Available online at www.sciencedirect.com
SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science 138 (2005) 61–77

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Contextual Labelled Semantics for Higher-order Process Calculi

Yongjian Li[1] [,2]

Institute for Software, Chinese Academy of Sciences, Beijing

Abstract

In this paper, we study a contextual labelled transition semantics for Higher-Order process calculi. The labelled transition semantics are relatively clean and simple, and corresponding bisimulation equivalence can be easily formulated based on it. Besides we develop a novel approach to reason about behaviours of a higher-order substituted process $P\{Q/X\}$, based on which we can directly prove a very important result – factorisation theorem. To show the correspondence between our semantics and the well-established ones, we characterize our bisimulation in a version of barbed equivalence.

1 Introduction

There are three major aims in this paper. The first is to study the principle to design LTS semantics in higher-order calculi, which gives not only the computation steps but also the congruence for higher-order processes. The second aim is to originate an approach to explicitly formulate higher-order substitution. We think higher-order substitution is an essential feature which makes higher order calculus different from first-order calculus. There should be a good formulation for behaviors of a higher-order substituted process $P\{R/X\}$. The third aim is to develop new techniques to solve the difficult questions in higher-order calculus such as congruence problems, factorization theorem, and organize the structure of theory with more intuition.

[1] This work is supported by NSF project of China under Grant No.60173020.
[2] Email: lyj238@ios.ac.cn

doi:10.1016/j.entcs.2005.05.005

Labelled semantics achieved great success in early studies of process calculi. Based on labelled semantics, the idea of bisimulation was introduced and formulated by Milner and Park [1][2][3] and has become one of the most important notions in process calculi. When moving to more powerful process calculi with mobility and higher order features, the traditional labelled semantics become hard to work with, and corresponding bisimulation formulations are often much more elaborated, resulting in less satisfactory theories. It seems that two problems made traditional labelled semantics difficult for richer process calculi. The first is passing of local names out of the scope where they are defined. The scope extrusion rule was introduced to cope with this [4], but it also introduced some delicate structure into the labels and complicated the definition of labelled transition relation. The second problem is the meaning of the label of higher order output action in traditional LTS. It only takes into account the object that the process emit, but not the context in which the emitted object is supposed to be used. For example, let us consider an output action $P \xrightarrow{\nu\widetilde{c}\,\overline{a}\langle K\rangle} P'$ in $HO\pi$'s LTS [8], where $\nu\widetilde{c}\,\overline{a}\,\langle K\rangle$ represents the output object and extruded names, and P' represents the residual of P. Although the label contains the complex information, the whole transition rule does not contain any information about the context in which emitted object is supposed to be used.

This definition of traditional higher order output transition will cause the non-naturalness problems in the definition of bisimulation. In more detail, when defining some kind of bisimulation $\mathcal{R}$ of two example processes P, Q, we can not use the traditional clause as below to define the matching of the higher-order output actions of the two processes,

- Whenever $P \xrightarrow{\mu} P'$, then Q' exists s.t. $Q \xrightarrow{\mu} Q'$ and $P'\mathcal{R}Q'$, where μ is a higher-order output action. (1)

Unlike first-order output action, the comparison of labels of higher order output action, for example processes, can not be based on syntactic identity in formulation of bisimulation, that would be too strong for any reasonable semantic equivalence. Thomsen used bisimilarity instead of identity in comparing labels[5], following earlier ideas by Astesiano and Boudol [6][7], but the resulting equivalence is still too strong.

Sangiorgi had some very illustrative examples of the problems in [8], [9]. He pointed out that the separation between the object part (i.e., the process emitted) and the context of a higher-order output prevents a satisfactory treatment of the channels private to the two, and then causes the problems of higher order bisimulation mentioned above. To avoid this separation between object part and context of an output action, he proposed a context bisimulation as below which explicitly takes into account the context in which the emitted object is supposed to be used.

- Whenever $P \xrightarrow{(\nu\widetilde{b})\bar{a}\langle K_1\rangle} P'$, then Q' exists s.t. $Q \xrightarrow{(\nu\widetilde{c})\bar{a}\langle K_2\rangle} Q'$ and for all G with $fn(G) \cap (\widetilde{b} \cup \widetilde{c}) = \emptyset$, $(\nu\widetilde{b})\,(P'|G\{K_1/U\})\,\mathcal{R}\,(\nu\widetilde{c})\,(Q'|G\{K_2/U\})$. (2)

In [11], Sewell introduced the contextual point of view of labelled semantics and pinpointed the connection between labelled semantics and reduction semantics by making explicit the intuition that labelled transitions capture the possible interactions between a term and a surrounding context. Roughly his idea is that labels of transitions from a process P will be contexts that, when put together with P, create an occurrence of a reduction rule. As a test of the idea, he shows that the new definition of labelled transition introduces the same bisimulation for a fragment of CCS. Notice that the key difference between Sewell's contextual transition label and traditional transition label is in that the former comes from the process, but the latter comes from the process's environment.

In this paper we want to show that the same idea can be successfully applied to higher-order process calculi. In our labelled transition semantics, there are five kinds of labels : τ, $a(U).Q$, $m.Q$, $\bar{a}\,\langle K\rangle.0$, $\bar{m}.0$. As usual, $P \xrightarrow{\tau} P'$ represents internal communication as traditional LTS. However $P \xrightarrow{a(U).Q} P'$, $P \xrightarrow{m.Q} P'$, $P \xrightarrow{\bar{a}\langle K\rangle.0} P'$and $P \xrightarrow{\bar{m}.0} P'$ represent that P can respond to the test $a(U).Q$, $m.Q$, $\bar{a}\,\langle K\rangle.0$, $\bar{m}.0$, which are fragments of program from context respectively, and then become P'. For example, the higher-order test label $a(U).Q$ means that the environment can accept an object emitted by P at port a and provide a continuation Q after the interaction, and the object emitted will be used to instantiate the higher-order variable U in Q.

