)]\!](\Pi)(\rho) = \Pi(\mathcal{X})([\![v]\!](\rho)). \tag{12}$$

$$\tilde{\mathcal{M}}[\![\delta]\!](\Pi)(\rho) = \emptyset. \tag{13}$$

$$\tilde{\mathcal{M}}[\![(e\ S)]\!](\Pi)(\rho) = \tilde{e}(\tilde{\mathcal{M}}[\![S]\!](\Pi)(\rho)). \tag{14}$$

$$\tilde{\mathcal{M}}[\![(!\ v\ v'\ S)]\!](\Pi)(\rho) = \tilde{!}([\![v]\!](\rho), [\![v']\!](\rho), \tilde{\mathcal{M}}[\![S]\!](\Pi)(\rho)). \tag{15}$$

$$\tilde{\mathcal{M}}[\![(?\ v\ x\ S)]\!](\Pi)(\rho) = \tilde{?}([\![v]\!](\rho), (\lambda\ g \in \mathbf{G}.\ \tilde{\mathcal{M}}[\![S]\!](\Pi)(\rho[g/x]))). \tag{16}$$

$$\tilde{\mathcal{M}}[\![(\nu\ x\ S)]\!](\Pi)(\rho) = \tilde{\nu}(\lambda g \in \mathbf{G}.\ \tilde{\mathcal{M}}[\![S]\!](\Pi)(\rho[g/x])). \tag{17}$$

$$\tilde{\mathcal{M}}[\![(\|\ S_1\ S_2)]\!](\Pi)(\rho) = \tilde{\|}(\tilde{\mathcal{M}}[\![S_1]\!](\Pi)(\rho), \tilde{\mathcal{M}}[\![S_1]\!](\Pi)(\rho)). \tag{18}$$

(3) From the *guardedness condition* in Definition 1(2) and the fact that all the semantic operations are nonexpansive with the three $\tilde{e}(\cdot)$, $\tilde{!}(g, g', \cdot)$, and $\tilde{?}(g, \cdot)$ being contractive, it follows that the mapping

$$\Phi_D(\rho) = (\lambda\ \Pi \in \mathbf{PEnv}.\ \{(X, \tilde{\mathcal{M}}[\![s]\!](\Pi)(\rho))|\ (X, s) \in D\} \\ \cup \{(\mathcal{X}, (\lambda g.\ \tilde{\mathcal{M}}[\![S]\!](\Pi)(\rho[g/x])))|\ (\mathcal{X}, (x, S)) \in D\}) \tag{19}$$

is a contraction from **PEnv** to itself, for every $\rho \in \mathbf{GEnv}$. Furthermore, the value $\Phi_D(\rho)$ does not depend on ρ, because, by Definition 1(4), we have

$$(X, S) \in D \Rightarrow fn(S) = \emptyset, \text{ and } (\mathcal{X}, (x, S)) \in D \Rightarrow fn(S) \subseteq \{x\}.$$

Let

$$\Pi_D = fix(\Phi_D(\rho)) \in \mathbf{PEnv}, \quad \text{where } \rho \in \mathbf{GEnv} \text{ is arbitrary.}$$

We define

$$\mathcal{M}[\![(D, S)]\!](\rho) = \tilde{\mathcal{M}}[\![S]\!](\Pi_D)(\rho). \tag{20}$$

For $(D, s) \in \tilde{\mathcal{L}}_{\text{prog}}[\emptyset]$, the value $\mathcal{M}[\![(D, s)]\!](\rho)$ does not depend on ρ. We denote this value by $\mathcal{M}[\![(D, s)]\!]$. ∎

We can establish the next lemma which states that $\mathcal{O}_\nu$ is a homomorphism with respect to the operators of $\mathcal{L}$; this lemma will play a key role for establishing the equivalence between $\mathcal{M}$ and $\mathcal{O}_\nu$ (see Lemma 3).

