An Extended 7r-calculus *

Ling Luo and Zhenhua DuanI

Institute of Computing Theory and Technology,
Xidian University, Xi'an, P.R. China
lingluo@stu.xidian.edu.cn, zhhduan@mail.xidian.edu.cn

Abstract. To specify time-dependant processes, in this paper, we present an extended 7r-calculus called p-7r in which action prefixes are divided into two kinds, instantaneous and interval ones. First, the syntax and operational semantics of p-7r are formalized. Then, a time-dependant system modeled by p-7r is given to show how the p-7r works.

Keywords: 7r-calculus, process algebra, formalism, verification, extension

1 Introduction

Process algebra is a useful kind of formalism which was proposed for the purpose of specification and verification of concurrent systems. Following the pioneering research of CCS, CSP [1] and ACP [2], lots of variations of process algebras were proposed for different purposes in the past thirty years, such as 7-calculus [3] for mobility, ACSR [4] for modeling real-time systems, Applied 7-calculus for verification of protocols. Due to development of new network computing technologies, e.g., SaaS , SOA , Cloud Computing, 7-calculus is widely used for verification of dynamic behaviors of distributed systems.

However, with 7-calculus, normally, the concurrent model is interleaving and the communication mechanism between processes is based on handshaking; further, the time duration an action consumed is not taken into account. Thus, 7-calculus is not suitable for modeling time-dependant systems. For example, 7-calculus is not easily to model synchronous hardware circuits since different components (like inverters and multiplexors) are connected to a central clock and all components perform a (possibly idle) step on each clock pulse [5]. To solve the problem, we are motivated to extend 7-calculus and to define p-7 so that time-dependant systems can be modeled by p-7. The extension is two-fold: (1) we extend the action prefixes of 7-calculus to contain interval action prefixes; (2) we define the rules related to the interval action prefixes to manipulate and enhance the operational semantics of p-it.

The rest of the paper is organized as follows. In section 2, the syntax and operational semantics of p-7 are defined. Further, as an example, a time-dependent system is modeled in section 3 by means of p-7. Finally, conclusions are drawn in Section 4.

* This research is supported by the NSFC Grant No. 61003078, 61133001 and 60910004, 973 Program Grant No. 2010CB328102 and ISN Lab Grant No. ISN1102001. Corresponding author.

2 Extended 7r-calculus p-7r

2.1 Syntax of p-7r

Let Ai- be an infinite set of names and *PR* a countable set of propositions. The syntax of p-ir processes is given as follows:

it ::= x(y)I $_{Y(Y)}$ ^{r} $_{I}$ /$_{p}$ I *skip*

P ::= ^{0} I ^{7}r•P I Pi + P2 I Pi I P2 I [a 1 = *a2]/' I va P* I ..., a_n)

Where *x, y,* a, al, a2, a_n are names ranging over *N.*

The action prefix it is divided into two categories: the instantaneous action prefix represents either sending *(x(y))* or receiving *(x(y))* a message, or making a silent transition *(r);* the interval action prefix represents either a property, (I_p = {p_i, p_n}(p_i E *PR* for all 1 < i < n)), to be satisfied over a unit interval or idle *(skip).*

A process can be an empty process 0, an action prefix guarded process ir.P, a summation P_1 +P_2 which means the nondeterministic choice, a parallel composition $P_1$1P_2, a match structure [a_l = a_2]P based on name equality, a restriction structure va *P* which behaves as *P* except that the communication on the bound name a is forbidden or a process identifier structure A (ai, …, a_n) with the definition of A *(xi* , x_n) = *PA* and A (a_l , a_n) = {d/Y}PA, in which *a* and x are the vectors of al, a_n and x_1, •••,*xn,* respectively.

The names in a process *P,* denoted by *n(P),* consists of bound and free names, noted by *bn(P)* and *fn(P)* respectively *(n(P) = bn(P) U fn(P)).* Usually, the bound names are names encompassed by () or new names and the others are free names The messages of input prefix, e.g., y in *x(y),* and the new names in restrictions, e.g., a in va *P,* are bound names and the actions and messages of output prefix, e.g., *x* and y in x(y), are free names.

