Extending CCS with Multiway Rendezvous

Rosvelter João Coelho da Costa* **Jean-Pierre Courtiat**

LAAS/CNRS – 7, Avenue du Colonel Roche – 31077 Toulouse Cedex France
E-mail: rosvelte@laas.fr, courtiat@laas.fr

Proceedings of the 10th Brazilian Symposium of Computer Networks (SBRC'92). Recife-PE (Brazil), April 1992. Also available as Rapport de Recherche LAAS-CNRS No. 91195, May 1991.

Abstract *Rendezvous is the basic mechanism adopted by many concurrent languages (such that CCS and CSP) for composing independent and cooperating parts of a concurrent system. Nevertheless, CCS, and contrary to CSP, has been defined only with a two-way rendezvous mechanism. This limitation appears as a drawback in many practical applications. Therefore, others CCS-based approaches (such that SCCS, Meije, ...) providing the capability of multiway rendezvous mechanism have been proposed in the literature. However, it is the authors' belief that none of these approaches can be considered as a natural extension of standard CCS. In this paper, we propose a general approach to extend CCS-like languages with a multiway rendezvous which is a direct extension of CCS synchronization mechanism and permits to generalize many (may be all) results obtained for standard CCS.*

Resumo *Rendezvous é o mecanismo básico adotado por muitas liguagens de especificação de sistemas concorrentes (tais como CCS e CSP) para a composição de partes independentes e cooperantes de um sistema concorrente qualquer. Entretanto, para CCS, unicamente o mecanismo de rendezvous simples foi definido. Este fato tem feito com que CCS seja inadequado para muitas aplicações práticas e, por isso, outras linguagens de especificações de sistemas concorrentes baseadas em CCS (tais como SCCS, Meije, ...), mas possuindo a capacidade de especificar rendezvous múltiplo, têm sido proposto na literatura. No entanto, é da opinião dos autores que nenhuma delas parecem ser uma extensão natural de CCS. Neste artigo, é proposto um novo método para capacitar linguagens de especificação de sistemas concorrentes baseadas em CCS com um mecanismo de rendezvous múltiplo.*

Keywords *Formal Description Techniques, Concurrent Languages, Rendezvous, Distributed/parallel Systems, Communication Protocols.*

Introduction

CCS (Calculus of Communicating Systems), introduced some years ago by Milner [11], has become one of the most important formal approaches for the design (specification, analysis, and implementation) of concurrent systems [13, 9]. Following Milner's work, CCS and related approaches such that CSP [3] have been the main sources of inspiration

*Professor of the Dept. of Computer Science and Statistics of the Universidade Federal de Santa Catarina (Florianópolis-Brazil). Partially supported by a research assistantship of MEC/CAPES - Proc. 100/88-11.

of many other algebraic-oriented approaches specifying concurrent systems (for instance, see [12, 8, 2, 4, 1, 10, 15] as a representative sample and further references). The aim of such formal approaches is to provide a mathematical foundation for studying concurrent systems, and trying to avoid, already at the design level, classical errors in concurrent environments (like deadlock, non-termination, ...).

Synchronization in CCS is expressed by means of a two-way rendezvous between CCS agents. Although a two-way rendezvous mechanism may seem to be powerful enough, some important applications may not be described without a multiway rendezvous [5]. In a comparison two-way versus multiway rendezvous, the later particularly permits to specify a concurrent system with a higher level of abstraction, for instance, for capturing user requirements.

To be really useful, multiway rendezvous has to be expressed in a way as simple and intuitive as the two-way rendezvous. Furthermore, any proposal for extending CCS with multiway rendezvous should consider the two-way rendezvous as a special case.

That is the purpose of this short paper which is organized as follows: section 1 introduces a simple synchronization formalism which will be the basis of our proposal; section 2 briefly reminds a few basic concepts of CCS, particularly its two-way rendezvous mechanism; then, section 3, defines MCCS, an extension of CCS with a multiway rendezvous; in this section a formal (operational) semantics of MCCS is provided as well as a simple example illustrating the power of MCCS.

1 A Simple synchronization formalism

In this section we introduce a simple synchronization formalism, which is the basis of our proposal (presented in section 3) for extending CCS with a multiway rendezvous mechanism.