Lemma 1 (Homomorphism Properties of $\mathcal{O}_\nu$)

(1) $\forall e \in \mathbf{E}[\ \mathcal{O}_\nu[\![(D, (\mathbf{e}\ s))]\!] = \tilde{e}(\mathcal{O}_\nu[\![(D, s)]\!])\ ].$

(2) $\forall g, g' \in \mathbf{G}[\ \mathcal{O}_\nu[\![(D, (!\ g\ g'\ s))]\!] = \tilde{!}(g, g', \mathcal{O}_\nu[\![(D, s)]\!])\ ].$

(3) $\mathcal{O}_\nu[\![(D, (?\ g\ x\ S))]\!] = \tilde{?}(g, (\lambda g' \in \mathbf{G}.\ \mathcal{O}_\nu[\![(D, S[g'/x])]\!])).$

(4) $\mathcal{O}_\nu[\![(D, (\nu\ x\ S))]\!] = \tilde{\nu}(\lambda g \in \mathbf{G}.\ \mathcal{O}_\nu[\![(D, S[g/x])]\!]).$

(5) $\mathcal{O}_\nu[\![(D, (\|\ s_1\ s_2))]\!] = \tilde{\|}(\mathcal{O}_\nu[\![(D, s_1)]\!], \mathcal{O}_\nu[\![(D, s_2)]\!]).$ ∎

5 Correctness of $\mathcal{M}$ w.r.t. the Reduction-Based Strong Bisimilarity

In this section, we first establish the correctness of $\mathcal{M}$ with respect to the strong bisimilarity based on the reduction system of Sect. 3. The correctness proof is achieved by using the intermediate operational semantics $\mathcal{O}_\nu$ defined in Sect. 4. In the latter part of this section, we show that $\mathcal{M}$ satisfies all laws SC1–SC6 for structural congruence except for SC5. This fact indicates more adequacy of $\mathcal{M}$ as a semantic model for $\mathcal{L}$ than simply being correct with respect to the strong bisimilarity based on the reduction system.

5.1 Correctness of $\mathcal{M}$ w.r.t. the Reduction-Based Strong Bisimilarity

Definition 10

(1) A *LTS with action set* $\mathbf{A}$ is a triple $\mathcal{T} = (c_0, \mathbf{C}, \langle \xrightarrow{a} \mid a \in \mathbf{A} \rangle)$ such that $c_0 \in C$ and $\forall a \in \mathbf{A}[\ \xrightarrow{a} \subseteq \mathbf{C} \times \mathbf{C}]$. We refer to c_0, $\mathbf{C}$, and $\langle \xrightarrow{a} \mid a \in \mathbf{A} \rangle$ as the *initial configuration*, the *configuration set*, and the *the set of transition relations* of $\mathcal{T}$, respectively.

(2) Let $\mathcal{T}_1 = (c_1, \mathbf{C}_1, \langle \xrightarrow{a}_1 \mid a \in \mathbf{A} \rangle)$ and $\mathcal{T}_2 = (c_2, \mathbf{C}_2, \langle \xrightarrow{a}_2 \mid a \in \mathbf{A} \rangle)$ be two LTSs with action set $\mathbf{A}$. A relation $\mathcal{R} \subseteq \mathbf{C}_1 \times \mathbf{C}_2$ is a *strong bisimulation* (for $\mathcal{T}_1$ and $\mathcal{T}_2$) iff for every $(c_1', c_2') \in \mathcal{R}$ properties (21) and (22) below hold for every $a \in \mathbf{A}$ (cf. [14, Sect. B.2.1]).

$$\forall c_1'' \in \mathbf{C}_1[\ c_1' \xrightarrow{a}_1 c_1'' \Rightarrow \exists c_2'' \in \mathbf{C}_2[\ c_2' \xrightarrow{a}_2 c_2'' \wedge (c_1'', c_2'') \in \mathcal{R}\ ]]. \tag{21}$$

$$\forall c_2'' \in \mathbf{C}_2[\ c_2' \xrightarrow{a}_2 c_2'' \Rightarrow \exists c_1'' \in \mathbf{C}_1[\ c_1' \xrightarrow{a}_1 c_1'' \wedge (c_1'', c_2'') \in \mathcal{R}\ ]]. \tag{22}$$