2.2 Semantics of p-ir

In this section, structural operational semantics of p-7r is defined. Further, the structural congruence is given. The execution of p-7r processes consists of two stages: one for instantaneous action prefix and the other for interval action prefix. The execution of an instantaneous action prefix does not take any time and its execution model is in interleaving manner. The communication between concurrent processes with instantaneous action prefixes is based on handshaking when they synchronize through matching channels. Usually, a p-7r process will execute several instantaneous action prefixes before the execution of an interval action prefix. Once all of the executable communications between action prefixes have completed, the remained processes are of the forms *skip.P* or I_p*.P* or 0. Accordingly, the execution of an interval action prefix is triggered. Unlike an instantaneous action prefix, the execution of an interval action prefix is assumed to take an interval with one time unit. To define the operational semantics of p-7r, we first specify the observable action as follows:

70::= (Y) *x(z)* *) I T Ip *skip* ^{7}c::=

x(y) I *x(z) x(z)* I ^{7}

7r_t::= /$_{p}$ I *skip*

Where *x* and y are free names *(f n(7r$_0$), f n*(71$^{-}_{c}$)) and z is a bound name *(bn(7„),bn(ir$_c$)).*

Precisely as in [6], a transition in p-7r is of the form $P \quad Q$ which means P can evolve into Q after performing the action 74,. In p-7r there are six kinds of actions, including the four kinds of actions called 7, as defined in [6] , and the other two kinds of actions called 74 defined above.

There exist several kinds of operational semantics of the 7-calculus [6-8] in the literature. The main difference among them is how to cope with input prefixes. Comparing with others, the *late* operational semantics may be the simplest one for demonstrating the result of our work. In the following, the *late* operational semantics of p-7r is defined based on the one given in [6]. All of the rules, except the last four, are analogous to the *late* operational semantics introduced in [6].

Tau: $\frac{}{T.P \; 4 \; P}$ Out: $\frac{}{\pm(y).P \xrightarrow{(Y)} P \; x(z).P}$ In: $\frac{}{\cdot p \; \{w \; I \; z\}P}$ (w *fn((z)P))*

Sum: $\frac{Pi \xrightarrow{} P \; I}{P1 + P2 \quad I \cdot I}$ Par: $\frac{Pi \quad I}{P_l'IP2}$ *(bn(7c)n* fn(P2) = 0)

Com: $\frac{, \_, \; x(y) \; , \; , \; x(z) \; P2}{/^{31} \; IP2^{4} \; Pi \; IfY/z\}P}$ Close: $\frac{, \; *(Y) \quad I \cdot 1 \rightarrow \quad 2 \cdot 2 \rightarrow \; 2}{Pi \; IP2 \; \cdot 4 \; vy \; (Pi \; 1PD}$

Res: $\frac{PAP'}{vx \; P \; vx \; P'}$ (x ,% n($7r_o$)) Open: $\frac{PAP'}{vy \; P^* \; \{z/y\}P'}$ *(x 0 y, z ,% fn(vy Pi))*

Str: $\frac{P'>Q \quad Q1}{P'>Q}$ Mat: $\frac{PAP'}{[al = a2]/^3 \quad P'}$ (al = a2)

Actt: $\frac{Pt \; '> \; 1i}{Pi \pm P2 \quad P_i}$ Sumt:

$skip \; P2$ Conat: $\frac{P1 \; 1> /^{31} \; P2 \quad 1;)}{Irti \; UTrt2}$ $= 0$

Sumidte: $skip \; P1 \; P2 \; ,2 \rightarrow$ (Irt1 fl it

$Pi \pm P2 \; \cdot) \; Pi/ \pm P2/$ $P1 \; I \; P2$ $PiIPZ$

Since the extension is mainly for interval action prefixes, here we focus on the rules related to the interval action prefixes. The rule Act_t is an axiom for the interval action prefix. The rule Sum_t is intuitionally forward. It is identical to the rule Sum. Further, the rule $Sumtdi_e$ is necessary to ensure the passage of the interval length is deterministic. + means nondeterministic choice between two processes in r-calculus and the difference comes with the action prefix *skip.* If two processes are just idling before one of them is chosen, the choice between them will not be made only by the passage of an unit time interval. That is to say, + is not decided by the action *skip* in p-71^{-}.

The last one Com_t is used for the parallel processes involving interval actions. As the interval actions proceed synchronously, each of the concurrent components will evolve and the properties the whole system will satisfy are the union of the properties satisfied by each of them. As an example, $7rt_1 = 1/^{31}$ p31 and 74_2 = {p2}, then $7ft_1 \; U \; rt_2$ = {Pi, p2, p3}. The auxiliary condition 741 11 742 = 0 is needed for the conflict avoidance. Other rules relative to interval action prefixes, i.e., rules Res, Str and Mat, are unified with those of instantaneous action prefixes.

