

Formal Modeling of Multi-Agent Systems using the π -Calculus and Epistemic Logic¹

Toinette Rorie and Albert Esterline
NASA ACE
Dept. of Computer Science
North Carolina A&T State University
Greensboro, NC 27411
{rorie,esterlin}@ncat.edu

1. Introduction

Multi-agent systems have become important recently in computer science, especially in artificial intelligence (AI). We allow a broad sense of agent, but require at least that an agent has some measure of autonomy and interacts with other agents via some kind of agent communication language. See [WJ95] for a good discussion of the notion of an agent. We are concerned in this paper with formal modeling of multi-agent systems, with emphasis on communication. We propose for this purpose to use the π -calculus, an extension of the process algebra CCS. Although the literature on the π -calculus refers to agents, the term is used there in the sense of a process in general. It is our contention, however, that viewing agents in the AI sense as agents in the π -calculus sense affords significant formal insight.

One formalism that has been applied to agents in the AI sense is epistemic logic, the logic of knowledge. The success of epistemic logic in computer science in general has come in large part from its ability to handle concepts of knowledge that apply to groups. We maintain that the π -calculus affords a natural yet rigorous means by which groups that are significant to epistemic logic may be identified, encapsulated, structured into hierarchies, and restructured in a principled way.

This paper is organized as follows. Section 2 introduces the π -calculus. Section 3 takes a scenario from the classical paper on agent-oriented programming [Sh93] and translates it into a very simple subset of the π -calculus. Section 4 then shows how more sophisticated features of the π -calculus may be brought into play. Section 5 discusses how the π -calculus may be used to define groups for epistemic logic. Section 6 is the conclusion.

2. The π -Calculus

The π -calculus can be used to model a system that consist of agents which interact with each other, and whose environment is constantly changing [MPW92]. The basic concept behind π -calculus is naming or reference. The names are the primary entities that refer to links or channels, and the processes, sometimes referred to as agents, are the only one other kind of entities. The calculus assumes an infinite set of names where the letters u, v, w, x, y range over the names, and a set of agent identifiers, where each one has an arity represented by an integer ≥ 0 . We let A, B, C, \dots range over agent identifiers. Also, P, Q, R, \dots range over agents or process expressions, of which there are six kinds

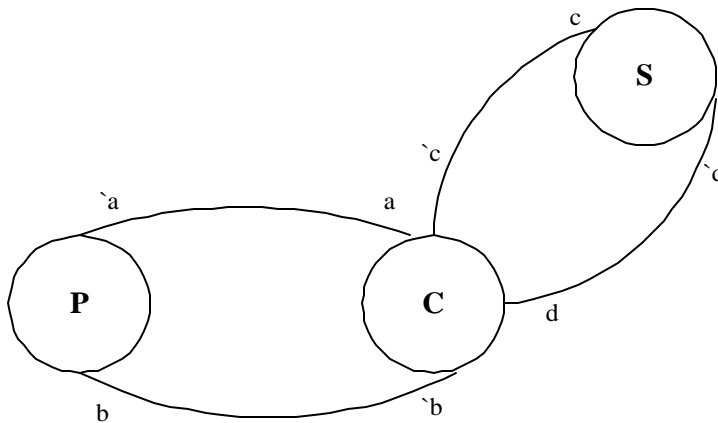
1. A summation $\sum_{i \in I} P_i$ where the set I is a finite index set. The agent behaves like one or another of the P_i . The empty summation is referred to as an inaction, and represented by the symbol $\mathbf{0}$. This is considered to be the agent that can do nothing. The binary summation is written as $P_1 + P_2$.
2. A prefix is presented in the form of $\bar{y}x.P, y(x).P$ or τP . " $\bar{y}x.P$ " is called a negative prefix. \bar{y} can be thought of as an output port of an agent which contains it. " $y(x)$ " is called a positive prefix, where y is the input port of an agent; it binds the variable x . At port y the arbitrary name z is input by $y(x).P$, which behaves like $P\{z/x\}$, where $P\{z/x\}$ indicates the result of substituting z for all free (unbound) occurrences of x in P . τ is the silent action, and τP first performs the silent action and then acts like P .
3. A composition $P_1 | P_2$. This is an agent that consists of P_1 and P_2 that act in parallel. P_1 and P_2 also have the ability to act independently.

¹ The first author has been support by grant NAG 5-4102, "Formal Foundations of Agents," from NASA/GSFC. The second author has been supported by the same grant and by grant NAG 2-1150, "Motion Planning in a Society of Intelligent Mobile Agents," from NASA/ARC.