When $\mathcal{T}_1 = \mathcal{T}_2$, a strong bisimulation for $\mathcal{T}_1$ and $\mathcal{T}_2$ is called a strong bisimulation *on* $\mathcal{T}_1$. We say $\mathcal{T}_1$ and $\mathcal{T}_2$ are *strongly bisimilar* iff there exists a strong bisimulation $\mathcal{R}$ for $\mathcal{T}_1$ and $\mathcal{T}_2$ such that $(c_1, c_2) \in \mathcal{R}$. We write $\mathcal{T}_1 \sim_{\mathrm{e}} \mathcal{T}_2$ to mean this property, where "e" in $\sim_{\mathrm{e}}$ stands for *events*. ∎

It immediately follows that $\sim_{\mathrm{e}}$ is an equivalence relation. In particular, it is transitive in the sense that the following holds for every LTSs $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$:

$$\mathcal{T}_1 \sim_{\mathrm{e}} \mathcal{T}_2 \wedge \mathcal{T}_2 \sim_{\mathrm{e}} \mathcal{T}_3 \Rightarrow \mathcal{T}_1 \sim_{\mathrm{e}} \mathcal{T}_3. \tag{23}$$

In terms of $\sim_{\mathrm{e}}$, the reduction system of Sect. 3 is related to the one induced from the LTS of Sect. 4.1, as follows.

Lemma 2 (Strong Bisimilarity between the Two Reduction Systems)
For $(D, s) \in \mathcal{L}_{\mathrm{prog}}[\emptyset]$ *and* $\Gamma \in \wp_{\mathrm{fin}}(\mathbf{G})$, *we have*

$$\begin{array}{l}((D, s), \mathcal{L}_{\mathrm{prog}}[\emptyset], \langle \xrightarrow{a} \mid a \in \mathbf{A}\rangle) \\ \sim_{\mathrm{e}} (\langle (D, s), \Gamma\rangle, \tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset] \times \wp_{\mathrm{fin}}(\mathbf{G}), \langle \xrightarrow{a}_{\nu} \mid a \in \mathbf{A}\rangle).\ ∎\end{array} \tag{24}$$

Proof. We define $\langle \overset{\alpha}{\succ\!\!\cdot\!\!\rightarrow} \mid \alpha \in \mathbf{A} \cup \{\iota\}\rangle$ in the same way as $\langle \overset{\alpha}{\succ\!\!\rightarrow} \mid \alpha \in \tilde{\mathbf{A}}_{\nu}\rangle$, except that we use the two rules COM'_{ν} and STR_{ν} given in Remark 1 instead of the three rules OUT_{ν}, IN_{ν} and COM_{ν} in Definition 5. Also, we define $\langle \xrightarrow{a} \mid a \in \mathbf{A}\rangle$ from $\langle \overset{\alpha}{\succ\!\!\cdot\!\!\rightarrow} \mid \alpha \in \mathbf{A} \cup \{\iota\}\rangle$ as we defined $\langle \xrightarrow{a}_{\nu} \mid a \in \mathbf{A}\rangle$ from $\langle \overset{\alpha}{\succ\!\!\rightarrow} \mid \alpha \in \mathbf{A} \cup \{\iota\}\rangle$ (see Definition 6). We can show that

$$\boxed{\begin{array}{l}\textit{The relation } \cong \textit{ is a strong bisimulation on} \\ \qquad (\tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset] \times \wp_{\mathrm{fin}}(\mathbf{G}), \langle \overset{\alpha}{\succ\!\!\cdot\!\!\rightarrow} \mid \alpha \in \mathbf{A} \cup \{\iota\}\rangle) \quad \textit{and on} \\ \qquad (\tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset] \times \wp_{\mathrm{fin}}(\mathbf{G}), \langle \overset{\alpha}{\succ\!\!\rightarrow}_{\nu} \mid \alpha \in \mathbf{A} \cup \{\iota\}\rangle).\end{array}} \tag{25}$$

