Deadlock verification of a DPS coordination strategy and its alternative model in pi-calculus

Pablo D. Robles-Granda, Elham S. Khorasani, Shahram Rahimi* and Norman Carver

Department of Computer Science,
Southern Illinois University,
Carbondale, 62901, USA
E-mail: pdrobles@siu.edu
E-mail: elhams@siu.edu
E-mail: rahimi@siu.edu
E-mail: carver@siu.edu
*Corresponding author

Abstract: A key issue for distributed problem solving (DPS) systems is coordination of the agent's actions, and methods for producing effective coordination strategies remain an active area of research. Because there are not yet approaches that can automatically produce such strategies, some human engineering is often still necessary. As a result, there is a need for a formal tool to support such human engineering. In a previous work (Khorasani et al., 2009), we investigated the use of pi-calculus as a formal language for modelling DPS coordination strategies and showed how such models could be used to evaluate the time performance of a strategy. In this paper, we focus on verification of coordination strategies. More specifically, we utilise the formal semantics of pi-calculus to detect deadlocks in a coordination strategy. We also show how, by imposing certain constraints on the pi-calculus model, one would be able to design a deadlock-free coordination strategy.

Keywords: pi-calculus; distributed problem solving; DPS; coordination strategies; multi-agent systems.

Reference to this paper should be made as follows: Robles-Granda, P.D., Khorasani, E.S., Rahimi, S. and Carver, N. (2012) 'Deadlock verification of a DPS coordination strategy and its alternative model in pi-calculus', *Int. J. Intelligent Information and Database Systems*, Vol. 6, No. 2, pp.154–179.

Biographical notes: Pablo D. Robles-Granda is a Master student in Computer Science at Southern Illinois University Carbondale. He received his BS in Computer Science from Universidad de Cuenca, Ecuador. His research interests include multi-agent systems, machine learning, and data mining.

Elham S. Khorasani is a PhD candidate of Computer Science at Southern Illinois University Carbondale. She received her Masters in Computer Science from Southern Illinois University in 2008. Her main areas of research include fuzzy logic and soft computing, computing with words, and formal languages. She has also served as a Research Assistant for an NSF-funded project on distributed problem solving in multi-agent systems.

Shahram Rahimi is the Undergraduate Programme Director and an Associate Professor of Computer Science at Southern Illinois University Carbondale. He has over 150 peer reviewed publications on multi-agent systems, distributed

computing and soft computing. He is the Editor-in-Chief for *International Journal of Computational Intelligence Theory and Practice* and an Editorial Board Member of a few other journals.

Norman Carver is an Associate Professor and the Graduate Programme Director for Computer Science Department at Southern Illinois University Carbondale. His main research area includes distributed problem solving and multi-agent systems.

1 Introduction

Distributed problem solving (DPS) (Durfee, 2001) systems are generally designed as a whole for their particular application, involve a fixed set of agents, which are inherently cooperative, can easily communicate, and so forth. This is very different from, say, a marketplace MAS where the agents are independently designed, heterogeneous, self-interested, and may come and go. While DPS may sound like it would be much easier to implement than such MAS, DPS systems face issues not faced by other MAS. These issues include severe time constraints, and computational, communication bandwidth, energy, and/or other resource limitations. In such systems, the key issue is coordinating the computational and communication actions of the agents so that high quality solutions are produced rapidly, with acceptable resource usage.

What makes coordination particular is that both problem solving and control are decentralised. Agents make their own decisions about communication and computational actions based on their own views of the state of the MAS. An agent's view is inherently incomplete and uncertain, however. The resource costs of communication mean that agents cannot afford to maintain complete and up-to-date models of the states of all the other agents. The stochastic nature of most DPS application domains means that even if an agent were able to learn the system's global state, it would at best be able to predict only the probability of future states. The bottom line then is that agent coordination strategies must attempt to select the (near) globally optimal computation and communication actions despite incomplete and uncertain knowledge of the global state of the DPS system.

At this point, it is not yet possible to completely automate the design of such strategies for real world DPS applications; generally some human engineering will be required. As a result, there is a need for tools that can support human engineering. In an earlier study (Khorasani et al., 2009), we developed a new tool that allowed human designers to model potential strategies in different levels of abstractions and evaluated the time performance of such strategies. We adopted pi-calculus (Milner, 1992a, 1992b) as a formal specification language to define coordination strategies. Pi-calculus allows a system designer to define tasks at virtually any level of abstraction, specify necessary dependencies, but leave the exact order of sets of operations unspecified if desired. Additionally, its formal semantics supports the development of tools that can analyse other properties of proposed coordination strategies including the performance evaluation and verification of the strategies. We defined appropriate abstractions for modelling a coordination strategy in pi-calculus and showed how such a model could be used for evaluating the time performance of the strategy.

In this paper, we focus on the verification of a coordination strategy and further demonstrate the importance of formal specification of DPS coordination strategies by means of process calculi such as pi-calculus. More specifically, we show how pi-calculus modelling makes the coordination strategy amenable to deadlock detection. We also propose certain restrictions that could be imposed on the pi-calculus model to avoid deadlock situations.