The second aim is to originate an approach to explicitly formulate higher-order substitution. We think higher-order substitution is an essential feature which makes higher order calculus different from first-order calculus. There should be a good formulation for behaviors of a higher-order substituted process $P\{R/X\}$. In fact, we can gain a very intuitive classification on behaviors of process $P\{R/X\}$, which are either solely due to a part of P, or solely due to a copy of R, or due to interactions between a component of P and one copy of R, or due to interactions between two copy of R. Despite the classification having clear intuition, people have never formulated it explicitly in higher order calculus. The difficulty lies in that P is an open process with free variables, which has not ever been dealt with directly. To overcome this shortcoming, we extend our LTS semantics to associate symbolic transition rules for open processes. For example, we will have transitions such as $P \xrightarrow{\alpha} P'$ for open processes, and these transitions describe behaviors which are independent in open variables in P. We will note that $P\{R/X\} \xrightarrow{\alpha} P'\{R/X\}$ if $P \xrightarrow{\alpha} P'$ in our LTS. Besides, we also develop a new technique to explore contexts where open variables are located, which will analyze those behaviors of $P\{R/X\}$ that are solely due to a copy of R. Note that if $P = C[X]$, and C is a static context, and $R \xrightarrow{\alpha} R'$, then $C[X]\{R/X\} \xrightarrow{\alpha} C[R']\{R/X\}$ in our framework.

The third aim is to develop new techniques to solve the difficult questions in higher-order calculus such as congruence problems, factorization theorem, and organize the structure of theory with more intuition. First of all, we place factorisation theorem as a more fundamental result than other authors in different semantics in higher-order calculus. Intuitively, this result tells us that higher-order substitution can be simulated by using trigger substitution and private resource. We directly prove the result just by showing how $P\{R/X\}$ and $\nu k(P\{\uparrow k/X\} | \langle k \Longleftarrow R\rangle)$ simulate each other. The techniques we adopt are closely related with a fine-grain formulation on higher-order substitution. It does not need any use of congruence property anymore, which shows factorisation theorem is a rather independent result in higher-order calculus. Furthermore, the congruence property of $\approx$ w.r.t higher-order substitution is a simple corollary of the factorisation theorem.

The rest of the paper is organized as follows: Section 2 states the syntax of the calculi that will be considered in this paper. Section 3 states the reduction semantics of our language, and the motivation of our work on the following contextual labelled semantics in detail. Section 4 specifies the contextual labelled transition semantics and introduces the corresponding bisimulation in the calculi. Section 5 proves the factorisation theorem and congruence properties of the bisimulation. Section 6 will show the correspondence between our labelled transition semantics and reduction semantics. Section 7 is conclusion and related work.

2 Syntax

The language we explored is similar to the second fragment of higher-order $\pi-$calculi proposed in [8] by Sangiorgi, which satisfies the following restrictions: (1) all arities are unary, (2) each process is finitely describable, (3)only guarded choices are permitted, (4)only processes are allowed to transmit (or processes are the only communicated values). In this calculi replication may be avoided because recursion and replication can be simulated using restriction and parallel composition in our calculi, as shown by Thomsen [5]. But for convenience, we introduce the so-called resource program $\langle k \Leftarrow P\rangle$, which is a server process referenced by k forever and generate a copy of P after being triggered once. Let $\mathcal{FN}$ ($\mathcal{HN}$) be the countable infinite set of first-order (higher-order) names. Then $\overline{\mathcal{FN}} =_{df} \{\overline{m}|m \in \mathcal{FN}\}$, $\overline{\mathcal{HN}} =_{df} \{\overline{a}|a \in \mathcal{HN}\}$, $\mathcal{N} = \mathcal{FN} \cup \mathcal{HN}$, $\overline{\mathcal{N}} = \overline{\mathcal{FN}} \cup \overline{\mathcal{HN}}$. The special symbol τ, which does not occur in $\mathcal{N}$, denotes a silent symbol. We use K, L, P, Q, R to range over processes, a, b, ..., to range over higher-order names, m, n, ... to range over first-order names, x, y,... to range over names, U, V, X, Y to range over variables, l, k to range over $\mathcal{FN} \cup \overline{\mathcal{FN}} \cup \{\tau\}$, α, μ to range over $\mathcal{N} \cup \overline{\mathcal{N}} \cup \{\tau\}$. By convention, if $l \neq \tau$, then $\overline{\overline{l}} = l$.

Definition 2.1 The syntax of our language is as follows:

$$P ::= \sum\{\alpha_i.P_i|i \in I\}|P_1|P_2|\nu aP|X|\,\langle k \Leftarrow P\rangle$$

$$\alpha ::= \bar{a}\,\langle P\rangle\,|a(X)|l$$

As usual, $\sum \varnothing$ is denoted by 0. We write $P\{Q/X\}$ for the capture-avoiding higher-order substitution of Q for the free occurrences of X in P, $fn(P)$($fv(P)$) for the set of free names (variables) occurring in P, $n(P)$($Vars(P)$) for the set of names (variables) occurring in P. We write Pr for the set of all the processes, Pr_c for the set of all closed processes. For convenience, we shall treat processes that are $\alpha-$equivalent as syntactically identical processes, and $P = Q$ means that P and Q are syntactically identical.

Definition 2.2 A context is obtained when the hole [] replaces an occurrences of 0 in a process-term given by the grammar in Definition 2.1. A static context C is defined by the BNF grammar: $C := [] \mid C|P \mid P|C \mid \nu bC$. And for a context C, we define $bn(C)$ by induction:

(1) $bn([]) = \emptyset$

(2) $bn(X) = \emptyset$

(3) $bn(C|P) = bn(C)$

(4) $bn(P|C) = bn(C)$

(5) $bn(\nu bC) = bn(C) \cup \{b\}$

We use C to range over contexts, and use $C[P]$ to denote the process obtained by replacing [] with P in C.

3 Reduction semantics

In this section, we first introduce reduction semantics of our calculi. The reason why we choose reduction semantics as a start lies in three points: (1) The interpretation of reduction semantics is uniformly given by a set of rewrite rules, a set of reduction contexts in which they may be applied, and a structural congruence. So it can be easily formulated in a style similar to that in contextual calculi. (2) As mentioned before, we adopt the contextual point view to work out a new labelled semantics, and essence of the so-called contextual point view is just making explicit the intuition that labelled transitions capture the possible interactions between a term and a surrounding context. So it is helpful to interpret our motivation of the new labeled semantics. (3) If the reduction system is available, the correctness of the new labelled semantics can be shown by proving the correctness between the two semantics.