Let us write $(\langle D, s\rangle, \Gamma) \cong (\langle D', s'\rangle, \Gamma)$ to mean that $\langle D, s\rangle \cong \langle D', s'\rangle$. Then we can prove that for every $\alpha \in \mathbf{A} \cup \{\iota\}$, the two relations $\overset{\alpha}{\succ\!\!\cdot\!\!\rightarrow}$ and $\overset{\alpha}{\succ\!\!\rightarrow}$ coincide up to $\cong$, namely that

$$\begin{array}{l}\forall \alpha \in \mathbf{A} \cup \{\iota\}, \forall \langle D, s\rangle, \langle D', s'\rangle \in \tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset], \forall \Gamma, \Gamma' \in \wp_{\mathrm{fin}}(\mathbf{G})[\\ \quad \textbf{(i)}\ (\langle D, s\rangle, \Gamma) \overset{\alpha}{\succ\!\!\cdot\!\!\rightarrow} (\langle D', s'\rangle, \Gamma') \Rightarrow (\langle D, s\rangle, \Gamma) \overset{\alpha}{\succ\!\!\rightarrow} \cdot \cong (\langle D', s'\rangle, \Gamma'), \\ \quad \textbf{(ii)}\ (\langle D, s\rangle, \Gamma) \overset{\alpha}{\succ\!\!\rightarrow} (\langle D', s'\rangle, \Gamma') \Rightarrow (\langle D, s\rangle, \Gamma) \overset{\alpha}{\succ\!\!\cdot\!\!\rightarrow} \cdot \cong (\langle D', s'\rangle, \Gamma')\ ].\end{array} \tag{26}$$

From this and (25), it immediately follows that $\cong$ is a strong bisimulation for

$$(\tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset] \times \wp_{\mathrm{fin}}(\mathbf{G}), \langle \xrightarrow{a} \mid a \in \mathbf{A}\rangle) \text{ and } (\tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset] \times \wp_{\mathrm{fin}}(\mathbf{G}), \langle \xrightarrow{a}_{\nu} \mid a \in \mathbf{A}\rangle).$$

Therefore, it holds for every $(D, s) \in \mathcal{L}_{\mathrm{prog}}[\emptyset]$ and $\Gamma \in \wp_{\mathrm{fin}}(\mathbf{G})$ that

$$\begin{array}{l}(\langle (D, s), \Gamma\rangle, \tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset] \times \wp_{\mathrm{fin}}(\mathbf{G}), \langle \xrightarrow{a} \mid a \in \mathbf{A}\rangle) \\ \sim_{\mathrm{e}} (\langle (D, s), \Gamma\rangle, \tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset] \times \wp_{\mathrm{fin}}(\mathbf{G}), \langle \xrightarrow{a}_{\nu} \mid a \in \mathbf{A}\rangle).\end{array} \tag{27}$$

Furthermore, we can prove for every $(D, s) \in \mathcal{L}_{\mathrm{prog}}[\emptyset]$ and $\Gamma \in \wp_{\mathrm{fin}}(\mathbf{G})$ that

$$\begin{array}{l}((D,s),\mathcal{L}_{\mathrm{prog}}[\emptyset],\langle \stackrel{a}{\rightarrow} \mid a \in \mathbf{A}\rangle) \\ \sim_{\mathrm{e}} (\langle (D,s),\Gamma\rangle,\tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset] \times \wp_{\mathrm{fin}}(\mathbf{G}),\langle \stackrel{a}{\rightarrow} \mid a \in \mathbf{A}\rangle).\end{array} \tag{28}$$

From (28),(27) and (23) (the transitivity of $\sim_{\mathrm{e}}$), we immediately obtain (24), the claim of this lemma. Proposition (28) can be proved along the lines of the proof of Theorem 1 of [11] (see also [12, 13]). Here we give a brief sketch.