Definition 1. *A binary relation S on the set of p-Ir processes P is a strong simulation PSQ for P,Q E iff*

1. *if P P', then for some Q', Q Q' and P'SQ'*
2. *if P P' and y n(P, Q), then for some Q', Q Q' and for all w, {w/y}P'S{w/y}Q'*
3. *if P P' and y n(P, Q), then for some Q', Q Q' and P'SQ'*
4. *if P P', then for some Q', Q Q' and P'SQ'*
5. *if P P', then for some Q', Q Q' and P'SQ'*
6. *if P P', then for some Q', Q Q' and P'SQ'*

The relation *S* is a strong bisimulation if both *S* and its inverse are strong simulations. Usually, we use to represent strong bisimilation.

Although systems with different internal structures may have different internal behaviour, however, they may be considered equivalent [3]. Thus, another bisimulation, i.e., weak bisimulation, in which the internal action T is ignored, is more widely used.

Definition 2. *A binary relation S on the set of p-7r processes P is a weak simulation PSQ for P,Q E P iff*

1. *if P P', then for some Q', Q Q' and P'SQ'*
2. *if P P' and y n(P,Q), then for some Q', Q Q' and for all w, {w/y}P'S{w/y}Q'*
3. *if P P' and y n(P, Q), then for some Q', Q Q' and P'SQ'*
4. *if P P', then for some Q', Q = Q' and P'SQ'*
5. *if P P', then for some Q', Q = Q' and P'SQ'*
6. *if P P', then for some Q', Q Q' and P'SQ'*

where the relations = and "> for any 71¹ E *{T(y)*, x(y), T(y), 4, *skip}*, are defined as follows:

1. $P = Q$ means that there is a sequence of zero or more internal actions $P \Rightarrow Q$. Formally, $\rightarrow^* \cdot \xrightarrow{\alpha} \cdot \rightarrow^*$, the transitive reflexive closure of $\xrightarrow{\tau}$.
2. $P \Rightarrow Q$ means $P = Q$. Formally, $\Rightarrow \cdot \xrightarrow{\alpha} \cdot \Rightarrow$

The relation *S* is a weak bisimulation if both *S* and its inverse are weak simulations. Normally, we use to represent weak bisimilation.

Definition 3. *Two process expressions P and Q in p-Ir are structurally congruent, written P Q, if we can transform one into the other by using the equations(in either direction) in the following, as introduced in [3].*

Si: vx P vy {y/x}P if y fn(P)

S 2 : P - F Q Q + P

S 3 : PIO P, PIQ QIP, PI(QIR)

S 4 : vx PIQ Plux Q, if x fn(P)

S 5 : vx0 0, vxyP vyxP

56: [x= 0, if x y

S7: A(cti, an) {d/Y}PA ^{if} A(xi, •••, *xn)* ^{def} *PA*

Definition 4. ***Strong bisimulation up to*** A *binary relation S over the set of p-7r processes P is a strong simulation up to PSQ (P, Q ∈ P), if P P' then there exists Q' such that Q Q' and P' S Q'.*

S is a strong bisimulation up to if both S and its inverse have this property.

Proposition 1. *Let S be a strong bisimulation up to If PSQ, then P N Q.*

Proof. By using Definition 3 and 4, the proposition can easily be proved.

3 Example

In this section, we consider a simple time-dependant mobile system consisting of a server process *R* composed in parallel with a client *C* (the system can be scaled up by increasing the number of clients) and two data service processes D_1 and *D2*. The server can communicate with the client and the two data services by *rc*, rd_1 and rd_2 respectively. The client has a local channel r used to attached to the objective data service (D_1 or D_2), but which one will be chosen is nondeterministic. Each data service process has a local channel d2(i=1,2) representing the relevant data source. The proposition set *PR = {p,* w}, in which *p* represents occupying the printer and w the writer. Further, the system needs to satisfy several time constraints: (1) the usage of the printer and the writer, which are shared resources, will occupy an interval with one time unit; (2) the usage of the same resources is exclusive which means a printer (or writer) can not be occupied by two processes at the same time; (3) the usage of different resources is compatible, e.g., a printer and a writer are able to be used by two processes at the same time; (4) a process will be idle (denoted by *skip* in p-7r) if it does nothing (or occupies nothing). These time constraints can not suitably be expressed by 7r-calculus. Fortunately, the system can be modeled in p-ir as follows.