4. A restriction $(x)P$ is an agent that acts like P , and prohibits actions at ports x and \bar{x} , with the exception of communication that takes place between components of P along the link x . Restriction like positive prefix, binds variables.
5. $[x = y]P$ is referred to as a match, where an agent behaves like P if the names x and y are identical.
6. A defined agent is represented by $A(y_1, \dots, y_n)$. There must be a unique defining equation $A(x_1, \dots, x_n) =^{\text{def}} P$, for any agent identifier A (with arity n) that is used, where x_1, \dots, x_n are distinct names and the only names that may occur free in P . Then $A(y_1, \dots, y_n)$ behaves like $P\{y_i/x_i\}$, for the simultaneous substitution of y_i for all free occurrences of x_i (for $1 \leq i \leq n$) in P .

3. Modeling Agents with the π -Calculus - An Example

We will give an example of an airline scenario taken from [Sh93] that will consist of three agents that interact with each other. The three agents are a passenger, clerk and supervisor. An example of a π -calculus translation will be given, based on the actions that take place between these agents. The example can be represented by the following diagram, where P is the passenger, C is the clerk and S is the supervisor. P sends messages to C along link a , C sends messages to P along link b , C sends messages to S along link c , and S sends messages to C along link d . The output port of a link x is indicated in the diagram as \bar{x} not \bar{x} .



Airline Scenario

March

P to C : Please inform me what flights you have from San Francisco to New York on April 18.

C to P : Flight #354 departs at 08:30, flight #293 departs at 10:00, flight # 441 departs at noon.

P to C : Please book me on #354.

C to P : That is sold out.

P to C : Please book me on # 293.

C to P : That is confirmed, your reservation number is 112358.

P to C : Please book me also on # 441.

C to P : That conflicts with #293, I am not allowed to double book a passenger.

P to C : Please get permission to do so.

C to S : I request permission for the following booking.

S to C : Permission denied.

C to P : Sorry, I cannot get approval.

April 18, at the airport

P to C : My name is P, I have a reservation for flight # 293.

C to P: Here is your boarding pass.

π -Calculus Translation

Each step of the above scenario will be represented by two π -calculus identities, one showing the actions of the agent who sends messages and one showing the actions of the agent who receives the messages. Each identity is of the form

$$X\dot{\alpha}n\dot{\alpha} = sX\dot{\alpha}n+1\dot{\alpha}$$

Here X is P , C , or S . For P , $0 \leq n \leq 12$, for C , $0 \leq n \leq 13$, and for S , $0 \leq n \leq 1$. $P\dot{\alpha}n\dot{\alpha}$ for example, is written as $P3$, and $P0$, $C0$, and $S0$ appear as, respectively, P , C , and S . σ is a sequence of prefixes. At any step, one agent outputs a sequences of messages – constants in negative prefixes – on a single link and the agent at the other end of that link inputs those messages in the order sent by means of variables (x , y , ...) in positive prefixes. The steps show how agent P evolves into agent $P1$, which evolves into $P2$, and so on. Similar evolution happens for agents C and S . We imagine, however, that there are really the same three agents throughout. We could reduce the number of steps by interleaving sequences of positive and negative prefixes. In this translation, the agents have minimum flexibility: one agent expects an input sequence matching in structure the sequence that another outputs. Also, the contents of messages have no effect. We shall see later how to relax these restrictions.

$$P = \dot{a}(4/18). \dot{a}(SF). \dot{a}(NY).P1$$

$$C = a(x).a(y).a(z).C1$$

$$C1 = \dot{b}(354). \dot{b}(8:30). \dot{b}(293). \dot{b}(10:00).C2$$

$$P1 = b(x). b(y). b(z). b(w).P2$$

$$P2 = \dot{a}(354).P3$$

$$C2 = a(x).C3$$

$$C3 = \dot{b}(\text{flight sold out}).C4$$

$$P3 = b(x).P4$$

$$P4 = \dot{a}(293).P5$$

$$C4 = a(x).C5$$

$$C5 = \dot{b}(\text{confirmed}). \dot{b}(112358).C6$$

$$P5 = b(x). b(y).P6$$

$$P6 = \dot{a}(441).P7$$

$$C6 = a(x).C7$$

$$C7 = \dot{b}(\text{conflicts with 293}). \dot{b}(\text{cannot double book}).C8$$

$$P7 = b(x). b(y).P8$$

$$P8 = \dot{a}(\text{get permission to double book}).P9$$

$$C8 = a(x).C9$$

$$C9 = \dot{c}(\text{request permission for double booking}).C10$$

$$S = c(x).S1$$

$$S1 = \dot{d}(\text{permission denied}).0$$

$$C10 = d(x).C11$$

$$C11 = \dot{b}(\text{not approved}).C12$$

$$P9 = b(x).P10$$

April 18 at the airport

$$P11 = \dot{a}(\text{Passengers name}). \dot{a}(\text{reservation for 293}).P12$$