First, we define $\bar{\mathcal{L}}$ in the same way as $\mathcal{L}$ except that each $g \in \mathbf{G}$ may appear in elements of $\bar{\mathcal{L}}$ as a variable representing a gate. Clearly $\mathcal{L} \subseteq \bar{\mathcal{L}}$. (Notice the difference between $\bar{\mathcal{L}}$ and $\tilde{\mathcal{L}}$: Elements of $\mathbf{G}$ appear as variables in elements of $\bar{\mathcal{L}}$, whereas they appear as constants in $\tilde{\mathcal{L}}$.) We define the structural congruence on $\bar{\mathcal{L}}$ in the same way as in Sect. 3.1, except that the set of gate variables used in $\bar{\mathcal{L}}$ is larger than $\mathcal{V}_{\mathrm{gate}}$ used in $\mathcal{L}$. We use the same symbol $\stackrel{\cdot}{\equiv}$ to denote the structural congruence on $\bar{\mathcal{L}}$. Let $(D,s) \in \mathcal{L}_{\mathrm{prog}}[\emptyset]$ and $\Gamma \in \wp_{\mathrm{fin}}(\mathbf{G})$. We let $\mathcal{T}_1$ and $\mathcal{T}_2$ be the left-hand and right-hand sides of (28), respectively. We define $\mathcal{R} \subseteq \mathcal{L}_{\mathrm{prog}} \times (\tilde{\mathcal{L}}_{\mathrm{prog}} \times \mathbf{G})$ by

$$\begin{array}{l}\mathcal{R} = \{\, ((D,s_1),\langle (D,s_2),\Gamma\rangle) \in \mathcal{L}_{\mathrm{prog}} \times (\tilde{\mathcal{L}}_{\mathrm{prog}} \times \mathbf{G}) \mid \\ \qquad (D,s_1) \stackrel{\cdot}{\equiv} (D,(\nu\ \langle g_1,\ldots,g_n\rangle\ s_2))\},\end{array} \tag{29}$$

where $\Gamma = \{g_1,\ldots,g_n\}$ and $\stackrel{\cdot}{\equiv}$ represents the structural congruence on $\bar{\mathcal{L}}$.

Then, we can show that $\mathcal{R}$ is a strong bisimulation for $\mathcal{T}_1$ and $\mathcal{T}_2$. Moreover, we clearly have $((D,s),\langle (D,s),\Gamma\rangle) \in \mathcal{R}$. Thus, by the definition of $\sim_{\mathrm{e}}$, we obtain (28). ■

Lemma 3 (Semantic Equivalence between $\mathcal{M}$ and $\mathcal{O}_\nu$)

$$\forall (D,s) \in \tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset][\ \mathcal{M}[\![(D,s)]\!] = \mathcal{O}_\nu[\![(D,s)]\!]\ ].\ \blacksquare \tag{30}$$

Proof. Let D be an arbitrary element of $\mathcal{D}$. By using Lemma 1, we can show that

$$\{(X,\mathcal{O}_\nu[\![(D,s)]\!]) \mid (X,s) \in D\} \cup \{(\mathcal{X},(\lambda g.\ \mathcal{O}_\nu[\![(D,S[g/x])]\!])) \mid (\mathcal{X},(x,S)) \in D\} \tag{31}$$

is the fixed-point of the higher-order mapping Φ_D defined in (19). Thus we have

$$\begin{cases} \text{(i) } \forall X \in \mathcal{V}_{\mathrm{proc}}[\ \mathcal{O}_\nu[\![(D,X)]\!] = \mathcal{M}[\![(D,X)]\!]\ ], \\ \text{(ii) } \forall \mathcal{X} \in \mathcal{V}^{(1)}_{\mathrm{proc}}, \forall g \in \mathbf{G}[\ \mathcal{O}_\nu[\![(D,(\mathcal{X}\ g))]\!] = \mathcal{M}[\![(D,(\mathcal{X}\ g))]\!]. \end{cases} \tag{32}$$

Thus, by induction on the structure on $s \in \mathcal{L}[\emptyset]$, we obtain (30) for every $s \in \mathcal{L}[\emptyset]$. ■

Definition 11 (Abstraction Function $\mathcal{A}$)