The structural congruence is as follows:

(i) $(P|Q)\,|R \equiv P|\,(Q|R)\,;\ P|Q \equiv Q|P;\ P|0 \equiv P$

(ii) $\nu x 0 \equiv 0;\ \nu x \nu y\ P \equiv \nu y \nu x\ P;\ (\nu x\ P)\,|Q \equiv \nu x\ (P|Q)\,,$ if $x \notin fn(Q)$.

The reduction relation are the set of the rules as following:

H-COM: $\sum\{a(U).P, ...\}|\sum\{\bar{a}\,\langle K\rangle\,.Q, ...\} \longrightarrow P\{K/U\}|Q$

F-COM: $\sum\{m.P, ...\}|\sum\{\bar{m}.Q, ...\} \longrightarrow P|Q$

R-COM: $\langle m \Leftarrow P\rangle\,|\sum\{\overline{m}.Q, ...\} \longrightarrow P|\,\langle m \Leftarrow P\rangle\,|Q$

PAR: $\dfrac{P \longrightarrow P'}{P|Q \longrightarrow P'|Q}$ RES: $\dfrac{P \longrightarrow P'}{\nu x\ P \longrightarrow \nu x\ P'}$

STRUCT: $\dfrac{Q \equiv P, P \longrightarrow P', P' \equiv Q'}{Q \longrightarrow Q'}$

Example 3.1 Let $R_1 = \nu x \overline{a} < S > .P$, $R_2 = \nu y a(U).Q$, $R = R_1|R_2$, and $x \neq y, x \cap fn(Q) = \emptyset$, obviously, there is one reduction $R \longrightarrow \nu x\,((\nu y Q)\,\{S/U\}|P)$

As shown above, the process R_1 can output an object S, which possibly contains the local name x. If R_1 is surrounded by a parallel context, i.e., we put R_1 in parallel with some environment process X, then there is one reduction in the term $R_1|X$ involving the interaction between R_1 and X if and only if X can perform an input action, i.e., $a(U).Y$ occurs unguarded in X with a unrestricted for some Y. After the reduction occurs, the effect of the object S to X will be represented by instantiating with S some variable U in the continuation part of X. In the above example, R_2 surely can provide an input action at name a, and the continuation of it is $\nu y Q$ after the action occurs. Therefore we can view the input action of X as a test provided by X, by which X can input the object emitted by R_1 at name a and pass a continuation X' to R_1. More formally, we can write the test as $a(U).X'$. For the above example, we can write R_2 can provide a test $a(U).\nu y Q$. An output action of R_1 is just its reaction to such a test. So it is natural for us to represent the meaning of the output action of R_1 by a reaction to an input test provided by its environment. Note that the syntax of the test $a(U).X'$ is extracted from the context representing the environment, and this is just the essence of the contextual point view.

On the other side, we can adopt the contextual view to work out the meaning of an input action of a process. But this procedure is much simpler. Say the process R_2 in the above example, if R_2 is put in parallel with some environment process X, then there is one reduction in the term $R_2|X$ involving the interaction between R_2 and X if and only if X can perform an output action, i.e., $\overline{a}\langle K\rangle .Y$ occurs unguarded in X for some Y. The effect of the interaction to R_2 can be represented by instantiating the continuation of R_2 with the object K. We can view the output action of X as a test provided by X , by which X can output the object K at name a. An input action of R_2 is just its reaction to such a test. So we can represent the meaning of the input action of R_1 by an input test provided by its environment. To uniformly represent tests as a fragment of context, we write such an input test as $\overline{a}\langle K\rangle .0$ instead of $\overline{a}\langle K\rangle$. In the above example, we can write R_1 can provide a test $\overline{a}\langle S\rangle .0$.

Similarly, we can adopt the contextual view to work out the meaning of first-input and first-output actions.

Having introduced the motivation behind the contextual point of view on the meaning of actions of the processes, then we can work out a so-called contextual labelled transition semantics for them. The transition labels in our LTS are *contextual* in the sense that each labelled transtion represents a small program which induces an appropriate reduction.[11] There are five kinds of labels: τ, $a(U).Q$, $\bar{a}\langle R\rangle .0$, $m.Q, \overline{m}.0$. As expected $P \xrightarrow{\tau} P'$ represents internal communication. However $P \xrightarrow{a(U).Q} P'$, $P \xrightarrow{m.Q} P'$, and $P \xrightarrow{\bar{a}\langle R\rangle .0} P'$, $P \xrightarrow{\overline{m}.0} P'$represent that P can respond to the test $a(U).Q$, $m.Q$, $\bar{a}\langle R\rangle .0$, and $\overline{m}.0$ respectively, and then becomes P'. In the following section, we will formally introduce our contextual labelled semantics.

4 Contextual Labelled Semantics

For notation simplicity, we also extend restriction and parallel operation on labels $a(U).Q$ and $m.Q$ in the following definition of transition semantics. If $\alpha = a(U).Q$, if $U \notin fv(P)$, $\alpha|P$ denotes $a(U).(Q|P)$, and $P|\alpha$ denotes $a(U).(P|Q)$; if $x \notin fn(\alpha)$, $\nu x\alpha$ denotes $a(U).\nu xQ$. Let $\alpha = m.Q, \alpha|P$ denotes $m.(Q|P)$, and $P|\alpha$ denotes $m.(P|Q)$;if $x \notin fn(\alpha)$, $\nu x\alpha$ denotes $l.\nu xQ$. We define $\alpha\{R/X\} =_{df} \tau$ if $\alpha = \tau$, $\alpha\{R/X\} =_{df} P\{R/X\}$ if $\alpha = P$ for some P.