$$C12 = a(x). a(y). C13$$

$$C13 = \bar{b}(\text{boarding pass}). 0$$

$$P12 = b(x). 0$$

The system we have described is represented by

$$P / Q / S$$

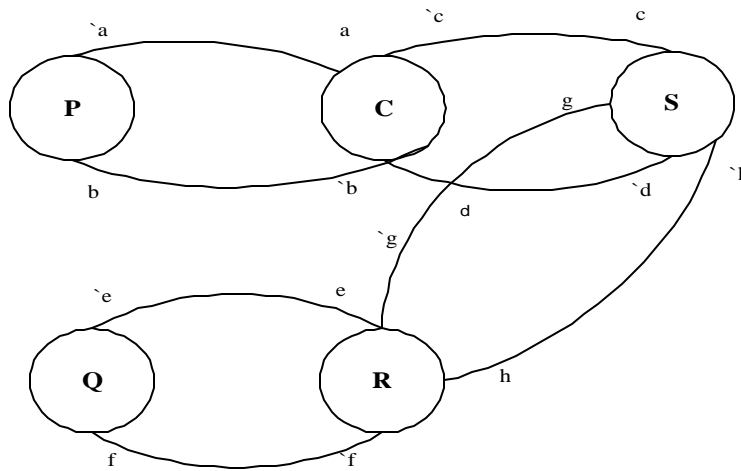
That is, our three agents execute in parallel. Since the P - C system is relatively self-contained, we might restrict the links. For example,

$$SYS1 = (a)(b)(P / C / S)$$

is a system that prohibits actions at ports a , \bar{a} , b , and \bar{b} except for communication between P and C along links a and b . When such a communication occurs, $SYS1$ performs a silent action, τ : the communication is not accessible outside the system.

4. Enhancements to π -Calculus Modeling

We can enhance and modify this specification in several ways. Suppose that there is another passenger Q and another clerk R . Suppose also that Q and R communicate in the same fashion as do P and C . Finally, suppose that R communicates with S just as C communicates with S . Let e be the link on which Q outputs to R , f be the link on which R outputs to Q , g be the link on which R outputs to S , and h be the link on which S outputs to R . We can portray this situation as follows



We have already defined the P - C subsystem, $SYS1$, hiding the a and b links. We now define a similar Q - R subsystem, restricting over links e and f :

$$SYS2 = (e)(f)(Q / R)$$

The only unrestricted ports in $SYS1$ are \bar{c} and d , through which it communicates with S . And the only unrestricted ports of $SYS2$ are \bar{g} and h , through which it communicates with S . It is natural, then, to view the overall system as the parallel composition of $SYS1$, $SYS2$, and S , where now we restrict over the remaining unrestricted links so that the system has no windows to the outside world:

$$(c)(d)(g)(h)(SYS1 / SYS2 / S)$$

Parallel composition and restriction are natural operations when we consider systems of intelligent agents.

Parallel composition allows us to associate intelligent agents into group hierarchies. Restriction allows us to encapsulate these groups so that only certain ports are accessible from outside.

Several other features of the π -calculus could be used to define systems of agents in a way more flexible and responsive than in the above example. We can easily define an agent that repeats its behavior. For example, given

$$P = \bar{a}(5). P,$$

P continually outputs 5 on link a . Mutual recursion is natural in this context. For example, P defined as

$$P = \text{`a}(5).P1$$

$$P1 = \text{`b}(6).P$$

repeatedly outputs 5 on link a and 6 on link b . In the airline scenario above, we could have the system repeat the same behavior again and again if we replace the $\mathbf{0}$ in the definition of $C13$ with C , replace the $\mathbf{0}$ in the definition of $P12$ with P , and replace the $\mathbf{0}$ in the definition of $S1$ with S .

We can endow an agent with memory by including variables that occur in positive prefixes in the subsequent part of the agent expression. For example, agent P defined as

$$P = a(x).P1(x)$$

$$P1(x) = \text{`b}(x).\mathbf{0}$$

inputs a value, to which x is bound, on link a , then it outputs that value on link b and stops. More sophisticated uses of variables could allow one to model beliefs, commitments, and other attitudes an agent might acquire.

More flexibility can be introduced by allowing alternatives, using '+', and by using the match form, $[x=y]$, to select the appropriate alternative. For example, in the above scenario, we have the step

$$P4 = \text{`a}(293).P5$$

$$C4 = a(x).C5$$

representing the passenger's request to be booked on flight #293 and the clerk's confirmation. We could make the clerk flexible by using the variable x to select among several alternatives:

$$C4 = a(x).([x=293]C5 + [x=301]C51 + \frac{1}{4})$$

Here, if x (the flight requested by the passenger) is 293, then the scenario proceeds as before. If, however, x is 301, then $C4$ evolves as $C51$ (whatever that may be). We have indicated any number of additional alternatives. If x has a value not covered by one of these alternatives, then the agent just stops.