Definition 1. *(Channels and synchronizations)* Let $\mathcal{T}$ be a finite set of *terminals*, ranged over by $p, q, r, \ldots$, with distinct letters denoting distinct terminals, unless otherwise stated. Then, we define a *basic channel* to be any ordered pair of distinct terminals, noted as $\vec{pq}$. Let $\mathcal{BC}$, ranged over by $A, B, \ldots$, denote the set of basic channels[1]. We also define the set $\mathcal{C} = \mathcal{BC} \cup \{1\}$ of *channels*, ranged over by Γ, where $1 \notin \mathcal{BC}$. Then, we define recursively a set $\mathcal{S}$ of *synchronizations* (ranged over by $M, N, \ldots$) as follows:

i. *Basis*: If $M \in \mathcal{C}$, then $M \in \mathcal{S}$
ii. *Induction step*: If $M, N \in \mathcal{S}$, then $M \circ N \in \mathcal{S}$
iii. *Closure*: $M \in \mathcal{S}$ only if it can be derived from the basis (i) by applying operator $\_\circ\_$ a finite number of times. □

The following set of axioms is further assumed:

(a1) $M \circ N = N \circ M$
(a2) $(M \circ N) \circ R = M \circ (N \circ R)$
(a3) $M \circ 1 = M = 1 \circ M$

[1] Note that, $\mathcal{BC} \neq \emptyset$ iff $\#\mathcal{T} \geq 2$.

Note that $(\mathcal{S}, \circ, 1)$ defines a commutative monoid. In the sequel, as usual, we denote $(\mathcal{S}, \circ, 1)$ by $\mathcal{S}$. Moreover, a term in $\mathcal{S}$ will be noted only by its basic channel components; for instance $(\vec{pq} \circ \vec{rs} \circ 1) \circ (\vec{pt} \circ 1 \circ (\vec{rs} \circ \vec{ts}))$ is noted as $\vec{pq} \circ \vec{rs} \circ \vec{pt} \circ \vec{rs} \circ \vec{ts}$.

Now we define two rewrite rules:

(r1) $\vec{pq} \circ \vec{qp} \rightarrow 1$
(r2) $\vec{pq} \circ \vec{qr} \rightarrow \vec{pr}$

These rewrite rules, in fact a family of rewrite rules, introduce in the formalism the notion of "direction" for the channels. Informally, (r1) specifies that two channels with equal "magnitude" and "opposite" direction cancel each other out and (r2) reflects "transitivity" among channels.

One can easily prove that $(\mathcal{S}_{|_{r1}}, \circ, 1)$ is an Abelian group[1]. But even though $(\mathcal{S}_{|_{r1,r2}}, \circ, 1)$ is noetherian, it is not confluent[2]; for instance: $\vec{pq} \circ \vec{qr} \circ \vec{qs}$ can lead either to $\vec{pr} \circ \vec{qs}$ or to $\vec{qr} \circ \vec{ps}$, which are two distinct normal forms. Nevertheless, for our purpose, confluence is not really required, as this synchronization formalism will be applied within a "non-deterministic" framework.

In the sequel, "$M \rhd N$" means that N *can be obtained* from M by applying r1 or r2 a finite number (possibly 0) of times. In the same way, we denote by "$M \underset{r2}{\rhd} N$" the reflexive and transitive closure of the relation induced by r2.

Let us now give some examples that illustrate the intuition behind the formalism.

Example 1.

i. $\vec{pr} \circ \vec{rq} \circ \vec{qp} \rhd 1$ (see figure 1)

ii. $\vec{pr} \circ \vec{rs} \circ \vec{sp} \circ \vec{pq} \circ \vec{qs} \rhd \begin{cases} \vec{pr} \circ \vec{rs} & \text{(see figure 2}(a)\text{)} \\ \vec{pq} \circ \vec{qs} & \text{(see figure 2}(b)\text{)} \end{cases}$