(1) For $\alpha \in \tilde{\mathbf{A}}_\nu$, we define $\stackrel{\alpha}{\rightarrowtail}_{\mathrm{d}} \subseteq (\mathbf{P} \times \wp_{\mathrm{fin}}(\mathbf{G}))^2$ as follows (where 'd' in $\stackrel{\alpha}{\rightarrowtail}_{\mathrm{d}}$ stands for *denotational*):

$$\forall (p,\Gamma),(p',\Gamma')[\ (p,\Gamma) \stackrel{\alpha}{\rightarrowtail}_{\mathrm{d}} (p',\Gamma') \Leftrightarrow p \ni \langle (\Gamma,\alpha,\Gamma'),p'\rangle\ ].$$

(2) For $a \in \mathbf{A}_\nu$, we define $\stackrel{a}{\rightarrow}_{\mathrm{d}} \subseteq (\mathbf{P} \times \wp_{\mathrm{fin}}(\mathbf{G}))^2$ as follows:

$$\forall (p,\Gamma),(p',\Gamma')[\ (p,\Gamma) \stackrel{a}{\rightarrow}_{\mathrm{d}} (p',\Gamma') \Leftrightarrow (p,\Gamma)\,(\stackrel{\iota}{\rightarrowtail}_{\mathrm{d}})^* \stackrel{a}{\rightarrowtail}_{\mathrm{d}} (p',\Gamma')\ ].$$

(3) For $p \in \mathbf{P}$, we define $\mathcal{A}(p)$ to be the LTS $((p,\emptyset),\mathbf{P} \times \wp_{\mathrm{fin}}(\mathbf{G}),\langle \stackrel{a}{\rightarrow}_{\mathrm{d}} \mid a \in \mathbf{A}\rangle)$. ■

The next proposition immediately follows from the definitions of $\mathcal{O}_\nu$ and $\mathcal{A}$.

Proposition 1 *For every* $(D,s) \in \tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset]$, *we have*

$$(\langle (D,s),\emptyset\rangle, \tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset] \times \wp_{\mathrm{fin}}(\mathbf{G}), \langle \xrightarrow{a}_{\nu} \mid a \in \mathbf{A}\rangle) \sim_{\mathrm{e}} \mathcal{A}(\mathcal{O}_{\nu}[\![(D,s)]\!]). \blacksquare \tag{33}$$

Proof. Let $(D,s) \in \tilde{\mathcal{L}}_{\mathrm{prog}}$ and let $\mathcal{T}_1$ and $\mathcal{T}_2$ to be the left-hand and right-hand sides of (33), respectively. We put

$$\mathcal{R} = \{(\langle (D',s'),\Gamma\rangle, \langle \mathcal{O}_{\nu}[\![(D',s')]\!],\Gamma\rangle) \mid (D',s') \in \tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset] \wedge \Gamma \in \wp_{\mathrm{fin}}(\mathbf{G})\}$$

Then, we can check that $\mathcal{R}$ is a strong bisimulation for $\mathcal{T}_1$ and $\mathcal{T}_2$. Clearly, we have $(\langle (D,s),\emptyset\rangle, \langle \mathcal{D}_{\nu}[\![(D,s)]\!],\emptyset\rangle$, where $\langle \mathcal{D}_{\nu}[\![(D,s)]\!],\emptyset\rangle$ is the initial configuration of $\mathcal{T}_2$. Thus, we obtain the desired consequence (33). ■

The correctness $\mathcal{M}$ is stated in terms of $\mathcal{A}$ by the next theorem:

Theorem 1 (Correctness of $\mathcal{M}$ w.r.t. the Reduction-Based Strong Bisimilarity)
For $(D,s) \in \mathcal{L}_{\mathrm{prog}}[\emptyset]$, we have

$$((D,s), \mathcal{L}_{\mathrm{prog}}[\emptyset], \langle \xrightarrow{a} \mid a \in \mathbf{A}\rangle) \sim_{\mathrm{e}} \mathcal{A}(\mathcal{M}[\![(D,s)]\!]). \blacksquare \tag{34}$$