The labelled transition relation is given by the rules in Table 1. together with the rules for the auxiliary relation $\downarrow$ used in the side condition for com-

$$\text{H-Input:}\sum\{a(U).P, ...\} \xrightarrow{\bar{a}\langle K\rangle.0} P\{K/U\}$$

$$\text{H-Output:}\sum\{\bar{a}\langle K\rangle.P, ...\} \xrightarrow{a(U).Q} P|Q\{K/U\}$$

$$\text{F-Input:}\sum\{m.P, ...\} \xrightarrow{\bar{m}.0} P$$

$$\text{F-Output:}\sum\{\overline{m}.P, ...\} \xrightarrow{m.Q} P|Q$$

$$\text{H-Coml:}\frac{P \xrightarrow{a(U).Q} R}{P|P' \xrightarrow{\tau} R}(P' \downarrow a(U).Q)$$

$$\text{F-Coml: }\frac{P \xrightarrow{m.Q} R}{P|P' \xrightarrow{\tau} R}(P' \downarrow m.Q)$$

$$\text{Parl:}\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

$$\text{Res:}\frac{P \xrightarrow{\alpha} P'}{\nu c.P \xrightarrow{\alpha} \nu c.P'}\ (c \notin fn(\alpha))$$

$$\text{Resource: } \langle m \Leftarrow P\rangle \xrightarrow{\bar{m}.0} P|\ \langle m \Leftarrow P\rangle$$

$$\tau\text{-Prefix: } \sum\{\tau.P, ...\} \xrightarrow{\tau} P$$

$$\text{F-Input-Pred:}\sum\{m.P, ...\} \downarrow m.P$$

$$\text{H-Input-Pred:}\sum\{a(U).P, ...\} \downarrow a(U).P$$

$$\text{Res-Pred:}\frac{P \downarrow \alpha}{\nu c\ P \downarrow \nu c\ \alpha}(c \notin fn(\alpha))$$

$$\text{Left-Pred:}\frac{P \downarrow \alpha}{P|Q \downarrow \alpha|Q}\ (bv(\alpha) \notin fv(Q))$$

$$\text{Resource: } \langle m \Leftarrow P\rangle \downarrow m.(P|\ \langle m \Leftarrow P\rangle)$$

Table 1
Rules for labelled transitions

munication rule.

The auxiliary predicate $P \downarrow a(U).P'$ says that P can provide the test $a(U).P'$. In this case it is the same as the commitment predicate in [15]. The communication rule $\dfrac{P \xrightarrow{a(U).Q} R}{P|P' \xrightarrow{\tau} R}(P' \downarrow a(U).Q)$ says that if P can react to the test $a(U).Q$ and become R, and P' can provide the test, then put them in parallel the system can do an internal communication and become R. In other words, an internal communication of R is due to a reaction of R_1 to a test provided by R_2, where R_1, R_2 are some subprocesses of R. We can easily prove the following three lemmas.

As usual, let $\Longrightarrow$ be the transitive and reflexive closure of $\xrightarrow{\tau}$, $\overset{\mu}{\Longrightarrow}$ be $\Longrightarrow\xrightarrow{\mu}\Longrightarrow$, and $\overset{\widehat{\mu}}{\Longrightarrow}$ be $\Longrightarrow$ if $\mu = \tau$, and $\overset{\mu}{\Longrightarrow}$ otherwise. $P \Downarrow \alpha$ denotes $P \Longrightarrow\downarrow \alpha$.

With this labelled transition relation the notion of bisimulation can be given in the usual way.

Definition 4.1 A binary relation $\mathcal{R} \subseteq Pr_c\times\ Pr_c$ is a strong/weak simulation if $P\ \mathcal{R}\ Q$ implies that whenever $P \xrightarrow{\mu} P'$, then there exists Q' such that $Q \xrightarrow{\mu} Q'/Q \overset{\widehat{\mu}}{\Longrightarrow} Q'$ and $P'\mathcal{R}\ Q'$, where $fv(\mu) = \varnothing$.

A relation $\mathcal{R}$ is a strong/weak bisimulation if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are strong/weak simulations. We say that P, Q are bisimilar, written $P \sim / \approx Q$ if $P\ \mathcal{R}\ Q$ for some strong/weak bisimulation $\mathcal{R}$.

Definition 4.2 [Bisimulation for open processes] For processes P_1, P_2 with free variables $fv(P_1) \cup fv(P_2) \subseteq \widetilde{X}$, we set $P_1 \sim / \approx P_2$ if for all closed processes $\widetilde{Q}$, it holds that $P_1\{\widetilde{Q}/\widetilde{X}\} \sim / \approx P_2\{\widetilde{Q}/\widetilde{X}\}$.

Lemma 4.3 *Suppose $P \approx Q$, and $P \downarrow a(X).S$, then for any K, there exists S' s.t. $Q \Downarrow a(X).S'$ and $S\{K/X\} \approx S'\{K/X\}$.*

The next lemma says that $\approx$ is preserved under non-object context.

Lemma 4.4 *Let P, Q, R, S be closed processes expressions, if $P \approx Q$, then*

(1) $\nu xP \approx \nu xQ$

(2) $\alpha.P + R \approx \alpha.Q + R$

(3) $P|R \approx Q|R$, $R|P \approx R|Q$

(4) $\langle k \Leftarrow P\rangle \approx \langle k \Leftarrow Q\rangle$

Remark 4.5 In lemma 4.4, proof of (3) desreves special care. Rather than proving that $\{< P|R \approx Q|R > \mid P \approx Q, P, Q, R \in Pr_c\}$ is a bisimulation, here we should prove that $\{(C[P], C[Q]) \mid P \approx Q, P, Q \in Pr_c\}$ is a bisimulation, where C is a static context.

Now we have proved the weak bisimulation is preserved under all operators except object constructor (i.e., $P \approx Q$ implies $\bar{a}\langle P\rangle .S + R \approx \bar{a}\langle Q\rangle .S + R$), which is the most difficult due to the higher-order setting to which our language belongs. To derive this, we must prove congruence property of $\approx$ over higher-order substitution (i.e. $P \approx Q$ implies $P\{R/X\} \approx Q\{R/X\}$). We can follow Sangiorgi's approach to prove this by induction on the structure of P [8][9]. But in this paper, we first prove factorisation theorem independently, then show congruence property of $\approx$ over higher-order substitution is just a corollary of the former.