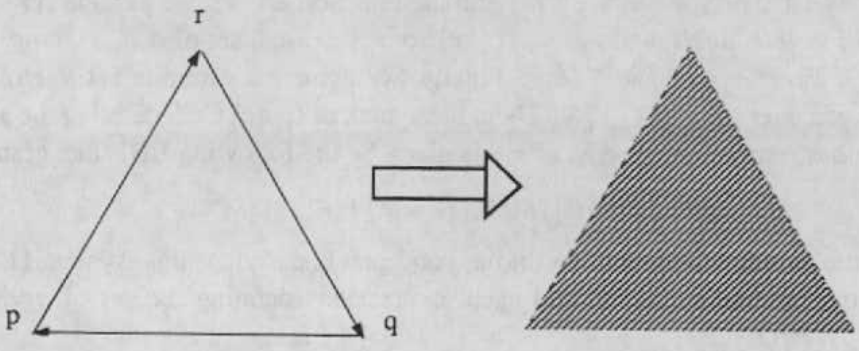

Figure 1

[1] Assuming r1 as an axiom ($\vec{pq} \circ \vec{qp} = 1$) and defining an operation $\overline{M}$, distributive over $\_ \circ \_$, such that $\overline{\vec{pq}} = \vec{qp}$.

[2] We have demonstrated that there are special cases where confluence is guaranteed; for instance, when $\#\mathcal{T} \leq 3$.

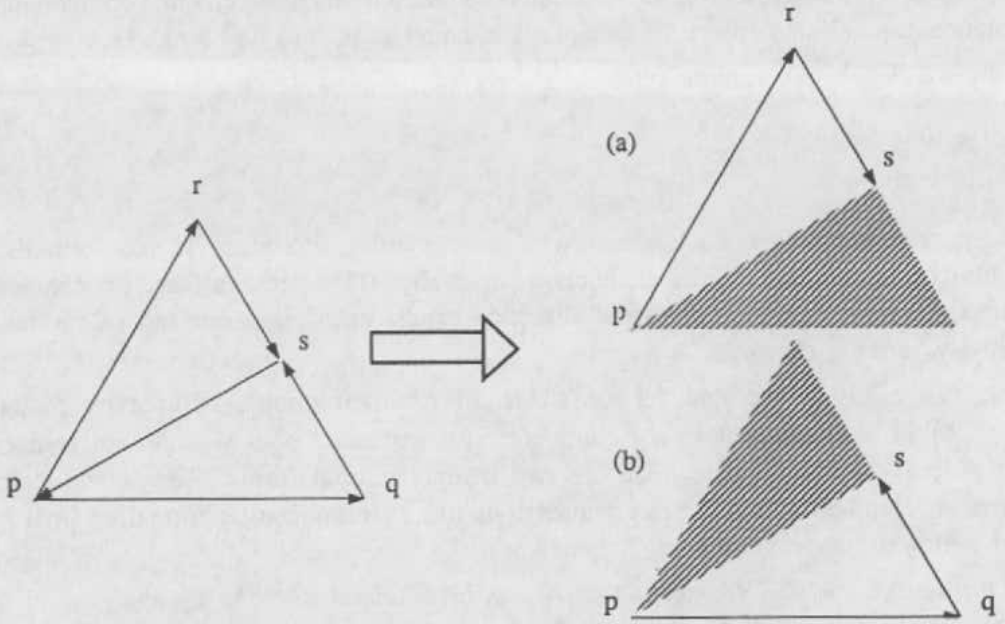

Figure 2

The basic intuition illustrated by these examples is very simple: we characterize multiway rendezvous by the existence of a closed path among channels.

2 CCS – The two-way rendezvous case

In this section, the usual syntax of (pure) CCS is presented. Then, we will survey, briefly and intuitively, some concepts related to the synchronization mechanism of CCS.

For this purpose, we assume a (finite or infinite) set of *labels* $\Delta = \{a, b, c, \ldots\}$, with L standing for an arbitrary finite subset of Δ. Furthermore, we define a set of *co-labels* (complementary labels) $\overline{\Delta} = \{\overline{a}, \overline{b}, \overline{c}, \ldots\}$ and the bijection $\overline{\overline{a}} = a$. We assume $\Lambda = \Delta \cup \overline{\Delta}$ to be the set of *visible labels*, and $\mathrm{Act} = \Lambda \cup \{\tau\}$ to be the usual set of *actions*, ranged over by $\alpha, \beta, \gamma, \ldots$, where $\tau \notin \Lambda$ and $\overline{\tau} \stackrel{def}{=} \tau$. Finally, we assume an infinite set Var of *agent variables*, ranged over by $\ldots, X, Y, Z$. Then, the syntax of (pure) CCS defining the set $\mathcal{E}$ of *agent expressions*, ranged over by $E, F, \ldots$, is given by the following BNF-like grammar[1]:

$$E ::= Nil \mid X \mid \alpha E \mid (E|E') \mid (E + E') \mid (E\backslash L) \mid \mathrm{rec}\, X.E$$

We assume a knowledge of the basic concepts underlying this syntax [12, 13], particularly, the notion of (Var) closed agent expressions defining the set of *agents*, $\mathcal{P}$, ranged over by $P, Q, \ldots$.