The final feature of the π -calculus we mention – changing structure – is perhaps its most powerful feature. Suppose that, in our scenario, the clerk connects the passenger with the supervisor when he requests to double book. Assume that the “ $C9$ ” in the definition of $C8$ has been changed to “ $C91$ ” and the “ $P8$ ” in the definition of $P7$ has been changed to “ $P81$ ”. Then the clerk agent will send the links connecting to the supervisor agent (links c and d) along link b to the passenger agent:

$$C91 = \text{`b}(c).\text{`b}(d).C12$$

The passenger agent will bind, say, variable x to link c and variable y to link d . It will then output its request on link c ($= x$) directly, now, to the supervisor. The passenger agent will now behave like $P91(y)$, who remembers the link d ($= y$) from the supervisor:

$$P81 = b(x).b(y).\text{`x}(request\ permission\ for\ double\ booking).P91(y)$$

The supervisor agent will behave as before – he is not concerned with who is at the other end of the links:

$$S = c(x).S1$$

$$S1 = \text{`d}(permission\ denied).S2$$

The passenger now waits for the message in the link it remembers, y ($= d$):

$$P91(y) = y(z).P10$$

The ability to restructure is of critical importance for a system of intelligent agents. It is generally assumed that the agents with which a given agent will communicate are not determined in advance.

5. Defining Groups for Epistemic Logic

In systems of intelligent agents, the knowledge that can be attributed to agents and groups of agents is particularly important. We follow [] in the syntax and definitions of epistemic logic, the logic of knowledge. Where φ is some proposition, $K_P\mathbf{j}$ means that agent P knows that φ . Where G is a group of agents, $E_G\mathbf{j}$ means that every agent in group G knows that φ (that is, $K_P\mathbf{j}$ holds for all $P \in G$). We can iterate the E_G operator:

$$E_G^i = \begin{cases} E_G\mathbf{j} & \text{if } i=1 \\ E_G E_G^{i-1}\mathbf{j} & \text{if } i>1 \end{cases}$$

Now, $C_G\mathbf{j}$ means that it is common knowledge in group G that φ :

$$C_G \mathbf{j} \text{ iff } E_G \mathbf{j} \wedge E_G^2 \mathbf{j} \wedge E_G^3 \mathbf{j} \wedge \dots$$

Common knowledge is particularly important for showing properties of communications protocols. Finally, $D_G \mathbf{j}$ means that it is distributed knowledge in group G that φ – that is, pooling their knowledge, the agents in G can infer φ .

The π -calculus in particular allows us to identify groups of agents, possibly hierarchically structured, by supplying for agent identifiers defining equations involving parallel composition. The π -calculus also lets us isolate groups using restriction and restructure groups by communicating links.

6. Conclusion

We have indicated how the π -calculus may be used to model communication, memory, selection of alternatives, grouping, and restructuring of groups in multi-agent systems. We have also discussed how the π -calculus can help define group structures that can be exploited by epistemic logic. Considerable work remains, however, before many of the fundamental notions of agent theory can be captured in the π -calculus. Notions, such as intention, commitment, and belief, that involve persisting attitudes require more work with the detailed structures of π -calculus agent expressions. Also, the relation between the semantic structures of epistemic logics (Kripke structures) and the π -calculus remain to be investigated. Indeed, the overlap in the domains of application of epistemic logics and the π -calculus remains to be investigated even though both are applied to communicating systems.

To get a better understanding of how the π -calculus can model a multi-agent system, we are using the Mobility Workbench [VM94]. This tool can be used to manipulate, animate, and analyze concurrent systems described in the π -calculus. Through various notions of bisimulation, it can determine when two agents have the same behavior, and it can determine when an agent satisfies a modal logic expression that specifies behavior at a high level.

References

- [FHMV95] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi, *Reasoning about Knowledge*. Cambridge, MA: The MIT Press, 1995.
- [MPW92] R. Milner, J. Parrow, and D. Walker, “A Calculus of Mobile Processes,” Parts I and II. *Journal of Information and Computation*, Vol. 100, Sept. 1992, pp. 1-77.
- [Sh93] Y. Shoham, “Agent-Oriented Programming.” *Artificial Intelligence*, Vol. 60, No. 1 (1993), pp. 51-92.
- [VM94] B. Victor and F. Moller, *The Mobility Workbench - A Tool for the π -Calculus*. Technical Report DoCS 94/45, Department of Computer Science, Uppsala University, Uppsala, Sweden, Feb. 1994.
- [WJ95] M. Wooldridge and N.R. Jennings, “Intelligent Agents: Theory and Practice.” *The Knowledge Engineering Review*, Vol. 10, No. 2 (1995), pp. 115-152.