5 Factorisation theorem and Congruence property of bisimulation w.r.t. Higher-order substitution

For convenience, we use $\uparrow k$ to denote a trigger program, which is a process of the form $\sum \overline{k}.0$. Intuitively, in the process $\nu k(P\{\uparrow k/X\} | \langle k \Leftarrow Q\rangle)$, the process Q represents a "local resource" for P, and k is a "pointer" to the resource, and the only function of the trigger $\uparrow k$ is to activate a copy of Q, and each copy of Q is activated by a trigger $\uparrow k$ when needed. It is natural for us to explore whether $\nu k(P\{\uparrow k/X\} | \langle k \Leftarrow Q\rangle)$ and $P\{Q/X\}$ are equivalent in the sense of weak bisimulation. The Factorisation theorem, which we are to prove, tells us it is so. In order to prove that $\nu k(P\{\uparrow k/X\} | \langle k \Leftarrow Q\rangle)$ is bisimilar with $P\{Q/X\}$, we need four lemmas, Lemma 5.2 and 5.3 analyze transitions which occurs in the process $\nu k(P\{\uparrow k/X\} | \langle k \Leftarrow Q\rangle)$, and Lemma 5.2 analyzes those derived by triggering the private resource, Lemma 5.3 analyzes those which have nothing to do with triggering the resource; Lemma 5.4, 5.5 analyze non-communication-derived and communication-derived transitions which occur in $P\{Q/X\}$. The central technique we use is based on a formulation of the non-communication-derived transition or communication-derived transitions performed by these terms. By analysis on the fine-grained formulation of the these transitions, we can see how the transitions of theirs can be simulated by the terms' counterparts. The detail of the formulation of a higher-order substituted process can be found in [17].

In our semantics, transitions fall into two groups: non-communication derived (or prefix) transitions; communication-derived transitions. For example, $\nu b(\bar{a}\langle 0\rangle .0) \xrightarrow{a(X).X} \nu b(0|0)$ is non-communication derived because it is due to a prefix transition $\bar{a}\langle 0\rangle .0) \xrightarrow{a(X).X} 0|0$; but the transition $\nu b(\bar{a}\langle 0\rangle .0|a(X).X) \xrightarrow{\tau} \nu b(0|0)$ is communication derived because it is due to a communication transition $\bar{a}\langle 0\rangle .0|a(X).X \xrightarrow{\tau} 0|0$.

Definition 5.1 Let $P \xrightarrow{\alpha} P'$ be a transition, if its deriving tree is not involved in a H-Coml, H-Comr, F-Coml, F-Comr rule, then we call it a non-commucation-derived transition; otherwise, we call it a commucation-derived transition.

The following lemma analyze the form of the result process, which is derived by a copy of the resource has been triggered.

Lemma 5.2 *Let $P \in Pr$, $fv(P) \subseteq \{X\}$, $Q \in Pr_c$, and k does not occur in P, Q, if $P\{\uparrow k/X\}| < k \Leftarrow Q > \xrightarrow{\tau} P'$, and the last rule applied to derive this transition is F-Comm , then there exists a context C such that $P = C[X]$, $P' \equiv C[Q]\{\uparrow k/X\}| < k \Leftarrow Q >$.*

The following lemma says that if $P\{\uparrow k/X\}$ perform an action α which has nothing to do with k, then P can perform α.

Lemma 5.3 *If $P\{\uparrow k/X\} \xrightarrow{\alpha} P'$, $fv(P) \subseteq \{X\}, fv(\alpha) = \emptyset$, and k does not occur in P, Q, α, then there exists a program P_0 such that $P' = P_0\{\uparrow k/X\}$, and $P \xrightarrow{\alpha} P_0$, moreover, it holds $P\{R/X\} \xrightarrow{\alpha} P_0\{R/X\}$ for any closed process R.*

The following lemma analyzes how a non-communication-derived transition $P\{R/X\} \xrightarrow{\alpha} A$ can be simulated by $\nu k(P\{\uparrow k/X\}|\,\langle k \Leftarrow R\rangle)$.

Lemma 5.4 *Let $P \in Pr$, $fv(P) \subseteq \{X\}$, if $P\{R/X\} \xrightarrow{\alpha} P'$, and this transition is non-communication-derived, then*

(1) either $P_0\{R/X\} = P'$ for some P_0, and $\nu k(P\{\uparrow k/X\}| < k \Leftarrow R >) \xrightarrow{\alpha} \nu k(P_0\{\uparrow k/X\}| < k \Leftarrow R >)$;

(2) or $P = C[X]$ with $R \xrightarrow{\alpha} R'$, $C[R']\{R/X\} = P'$ for some C, R', and $\nu k(P\{\uparrow k/X\}| < k \Leftarrow R >) \xrightarrow{\tau}\xrightarrow{\alpha} \nu k(C[0|R'| < k \Leftarrow R >]\{\uparrow k/X\})$.

The following Lemma analyzes how communication-derived transition of $P\{R/X\}$ can be simulated by $\nu k(P\{\uparrow k/X\}|\,\langle k \Leftarrow R\rangle)$.

Lemma 5.5 *Let $P \in Pr$, $fv(P) \subseteq \{X\}$, if $P\{R/X\} \xrightarrow{\tau} P'$, and this transition is communication-derived, then there exists a process P_0 such that $P' = P_0\{R/X\}$, and $\nu k(P\{\uparrow k/X\}| < k \Leftarrow R >) \left(\xrightarrow{\tau} \cup \xrightarrow{\tau}\xrightarrow{\tau} \cup \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}\right) \equiv \nu k(P_0\{\uparrow k/X\}| < k \Leftarrow R >)$.*

Before we proceed in our proof the Factorization theorem, we first define a relation $TrigForm$,

Definition 5.6 A binary relation $TrigForm \subset Pr_c \times Pr_c$ is defined as following:

$$TrigForm =_{df} \left\{ \langle P, Q\rangle \;\middle|\; \begin{array}{l} P = P_0\{R/X\}, \text{ and } Q \equiv \nu k(P_0\{\uparrow k/X\}|\,\langle k \Leftarrow R\rangle) \\ \text{for some } k, P_0, R \text{ with } fv(P_0) \subseteq \{X\} \text{ and } R \in Pr_c \end{array} \right\}$$

Lemma 5.7 *Let $P, Q \in Pr$, $TrigForm(P, Q)$ implies $TrigForm(C[P], C[Q])$ for any static context C.*

Combining Lemma 5.2, 5.3, 5.5, and 5.4, we can analyze how $P\{R/X\}$ and $\nu k(P\{\uparrow k/X\}|\,\langle k \Leftarrow R\rangle)$ can simulate each other.