In CCS, for any agent expression E, action $\alpha \in \mathrm{Act}$ defines a prefixing operation αE. Intuitively, "synchronization" is obtained assuming the following conversion rule:

$$aE|\overline{a}E' \rightarrow a(E|\overline{a}E') + \tau(E|E') + \overline{a}(aE|E')$$

[1] For the sake of simplicity and without any loss of generality, the renaming operation of CCS is omitted.

Term $\tau(E|E')$ indicates the occurrence of an a-synchronization; and the two other terms indicate that additional a-synchronization can take place in the context where $aE|\overline{a}E'$ will be placed. Note that the restriction operation $E\backslash a$ determines a boundary for a-synchronizations in E, as neither a nor $\overline{a}$ can be derived from $E\backslash a$.

3 Extending CCS with multiway rendezvous

In this section, we introduce MCCS, a simple extension of CCS which permits to express multiway synchronizations.

3.1 The syntax of MCCS

In order to define the syntax of MCCS, another definition is required for the set of actions Act.

Definition 2. (*Actions*) Let $\mathcal{C}$ be a set of channels with respect to some set $\mathcal{T}$ of terminals. Let also $\mathcal{G} = \{a, b, \ldots\}$ be a set of *gates*. Then, an *action* is defined to be any gate $a \in \mathcal{G}$ labeled by a channel $\Gamma \in \mathcal{C}$, in symbols a^{Γ}. In the sequel, Act will denote the set of actions, as defined here, and we will continue to use $\alpha, \beta, \gamma, \ldots$ to range over Act. Also, for the sake of simplicity, whenever there is no danger of confusion, a^1 will be shortly noted as a. □

The syntax of MCCS is almost the same as the one of CCS introduced in the previous section, with one difference the set of actions is now specified as in definition 2; moreover, L appearing in the restriction operation $E\backslash L$, is now a finite subset of $\mathcal{G}$.

3.2 Operational semantics for MCCS

In the sequel, an operational semantics for MCCS in the Plotkin's SOS style [14] is provided, which corresponds merely to an adaptation of the usual transitional semantics given for CCS [13], further taking into account the new synchronization mechanism.

Definition 3.

(i) $\dfrac{}{\alpha E \xrightarrow{\alpha} E}$ (ii) $\dfrac{E \xrightarrow{\alpha} E'}{E+F \xrightarrow{\alpha} E' \qquad F+E \xrightarrow{\alpha} E'}$

(iii) $\dfrac{E \xrightarrow{\alpha} E'}{E|F \xrightarrow{\alpha} E'|F \qquad F|E \xrightarrow{\alpha} F|E'}$ (iv) $\dfrac{E \xrightarrow{a^{\vec{p}}\vec{q}} E' \qquad F \xrightarrow{a^{\vec{q}}\vec{p}} F'}{E|F \xrightarrow{a} E'|F'}$

(v) $\dfrac{E \xrightarrow{a^{\vec{p}}\vec{q}} E' \qquad F \xrightarrow{a^{\vec{q}}\vec{r}} F'}{E|F \xrightarrow{a^{\vec{p}}\vec{r}} E'|F' \qquad F|E \xrightarrow{a^{\vec{p}}\vec{r}} F'|E'}$

(vi) $\dfrac{E \xrightarrow{a^{\Gamma}} E'}{(E\backslash L) \xrightarrow{a^{\Gamma}} (E'\backslash L) \qquad \text{if } a \notin L \vee \Gamma = 1}$ (iv) $\dfrac{[\mathrm{rec}\, X.E/X]E \xrightarrow{\alpha} E'}{\mathrm{rec}\, X.E \xrightarrow{\alpha} E'}$ □

Note that axiom (i) and inference rules (ii), (iii) and (vii) are the standard rules for CCS. Inference rules (iv) and (v) implement rules r1 and r2 of the synchronization formalism presented in section 1. Finally inference rule (vi) is similar to the equivalent rule of CCS, with the difference that internal action τ of CCS is replaced by a (remember that a is a shorthand notation for a^1).