Proof. Let $(D,s) \in \mathcal{L}_{\mathrm{prog}}[\emptyset]$. Then, it follows from Lemma 2 that

$$\begin{aligned} &((D,s), \mathcal{L}_{\mathrm{prog}}[\emptyset], \langle \xrightarrow{a} \mid a \in \mathbf{A}\rangle) \\ &\sim_{\mathrm{e}} (\langle (D,s),\emptyset\rangle, \tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset] \times \wp_{\mathrm{fin}}(\mathbf{G}), \langle \xrightarrow{a}_{\nu} \mid a \in \mathbf{A}\rangle), \end{aligned} \tag{35}$$

where we set Γ in (24) to $\emptyset$. By Proposition 1, we have

$$(\langle (D,s),\emptyset\rangle, \tilde{\mathcal{L}}_{\mathrm{prog}}[\emptyset] \times \wp_{\mathrm{fin}}(\mathbf{G}), \langle \xrightarrow{a}_{\nu} \mid a \in \mathbf{A}\rangle) \sim_{\mathrm{e}} \mathcal{A}(\mathcal{O}_{\nu}[\![(D,s)]\!]). \tag{36}$$

From this, (35) and (23), we obtain

$$((D,s), \mathcal{L}_{\mathrm{prog}}[\emptyset], \langle \xrightarrow{a} \mid a \in \mathbf{A}\rangle) \sim_{\mathrm{e}} \mathcal{A}(\mathcal{O}_{\nu}[\![(D,s)]\!]). \tag{37}$$

By Lemma 3, we have $\mathcal{A}(\mathcal{M}[\![(D,s)]\!]) = \mathcal{A}(\mathcal{O}_{\nu}[\![(D,s)]\!])$. From this and (37), we obtain the desired consequence (34). ■

5.2 Laws for Structural Congruence in $\mathcal{M}$

It can be shown that $\mathcal{M}$ satisfies all laws SC1–SC6 for structural congruence except for SC5. This fact indicates more adequacy of $\mathcal{M}$ as a semantic model for $\mathcal{L}$ than simply being correct with respect to the strong bisimilarity based on the reduction system.

Proposition 2 *Let $D \in \mathcal{D}$, $S, S_1, S_2, S_3 \in \mathcal{L}$, and $\rho \in \mathbf{GEnv}$.*

(1) *If $S_1 \equiv_{\alpha} S_2$, then*

$$\mathcal{M}[\![(D,S_1)]\!](\rho) = \mathcal{M}[\![(D,S_2)]\!](\rho).$$

(2) $\mathcal{M}[\![(D,(\|\ S_1\ (\|\ S_2\ S_3)))]\!](\rho) = \mathcal{M}[\![(D,(\|\ (\|\ S_1\ S_2)\ S_3))]\!](\rho)$.

(3) $\mathcal{M}[\![(D,(\|\ S_1\ S_2))]\!](\rho) = \mathcal{M}[\![(D,(\|\ S_2\ S_1))]\!](\rho)$.

(4) *For $x, y \in \mathcal{V}_{\mathrm{gate}}$,*

$$\mathcal{M}[\![(D,(\nu\ \langle x,y\rangle\ S))]\!](\rho) = \mathcal{M}[\![(D,(\nu\ \langle y,x\rangle\ S))]\!](\rho).$$

(5) *If $x \notin fn(S_2)$, then*

$$\mathcal{M}[\![(D,(\nu\ x\ (\|\ S_1\ S_2)))]\!](\rho) = \mathcal{M}[\![(D,(\|\ (\nu\ x\ S_1)\ S_2))]\!](\rho).$$

(6) *If* $(X, s_X) \in D$, *then*

$$\mathcal{M}[\![(D,X)]\!](\rho) = \mathcal{M}[\![(D,s_X)]\!](\rho).$$

(7) *If* $(\mathcal{X},(x,S)) \in D$, *then for every* $v \in \mathcal{L}_{\text{gate}}$,

$$\mathcal{M}[\![(D,(\mathcal{X}\ v))]\!](\rho) = \mathcal{M}[\![(D,X)]\!](\rho[[\![v]\!](\rho)/x]).\ \blacksquare$$

It is kown that the law SC5 for the structural congruence does not hold in $\mathcal{M}$. It will be necessary to abstract away some unnecessary information in $\mathcal{M}$ in order to make SC5 hold.

6 Concluding Remarks

We conclude this paper with several remarks about related work and directions for further study.