Theorem 5.8 *Suppose $P \in Pr$, $R \in Pr_c$, $fv(P) \subseteq \{X\}$, and k does not occur free in P and R, then $\nu k(P\{\uparrow k/X\}| < k \Leftarrow R >) \approx P\{R/X\}$. Furthermore, it also holds that:*

(1) *whenever* $\nu k(P\{\uparrow k/X\}| < k \Leftarrow R >) \xrightarrow{\alpha} P'$, *there exists* Q' *s.t.* $P\{R/X\} \overset{\widehat{\alpha}}{\Longrightarrow} Q'$ *and* $TrigForm(Q', P')$;

(2) *and vice versa, whenever* $P\{R/X\} \xrightarrow{\alpha} P'$, *there exists* Q' *s.t.* $\nu k(P\{\uparrow k/X\}| < k \Leftarrow R >) \overset{\widehat{\alpha}}{\Longrightarrow} Q'$ *and* $TrigForm(P', Q')$;

Proof. Let
$\Re = \{\langle \nu k(P\{R/X\}, P\{\uparrow k/X\}| \langle k \Leftarrow R\rangle)\rangle$ |if k does not occur free in $P, R\}$. We show $\Re$ is a bisimulation up to $\approx$. Note that $TrigForm(P', Q')$ implies $P'\Re Q'' \equiv Q'$ for some Q''.

I. We show $P\{R/X\}$ can simulate any action performed by $\nu k(P\{\uparrow k/X\} | \langle k \Leftarrow R\rangle)$.

Suppose $\nu k(P\{\uparrow k/X\}| \langle k \Leftarrow R\rangle) \xrightarrow{\alpha} \nu kP'$, then $P\{\uparrow k/X\}| \langle k \Leftarrow R\rangle \xrightarrow{\alpha} P'$, then by case analysis on the last rule to derive $P\{\uparrow k/X\}| \langle k \Leftarrow R\rangle \xrightarrow{\alpha} P'$.

Case 1 The last rule is F-Coml, $\alpha = \tau$, then by Lemma 5.2, there exists a context C such that $P = C[X]$, $P' \equiv C[R]\{\uparrow k/X\}| \langle k \Leftarrow R\rangle$, by $P = C[X]$, then $P\{R/X\} = C[R]\{R/X\}$. So $P\{R/X\} \Longrightarrow P\{R/X\} = C[R]\{R/X\}$, and $TrigForm(C[R]\{R/X\}, \nu kP')$.

Case 2 The last rule is Parl, then there exists a program P' such that $P\{\uparrow k/X\} \xrightarrow{\alpha} P'$, k does not occur in P, Q, α, so by Lemma 5.3, there exists a program P_0 such that $P \xrightarrow{\alpha} P_0$, $P' = P_0\{\uparrow k/X\}$, $P\{R/X\} \xrightarrow{\alpha} P_0\{R/X\}$, and $TrigForm(P_0\{R/X\}, \nu k(P'| \langle k \Leftarrow R\rangle))$.

II. We show $\nu k(P\{\uparrow k/X\}| \langle k \Leftarrow R\rangle)$ can simulate any action performed by $P\{R/X\}$.

Suppose $P\{R/X\} \xrightarrow{\alpha} P'$, then

Case 1 This transition is communication-derived, by Lemma 5.5, there exists a process P_0 such that $P' = P_0\{R/X\}$, and $\nu k(P\{\uparrow k/X\}| \langle k \Leftarrow R\rangle)(\xrightarrow{\tau} \cup \xrightarrow{\tau}\xrightarrow{\tau} \cup \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}) = Q' \equiv \nu k(P_0\{\uparrow k/X\}| \langle k \Leftarrow R\rangle)$ for some Q', and $TrigForm(P', Q')$;

Case 2 This transition is non-communication-derived, by Lemma 5.4, then there are 2 cases:

(2-1) either $P_0\{R/X\} = P'$ for some P_0, and $\nu k(P\{\uparrow k/X\}| \langle k \Leftarrow R\rangle) \xrightarrow{\alpha} \nu k(P_0\{\uparrow k/X\}| \langle k \Leftarrow R\rangle)$, and $TrigForm(P_0\{R/X\}, \nu k(P_0\{\uparrow k/X\}| \langle k \Leftarrow R\rangle))$;

(2-2) or $P = C[X]$ with $R \xrightarrow{\alpha} R'$, $C[R']\{R/X\} = P'$ for some C, R', and

$\nu k(P\{\uparrow k/X\}|k \Leftarrow R) \xrightarrow{\tau}\xrightarrow{\alpha} \nu k(C[0]\{\uparrow k/X\}| \langle k \Leftarrow R\rangle |R') \equiv \nu k(C[R']\{\uparrow k/X\}| \langle k \Leftarrow R\rangle)$.

obviously, $TrigForm(C[R']\{R/X\}, \nu k(C[0]\{\uparrow k/X\}| \langle k \Leftarrow R\rangle |R'))$. □

Lemma 5.9 (abs-congruence) *Let P_1, P_2 , $P \in Pr$, $fv\{P\} \subseteq \{X\}, P_1 \approx P_2$ implies $P\{P_1/X\} \approx P\{P_2/X\}$.*

Proof. Combine Lemma 4.4, and 5.8, we have
$P\{P_1/X\} \approx$ (Lemma 5.8)
$\nu k\,(P\{\uparrow k/X\}|\,\langle k \Rightarrow P_1\rangle) \approx$ (Lemma 4.4)
$\nu k\,(P\{\uparrow k/X\}|\,\langle k \Rightarrow P_2\rangle) \approx$ (Lemma 5.8)
$P\{P_2/X\}$ □

Theorem 5.10 (Congruence of $\approx$ under object operator) *Let P, Q, R, S be processes expressions, if $P \approx Q$, then $\bar{a}\,\langle P\rangle\,.S + R \approx \bar{a}\,\langle Q\rangle\,.S + R$*

Proof. Simply by Lemma 5.9. □

6 Barbed equivalence

Here we will present a notion of barbed equivalence in the style of [13], which is a variation of barbed bisimulation proposed in [8], which is also called "reduction-closed" barbed bisimulation by Sangiorgi in [10]. The interested readers can see [14], [10] for a discussion regarding the two approaches. As usual, $P\downarrow_a$represents that $a(U).Q$, $\bar{a}\,\langle K\rangle\,.Q$ occurs unguarded in P with a unrestricted, i.e., the occurrence is not a subexpression of $c(U').R$, $c\,\langle K'\rangle\,.R$, and not within the scope of νa. Similarly, we can define $\downarrow_m$ where m is a first-order name. Besides, we define $P \Downarrow a(U).S$ to denote $P(\longrightarrow)^* \downarrow a(U).S$.