Following this operational semantics, it should be clear that the usual algebraic laws of CCS still hold for MCCS, in particular, the monoidal laws for "+" and "|", the idempotent law for "+", and also the expansion law. All these laws implicitly assume an interpretation of "=" in terms of *strong bisimulation equivalence* [13], which we assume to be sufficiently clear within the context of MCCS ([1]).

For instance, the MCCS expansion law may be formalized in the following way:

Let $P \equiv (P_1|\ldots|P_m)\backslash L, \quad m > 0.$

Let also $\Psi(a^{\vec{p}\vec{q}}) = P_r, \ldots, P_s$ be a non-empty index-ordered sequence such that

- $\{P_r, \ldots, P_s\} \subseteq \left\{ P_j | P_j \in \{P_1, \ldots, P_m\} \wedge P_j \stackrel{a^{A_j}}{\rightarrow} P'_j \right\}$, and
- $A_r \circ \ldots \circ A_s \underset{r2}{\rhd} \vec{p}\vec{q}$

and $\Psi(a) = P_r, \ldots, P_j, \ldots, P_s$ be either a sequence of one element P_j such that $P_j \stackrel{a}{\rightarrow} P'_j$, or an index-ordered sequence of more than one element such that

- $\{P_r, \ldots, P_s\} \subseteq \left\{ P_j | P_j \in \{P_1, \ldots, P_m\} \wedge P_j \stackrel{a^{A_j}}{\rightarrow} P'_j \right\}$, and
- $\exists j$ such that $A_r \circ \ldots \circ A_{j-1} \circ A_{j+1} \circ \ldots \circ A_s \underset{r2}{\rhd} \vec{p}\vec{q} \wedge A_j = \vec{q}\vec{p}$

Then,

$$P = \sum \left[a^{\vec{p}\vec{q}}(P''_1|\ldots|P''_i|\ldots|P''_m)\backslash L : \Psi(a^{\vec{p}\vec{q}}),\ a \notin L,\ P''_i = \begin{cases} P'_i & \text{if } P_i \in \Psi(a^{\vec{p}\vec{q}}) \\ P_i & \text{otherwise} \end{cases} \right]$$
$$+ \sum \left[a(P''_1|\ldots|P''_i|\ldots|P''_m)\backslash L : \Psi(a),\ a \in L,\ P''_i = \begin{cases} P'_i & \text{if } P_i \in \Psi(a) \\ P_i & \text{otherwise} \end{cases} \right]$$

It should be clear that if $\#\mathcal{T} = 2$, only two-way rendezvous is possible, and then we boil down to standard CCS. For instance one may note that if $\mathcal{T} = \{0,1\}$, then we have $\Delta = \left\{ \mathtt{a}^{\vec{01}}, \mathtt{b}^{\vec{01}}, \ldots \right\}$, $\overline{\Delta} = \left\{ \mathtt{a}^{\vec{10}}, \mathtt{b}^{\vec{10}}, \ldots \right\}$, such that $\overline{\mathtt{a}^{\vec{01}}} = \mathtt{a}^{\vec{10}}$ and τ is any action $\mathtt{a}^{\mathbf{1}}$.

Example 2. (*Dining philosophers problem*) With the purpose of illustrating MCCS, we use the well-known "Dining philosophers Problem", originally stated and solved by E.W. Dijkstra [6]. Since then, it has been considered as a classical synchronization problem, not because of its practical importance, but because it is a representative example of a large class of concurrency problems. We are not interested here in solving the problem, but rather on showing how the problem can be stated, at a high level of abstraction, using the multiway rendezvous of MCCS.

[1] The notion of strong bisimulation, and also other (may be all) notions, defined for CCS still hold (or can be straightforward adapted) for MCCS.