$$v\ rc,rdi,rd2\ (R(rc,rdi,rd2)\ C\ (rc)\ Di(rdi)\ I\ D2(rd2))$$

$R(rc', ree_l, r\ d'_2)t^f r\ (r_x).(rdc.(r_x).\{p\}\ ..R(r\ ,\ rdi,\ rd2) + rd2(r_x).\{p\}..1i(rc'\ ,\ rdi,\ rd_2))$

$C(rci) \quad = v\ r\ (r(r).r(d).\{w\}.C(rc^i))$

$Di(rd_i) \quad {}^c h^f\ vd2\ (rd_i(rx).(di).skip.Di(rd_i) + skip.Di(rd'_i))(i = 1, 2)$

The idea is that the client sends a service request to the server and the server chooses one data service to link to the client by the abstract channel r_x which will be replaced by the channel r and then the client can communicate with the objective data service by the channel r. Afterwards, the server will occupy the printer for an unit interval ({p}) and the client the writer ({w}). This process can be repeatedly executed. The transitions of the system will be shown according to the operational rules and the structural congruence.

v rc,rdi,rd2 (R(rc,rdi,rd2) C(rc) I Di(rdi) D2(rd2))

v rc, rdi, rd2 (rc(r_z).(rd$_i$ *(T.) •{P}.R(rc, rdi, rd2) + rd2(r.)•{P}.R(rc,rdi,rd2))1 v r (7·7(r).r(d).{w}.C(rc))1v di (rdi(r_x).7(di).skip.Di(rdi) + skip.Di(rdi))I v d2 (rd2(r_x).(d2).skip.D2(rd2) + skip.D2(rd2)))(by* definition and S7)

v rc, rdi, rd2 (v r (rdi(r) • {P}.R(rc, rdi, rd2) + rd2(r)•{P}.R(rc,rdi,rd2)I r(d).{w}.skip.C(rc))1v di (rdi(r.).T;(di).skip.Di(rdi) + skip.Di(rdi))1 v d2 (rd2(r_x).ic (d2).skip.D2(rd2) + skip.D2(rd2)))(by Out,In,Open,Close,Par and Res)

v rc, rdi, rd2 (v r ({p}.R(rc, rdi, rd2) Ir(d).{w}.0 (rc) Iv di (1¯(di).skip.Di(rdi)))1v d2 (rd2(r_x).r(d2.skip.D2(rd2) + skip.D2(rd2)))(by Out,Sum,In,Open,Close,Par and Res)

v rc, rdi, rd2 apl.R(rc,rdi,rd2)1{7.0}•C(re)Iskip.Di(rdi)
d2 (rd2(r_x).7(d2).skip.D2(rd2) + skip.D2(rd2)))(by Out,In,Open,Close,Par,Res,S4)

{p,w} v *rc,rdi,rd2(R(rc,rdi,rd2)1C(rc)Ipi(rdi)* I D2 (rd2))(by Actt,Sumt,Comt and Res)

4 Conclusion

In this paper, we proposed an extended 7r-calculus, p-7r, which contains interval action prefixes. Further, the operational semantics of p-7r is presented. This enables us to model and verify time-dependent systems in a convenient manner. However, we have not examined p-7r techniques for an industrial level example. So a big case study is required for the future research. Also, we need to investigate some verification techniques based on p-7r such as model checking and theorem proving in the near future.

References

1. Hoare , C.A.R.: Communicating Sequential Processes, Prentice-Hall (1985)
2. Bergstra, J.A., J.W.Klop: Algebra of Communicating Processes with Abstraction. J. Theoretical Computer Science. 37,77-121 (1985)
3. Milner, R: Communicating and Mobile Systems: The -calculus. Cambridge University Press (1999)
4. Lee, I., Bremond-Gregoire, P.: A Process Algebraic Approach To The Specification and Analysis of Resource-Bound Real-time Systems. Technical report, MS-CIS-93-08, University of Pennsylvania Department of Computer and Information Science technical (1993)
5. Baier, C., Katoen, J.: Principles of Model Checking. ISBN 978-0-262-02649-9. The MIT Press Cambridge, Massachusetts London, England (2007)
6. Miler, R., Parrow, J., Walker, D.: A Calculus of Mobile Processes. J. Inf. Comput. 100, pp. 1-77 (1992)
7. Miler, R., Parrow, J., Walker, D.: Modal Logics for Mobile Processes. J. Theoret. Comp. 114, pp. 149-171 (1993)
8. Sangiorgi, D., Walker, D.: The -calculus: a Theory of Mobile Processes. Cambridge University Press (2002)