Definition 6.1 A binary relation $\mathcal{R} \subseteq \mathrm{Pr} \times \mathrm{Pr}$ is a barbed bisimulation if $P_1\mathcal{R}P_2$ implies

(1) for any process $Q \in \mathrm{Pr}$, $P_1|Q\ \mathcal{R}\ P_2|Q$.

(2) for each name a, $P_1 \Downarrow_x$ if and only if $P_2 \Downarrow_x$;

(3) whenever $P_1 \longrightarrow P_1'$, then there exists P_2' such that $P_2\,(\longrightarrow)^*\,P_2'$ and $P_1'\ \mathcal{R}\ P_2'$.

(4) whenever $P_2 \longrightarrow P_2'$, then there exists P_1' such that $P_1\,(\longrightarrow)^*\,P_1'$ and $P_1'\ \mathcal{R}\ P_2'$.

Two processes P and Q are barbed equivalent, written $P \approx^b Q$ if $P\mathcal{R}Q$ for some bisimulation $\mathcal{R}$.

Now we show that $\approx^b$and $\approx$ coincide for the higher-order calculi. We need the following lemmas which are easy to prove.

The following Lemma relates the correspondence between $\longrightarrow$ and $\xrightarrow{\tau}$.

Lemma 6.2 *$P \longrightarrow P'$ if and only if there exist P_0, P_0' such that $P \equiv P_0$, $P_0 \xrightarrow{\tau} P_0'$, $P_0' \equiv P'$.*

Lemma 6.3 *For any process P and name x, it holds that*

(1) if x is a higher-order name, it holds that $P \downarrow_x$if and only if $P \xrightarrow{\alpha} P'$ where $\alpha = x(U).Q$, $\bar{x}\ \langle K \rangle .0$ for some U, K, Q, P'.

(2) if x is a first-order name, then $P \downarrow_x$if and only if $P \xrightarrow{\alpha} P'$ where $\alpha = x.Q$, $\bar{x}\ .0$ for some Q, P'.

Lemma 6.4 *Let z be a name not occur free in P, $a(U).Q$, $\bar{a}\ \langle K \rangle .0$, $m.Q$, $\bar{m}\ .0$ then*

(1) if $P \overset{a(U).Q}{\Longrightarrow} P' / P \overset{m.Q}{\Longrightarrow} P'$ then $P|z\,(X)\,.0|a(U).\overline{z}\,\langle 0 \rangle\,.Q \Longrightarrow\equiv P' / P|z\,(X)\,.0|$ $m.\overline{z}\,\langle 0 \rangle\,.Q \Longrightarrow\equiv P'$;

(2) if $P \overset{\bar{a}\langle K \rangle .0}{\Longrightarrow} P' / P \overset{\bar{m}.0}{\Longrightarrow} P'$ then $P|z\,(X)\,.0|\ \bar{a}\ \langle K \rangle\,.\overline{z}\,\langle 0 \rangle\,.0 \overset{\tau}{\Longrightarrow}\equiv P' / P|\ \bar{m}$ $.z(U).0$ $\overset{\tau}{\Longrightarrow}\equiv P'$;

(3) if $P|z\,(X)\,.0|a(U).\overline{z}\,\langle 0 \rangle\,.Q\,(\longrightarrow)^*\,P''\ / P|z\,(X)\,.0|m.\overline{z}\,\langle 0 \rangle\,.Q\,(\longrightarrow)^*\,P''$, and $P''\ \not\Downarrow_z$ then there exists P' such that $P'' \equiv P'$, and $P \overset{a(U).Q}{\Longrightarrow} P' / P \overset{m.Q}{\Longrightarrow} P'$;

(4) if $P|z\,(X)\,.0|\ \bar{a}\ \langle K \rangle\,.\overline{z}\,\langle 0 \rangle\,.0\,(\longrightarrow)^*\,P'' / P|z\,(X)\,.0|\overline{m}.\overline{z}\,\langle 0 \rangle\,.0\,(\longrightarrow)^*\,P''$ and $P''\ \not\Downarrow_z$ then there exists P' such that $P'' \equiv P'$, and $P \overset{\bar{a}\langle K \rangle .0}{\Longrightarrow} P' / P \overset{\bar{m}.0}{\Longrightarrow} P'$;

Theorem 6.5 *For two processes P and Q, $P \approx^b Q$ if and only if $P \approx Q$.*

Proof. If: We show that $\approx$ is a barbed bisimulation. Suppose $P \approx Q$. By Lemma 4.4, $\approx$ is preserved in parallel operator. If $P \longrightarrow P'$, then by Lemma 6.2, there exists P_0' such that $P \equiv P_0$, $P_0 \xrightarrow{\tau} P_0'$, $P_0' \equiv P'$. Since $\equiv \subseteq \approx$, then $P_0 \approx P \approx Q$, so $Q \Longrightarrow Q_0'$ for some Q_0' with $Q_0' \approx P_0' \equiv P'$. Thus by Lemma 6.2 again, we can find Q' such that $Q\,(\longrightarrow)^*\,Q'$ and $P' \approx Q_0' \equiv Q'$. By Lemma 6.3 and $P \approx Q$, we have that $P \Downarrow_x$if and only if $Q \Downarrow_x$.

Only if: we prove that $\approx^b$ is a bisimulation up to $\approx$. Consider $P \approx^b Q$. Suppose that $P \xrightarrow{a(U).R} P'$. Choose some $z \notin fn(P, a(U).R)$, by definition we have $P|z\,(X)\,.0|a(U).\overline{z}\,\langle 0 \rangle\,.R\ \approx^b\ Q|z\,(X)\,.0|a(U).\overline{z}\,\langle 0 \rangle\,.R$, and by Lemma 6.4(1), $P|z\,(X)\,.0|a(U).\overline{z}\,\langle 0 \rangle\,.R \Longrightarrow P_1 \equiv P'$ for some P_1 with $P_1\ \not\Downarrow_z$, then by Lemma 6.2, $P|z\,(X)\,.0|a(U).\overline{z}\,\langle 0 \rangle\,.R\,(\longrightarrow)^*\,P'$, thus $Q|z\,(X)\,.0|a(U).\overline{z}\,\langle 0 \rangle\,.R\,(\longrightarrow)^*\,Q_1$ for some Q_1 with $Q_1\ \not\Downarrow_z$ and $Q_1 \approx^b P_1$. By Lemma 6.4(3), there exists Q' such that $Q \overset{a(U).R}{\Longrightarrow} Q' \equiv Q_1$, then $P' \equiv\approx^b\equiv Q'$. The case for $P \xrightarrow{\bar{a}\langle b \rangle .0} P'$, $P \xrightarrow{m.R} P'$, $P \xrightarrow{\overline{m}.0} P'$ can be proved in the same way. The case for $P \xrightarrow{\tau} P'$ is simpler.□