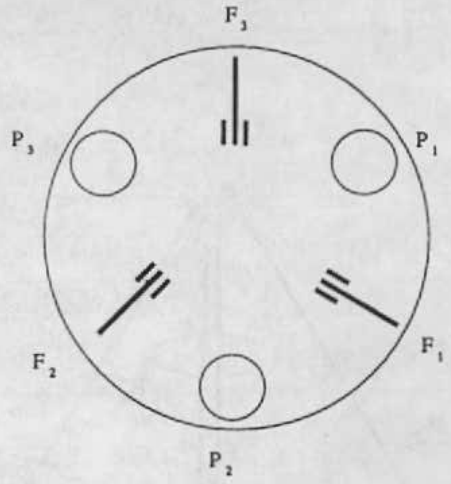

Figure 3 The dining philosophers problem

Let P and F be the agents specifying respectively the behavior of both philosophers and forks. We assume, without loss of generality, that there are three philosophers and three forks[1] (see figure 3). The behavior of each philosopher is given by $P_i = \mu X.u_i^{\vec{pq}}\, e_i\, d^{\vec{st}} X$ where u (picking up two forks), e (eating) and d (putting down the forks) have their usual meaning (for the sake of simplicity, we omit the "thinking activity" in the behavior of the philosophers).

The behavior of the forks may be expressed in the following way:

$$F_1 = \mu X.u_1^{\vec{qr}}\, d^{\vec{tu}} X + u_2^{\vec{rp}}\, d^{\vec{us}} X$$
$$F_2 = \mu X.u_2^{\vec{qr}}\, d^{\vec{tu}} X + u_3^{\vec{rp}}\, d^{\vec{us}} X$$
$$F_3 = \mu X.u_3^{\vec{qr}}\, d^{\vec{tu}} X + u_1^{\vec{rp}}\, d^{\vec{us}} X.$$

The "dining philosophers problem" may then be specified as:

$$D = (P_1|P_2|P_3|F_1|F_2|F_3)\backslash u_1\, u_2\, u_3\, d$$

Figure 4 (where distinct shaded arrows denote distinct gates) describes intuitively the synchronization mechanism involved in the case of three philosophers and forks.

Applying several times the expansion law (and other laws mentioned above) in D, we will obtain the following derivation:

$$\begin{aligned} D = & \\ & u_1\left(e_1\, d^{\vec{st}} P_1|P_2|P_3|d^{\vec{tu}} F_1|F_2|d^{\vec{us}} F_3\right)\backslash u_1\, u_2\, u_3\, d \;+ \\ & u_2\left(P_1|e_2\, d^{\vec{st}} P_2|P_3|d^{\vec{us}} F_1|d^{\vec{tu}} F_2|F_3\right)\backslash u_1\, u_2\, u_3\, d \;+ \\ & u_3\left(P_1|P_2|e_3 d^{\vec{st}} P_3|F_1|d^{\vec{tu}} F_2|d^{\vec{us}} F_3\right)\backslash u_1\, u_2\, u_3\, d \\ & = u_1\, e_1\, d\, D + u_2\, e_2\, d\, D + u_3\, e_3\, d\, D \end{aligned}$$

which clearly describes the deadlock-free (but yet unfair) behavior of the initial specification.

[1] The original example of Dijkstra considers five philosophers and five forks.