The reduction system of Sect. 4.1 is based on the $\nu\pi$-calculus of [11, 12, 13]. However, we have made a few modifications to the reduction rules of [11, 12, 13]. A major motive for these modifications is our desire to make the metric model $\mathcal{M}$ satisfy the laws for structural congruence. As mentioned above, SC5 dones not hold in $\mathcal{M}$, and it remains for future research to find a natural adaptation of $\mathcal{M}$ such that SC5 holds in it.

It is not known whether the model $\mathcal{M}$ is fully abstract with respect to the observation criterion that we deal with in this paper. It remains for future research to construct such a fully abstract model. A few denotational models for the π-calculus have been proposed and shown to be fully abstract with respect to certain operational criteria [7, 5, 21]. But the operational criteria used in [7, 5, 21] are different from the one used in this paper. It is not known, to our knowledge, whether any of the models in [7, 5] is fully abstract with respect to our operational criterion.

References

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

[2] J.W. de Bakker and E.P. de Vink (1996), *Control Flow Semantics*, The MIT Press.

[3] G. Berry and G. Boudol (1992), The chemical abstract machine, *Theoretical Computer Science, Vol. 96*, pp. 217–248.

[4] R. Cleaveland and M. Hennessy (1987), Priorities in process algebra, *Information and Computation, Vol. 87*, pp. 58–77.

[5] M.P. Fiore, E. Moggi, and D. Sangiorgi (1996), *A Fully-Abstract Model for the π-calculus*, in *Proceedings of Eleventh Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, pp. 43–54.

[6] R. Gorrieri, S. Charchetti, and U. Montanari (1990), A^2CCS: atomic actions for CCS, in *Theoretical Computer Science, Vol. 72*, pp. 203–223.

[7] M. Hennessy (1992), *A model for the π-Calculus*, Computer Science Technical Report, Vol. 91:08, University of Sussex (available via WWW from `http://www.cogs.susx.ac.uk`).

[8] E. Horita (1992), Deriving compositional models for concurrency based on de Bakker-Zucker metric domain from Structured Operational Semantics, *IEICE Transactions on Information and Systems Vol. E75-A, No. 3*, pp. 400–409.

[9] E. Horita (1992), Deriving denotational models for nonuniform concurrency from Structured Operational Semantics, *IPSJ Technical Report, Vol. 92, No. 67, 92-PRG-8*, pp. 1–8.

[10] E. Horita (1996), Deriving failures models for nonuniform concurrency from Structured Operational Semantics, *New Generation Computing*, Vol. 14, No. 3, pp.343–389.

[11] E. Horita and K. Mano (1995), *Nepi: A Network Programming Language Based on the π-Calculus*, ECL Technical Report, Vol. 11933, NTT Communication Science Laboratories.

[12] E. Horita and K. Mano (1995), Nepi: a network programming language based on the π-calculus, *IPSJ SIG Notes, 95-PRO-2*, pp. 161–168 (1995).

[13] E. Horita and K. Mano (1996), Nepi: a network programming language based on the π-calculus, in *Proc. of the 1st International Conference on Coordination Models, Languages and Applications 1996*, Lecture Notes in Computer Science Vol. 1061, Springer, pp. 424–427.

[14] 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, ISO.

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

[16] R. Milner (1991), *The Polyadic π-Calculus: A Tutorial*, Technical Report ECS-LFCS-91-180, LFCS, Department of Computer Science, Univ. of Edinburgh.

[17] R. Milner (1992), Functions as processes, *Mathematical Structure in Computer Science, Vol. 2*, pp. 119–141.

[18] 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.

[19] B. C. Pierce and D. N. Turner (1994), *Concurrent objects in a process calculus*, in Proceedings of International Workshop TPPP'94, Lecture Notes in Computer Science, Vol. 907, pp. 187–215, Springer.

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

[21] I. Stark (1996), A Fully Abstract Domain Model for the π -Calculus, in *Proceedings of Eleventh Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, pp. 36–42.

[22] P. Viry (1996), *π-calculus with explicit substitutions as a real input-output model*, preprint available via WWW from `http://www.di.unipi.it/~viry`.