7 Conclusion and Related Works

In fact, the work in this paper is the continuation and extension of that in [16]. In [16], we have studied the contextual labelled semantics in the first-order $\pi-$calculus and a simple fragment of higher-order π-calculus separately. In fact, the syntax of that fragment of higher-order $\pi-$calculus is very restricted, and much simpler than that in this paper. In this paper, when we extend the semantic framework to full second-order calculus, the routine proof method for parallel operator fails, and it must be revised as Remark 4.5 said. Besides, factorisation theorem has been derived here, which is not dealt with in [16]. This is a great progress which has been made because factorisation theorem reveals the essence of higher-order calculus.

Sangiorgi did the most pioneering study on the similar topic on higher process calculi [8][9]. In [9] he proposed a traditional late labelled semantics and context bisimulation for higher process calculi. To prove the congruence properties of context bisimulation, he first proved it to be preserved under parallel and replication operator, then he used structural induction on the structure of P to prove congruence property w.r.t higher-order substitution. By using the congruence property and some smart techniques, he prove the distributivity of the so-called implication substitution, then obtain the result of the factorisation theorem. Our approach to achieve the central results is substantially different from his. The key difference lies in the techniques to prove congruence property w.r.t higher-order substitution and factorisation theorem. In his proof, the first question is primary and proved independently, and the proof of the second question heavily uses the result of the first. We emphasize on the fact that the proof of factorisation theorem is not easy even with the result of congruence [9]. But we think of factorisation theorem as a more fundamental result, and it can be proved independently, the congruence property of $\approx$ w.r.t higher-order substitution is a simple corollary of the factorisation theorem. Our main technique to prove the factorisation theorem is on the intuition that how $P\{R/X\}$ and $\nu k(P\{\uparrow k/X\}|\langle k \Leftarrow X\rangle)$ can simulate each other, which is more intuitive and easier to understand. Factorisation theorem is a key result in higher-order calculi, which reveals that higher-order substitution can be simulated by using trigger and private resource. Basing on it, congruence property can be derived easily. Furthermore, it plays a key role when we derive normal bisimulation– which is a simple characterisation of context bisimulation.

Jeffrey and Rathke adopted the same idea and gave a labelled semantics for a fragment of CML with local names [12], and proved that the bisimulation equivalence thus obtained is the same as a version of barbed equivalence. The proof of the first characterization theorem in our paper is similar to their work. But due to the functional nature of CML, they adopted a triggered semantics to prove the congruence properties of the bisimulation equivalence in their work, roughly speaking, they first directly prove the congruence property on trigger semantics, and then they proved the correspondence between the original semantics and the trigger semantics, which yields the congruence property of the original, so their proof is much more difficult than ours. In [18], they directly introduced a trigger semantics as a canonical semantics for a variety of higher-order $\pi - calculus$, and proved the correspondence between bisimulation equivalence in the trigger semantics and barbed equivalence in reduction semantics. The most important result is that they prove the congruence properties of the bisimulation equivalence in this semantics for a language with recursive types. His proof technique is still a result like factorisation theorem, rather than induction techniques on the types, which generally fails for a higher-order language with recursive types. It would be very interesting to understand whether our semantics and formulation for higher-order calculi can be applied to these calculi.

References

[1] R. Milner. *Calculus of Communicating Systems.* LNCS 92, Springer Verlag, 1980.

[2] D. Park. *Concurrency and automata on infinite sequences.* LNCS 104, 1981.

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

[4] R. Milner, J. Parrow, and D. Walker. *A calculus of mobile processes*, (parts I and II). Information and Computation, 100, 1992.

[5] B. Thomsen. *Calculus for Higher Order Communicating Systems.* PHD thesis, Department of Computing, Imperial College, 1990.

[6] E. Astesiano and A. Giovini. *Generalized bisimulation in relational specifications.* LNCS 294, pages 207-266.

[7] G. Boudol. *Towards a lambda calculus for concurrent and Communicating systems.* LNCS 351, pages 149-161.

[8] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms.* Ph.D. thesis, University of Edinburgh, Mayfield Road, Edinburgh, Scotland, 1992.

[9] D. Sangiorgi. *Bisimulation for higher-order process calculi.* Information and Computation,131(2):141-178,1996.

[10] D. Sangiorgi and Davide Walker. *The π − calculus: a* theory of mobile processes. Pages 42-43. Cambridge university press, 2001.

[11] Peter Sewell. *From rewrite rules to bisimulation congruences.* In proceedings of Concur'98, LNCS 1466, pages 269-284, 1998.

[12] Alan Jeffrey and Julian Rathke. *A theory of bisimulation for a fragment of concurrent CML with local names.* In proceedings on Logic in Computer Science, 2000.

[13] Kohei Honda, and Nobuko Yoshida. *On reduction-based process semantics.* Theoretical Computer Science, 152(2):437-486, 1995.

[14] C. Fournet and G. Gonthier. *A hierarchy of equivalence for asynchronous calculi.* LNCS 1443, 1998. In proceedings of International Colloquium on Algorithms, Language and Programming 1998.

[15] R. Milner. *The polyadic $\pi-$Calculus: a tutorial.* The Proceedings of the International Summer School on Logic and Algebra of Specification, 1991.

[16] Liu Xinxin, and Li yongjian. *Bisimulation for Higher-order $\pi-$Calculus.* The Proceedings of the third Asian workshop on Programming Language and systems (Aplas'03). 2002.

[17] Li yongjian, Contextual labelled semantics for Higher-Order process calculi (full version), http://lcs.ios.ac.cn/~lyj238/fgc04.ps.

[18] Alan Jeffrey and Julian Rathke. Contextual equivalence for higher-order pi-calculus revisited. Proc. Mathematical Foundations of Programming Semantics. Elsevier. 2003.