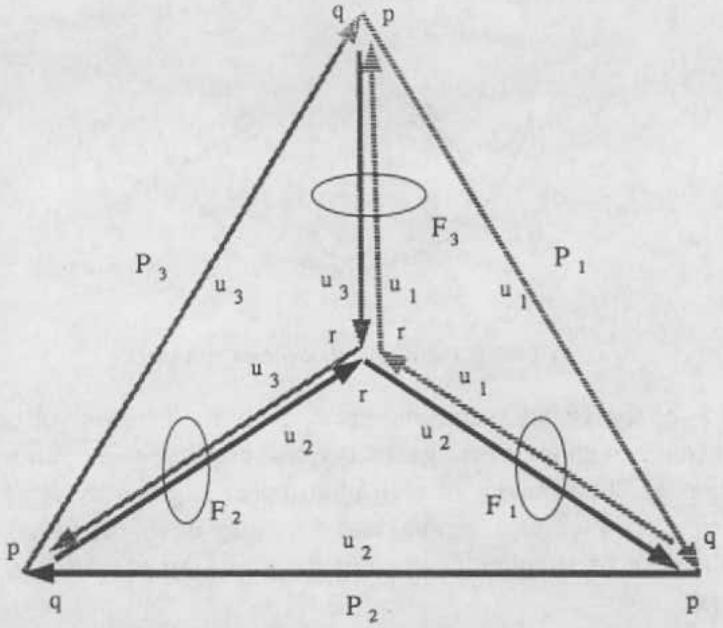

Figure 4 Synchronization problem of the dining philosophers

4 Conclusion

The main result presented in this paper dealt with an original proposal for extending CCS two-way rendezvous mechanism. This proposal is not the first attempt to provide process calculi with multiway rendezvous. Indeed, several solutions have already been presented in the literature. SCCS [12] and Meije [1] are two process calculi close to CCS which have similar synchronization mechanisms allowing multiway rendezvous. Circal [10] is another CCS-like calculus permiting also multiway rendezvous. Although these three calculi are very close to CCS, none of their synchronization mechanisms seem to be a natural generalization of the two-way rendezvous of CCS. On the other hand, CSP [7, 3, 8] and derived approaches like LOTOS [2], have a synchronization mechanism rather different from the previous ones. Nevertheless, as CCS is (even intuitively) different from CSP, the task of comparing the two synchronization approaches is not easy. However, the belief of the authors is that the proposed approach appears to be much more natural and intuitive than these other approaches.

This short paper has presented preliminary results. Of course, the approach developed here is not unique. The synchronization formalism introduced in section 1 can be improved (or modified) in several ways (for instance, "half-duplex" channels could be considered in addition of the "simplex" channels formalized here).

Current work on this topic deals also with the extension of the proposed multiway synchronization with a value-passing mechanism.

References

[1] Austry, L., and Boudol, G. Algèbre de processus et synchronisation. *Theoretical Computer Science 30* (1984), 91–131.

[2] Bolognesi, T., and Brinksma, E. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, 14 (1987), 25–59.

[3] Brookes, S., Hoare, C., and Roscoe, A. A theory of communicating sequential processes. *J. of ACM 31* (1984), 560–599.

[4] Courtiat, J.-P., and Coelho da Costa, R. J. A LOTOS based calculus with true concurrency semantics. In *Proceedings of the IFIP TC6/WG6.1 Fourth International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, FORTE'91* (Sydney, Australia, Nov. 1991), G. A. Rose and K. R. Parker, Eds., North-Holland.

[5] de Saqui-Sannes, P., and Courtiat, J.-P. An extension of the multiway synchronization mechanism concealed by estelle. In *IFIP 11th International Symposium on Protocol Specification, Verification and Testing* (Stockholm, June 1991), North-Holland.

[6] Dijkstra, E. W. Cooperating sequential processes. Tech. Rep. EWD-123, Technological University, Eindhoven, The Netherlands, 1965.

[7] Hoare, C. Communicating sequential processes. *Comm. of the ACM 21*, 8 (1978), 666–677.

[8] Hoare, C. *Communicating Sequential Processes.* Prentice-hall, 1985.

[9] Lopes de Souza, W., and Riso, B. G. Using CCS for protocol specifications by step-wise refinements. In *2nd International Symposium on Interoperable Information Systems* (Japan, 1988).

[10] Milne, G. J. Circal and the representation of communication, concurrency and time. *ACM Trans. on Progr. Lang. and Systems 7* (1985), 270–298.

[11] Milner, R. *A Calculus of Communicating Systems*, vol. 92 of *LNCS*. Springer-Verlag, 1980.

[12] Milner, R. Calculi for synchrony and asynchrony. *Th. Comp. Sci. 25* (1983), 267–310.

[13] Milner, R. *Communication and Concurrency.* C.A.R. Hoare Series Editor. Prentice Hall, 1989.

[14] Plotkin, G. A structural approach to operational semantics. Report Daimi FN-19, Aarhus University (Denmark), 1981.

[15] Sobral, J. B. M., Leão, J. L. S., and Pedrosa, A. C. P. CSP*: Um dialeto de CSP e sua aplicação à especificação de protocolos de comunicação. In *9º Simpósio Brasileiro de Redes de Computadores* (Florianópolis-SC (Brazil), May 1991).