The rest of the paper is organised as follows: Section 2 introduces the related research in designing coordination strategies. Section 3 provides background on pi-calculus. Section 4 briefly reviews our previous study on the pi-calculus modelling of a coordination strategy. Section 5 demonstrate the deadlock detection in a coordination strategy, and Section 6 shows how deadlock could be avoided by imposing certain constraints on the pi-calculus model.

2 Related works

At this point in time, there are not yet techniques that can automatically produce (near) optimal coordination strategies for complex, real-world DPS systems. Two basic classes of techniques for constructing strategies are being pursued:

1 offline, in which strategies are developed prior to system operation as part of system design

2 runtime, in which agents determine appropriate strategies during system execution.

Automatic offline strategy generation is often based on formal techniques. For example, solving a *decentralised Markov decision pro*cess (DEC-MDP) model produces complete or nearly complete action policies for all agents (Becker et al., 2004; Shen et al., 2003). Currently though, such methods remain intractable for all but tiny DPS systems, and use highly simplified cost models. While work continues on approximation techniques, the ultimate practical value of automated formal approaches remains uncertain.

Runtime approaches generally make use of informal or semi-formal techniques to control the scheduling of agent problem-solving and communication actions, and often involve inter-agent communication of control information (e.g., global plans). Arguably the most formal runtime approach to determining coordination strategies is the generalised partial global planning (GPGP) framework (Lesser, 2002; Carver and Lesser, 2003). GPGP relies on a global task structure graph (TAEMS) that describes possible agent activities as well as inter-agent dependencies and effects on system value. A heuristic scheduler uses this structure to schedule agent actions. Producing the TAEMS task structures for GPGP can involve substantial engineering, and GPGP can require a significant amount of computation and communication overhead at runtime. One important consequence of this is that runtime techniques like GPGP are rarely practical for scheduling fine-grained agent actions. This means that a two-level approach is required, where GPGP is used to determine appropriate sub-goals for the agents, and another more efficient control mechanism must be used to select actions to solve these sub-goals. In addition, because GPGP uses a runtime scheduler without formal semantics, this framework cannot serve as the basis for offline analysis of strategy properties.

Because of the limitations of existing methods for constructing coordination strategies, at least some *human engineering* is currently going to be required to develop

coordination strategies for real-world DPS systems. Obviously, it is not practical to expect complete human engineering for systems with large numbers of agents. However, engineering of strategy components for common or critical situations may be able to reduce the computational issues with formal automatic methods, and the lower levels of strategies may need to be engineered for runtime frameworks like GPGP that may not be efficient enough to schedule individual tasks.

3 Pi-calculus background

This section provides a brief overview of the main elements of pi-calculus. The interested reader is referred to (Milner, 1992a, 1992b; Sangiorgi and Walker, 2001) for more details.

In pi-calculus, two main entities are specified, 'names' and 'processes' (or 'agents'). 'Name' is defined as a channel or a value that can be transferred by a channel. We follow the naming rule and syntax in (Milner, 1992a, 1992b), in which u, v, w, x, y, z range over names and A, B, C,... range over agent identifiers.

The syntax of agent is defined as follows:

$$P ::= 0 \left| \sum_{i \in I} \lambda_i.Q_i \right| \overline{y}x.Q \left| \overline{y}(x).Q \right| y(x).Q \left| \tau.Q \right| Q_1 \left| Q_2 \right| Q_1 + Q_2 \left| \nu(x)Q \right| [x = y]Q \left| !Q \right.$$

- 0: Agent P does nothing (null action).
- $\sum_{i \in I} \lambda_i.Q_i$: Is called summation or choice operator. Agent P behaves as $\lambda_i Q_i$ for exactly one i. If $I = \phi$ then P behaves like 0. λ_i denotes any legal action in P (such as τ, $y(x)$, and so forth).
- $\overline{y}x.Q$: Agent P sends free name x out along channel y and then behaves like Q. Name x is said to be free if it is not bound to agent P. In the same way, if x is bound to P (also say private to P), that means x can only used inside of P. To illustrate the difference between the free name and the bound name, consider a system that consists of two agents P and Q. Both agents contain free name x and bound name y. here the free name x is the same in both agents while the bound name y in P is different from the bound name y in Q, although they have the same name.
- $\overline{y}(x).Q$: Agent P sends bound name x out along channel y and then behaves like Q. Both $\overline{y}x.Q$ and $\overline{y}(x).Q$ are called output prefix.
- $y(x).Q$: Is called input prefix. Agent P can receive any name along channel y and then behaves like Q with the received name substituted for x.
- $\tau.Q$: Agent P performs the silent action τ and then behaves like Q. τ can be a computation or an internal communication of an agent.
- $Q_1|Q_2$: Is called composition. Agent P performs Q_1 and Q_2 in parallel. Q_1 and Q_2 may behave independently or they may interact with each other, e.g., if $Q_1 = \tau_1.P_1$ and $Q_2 = \tau_2.P_2$, then Q_1 and Q_2 will behave independently. Otherwise, if $Q_1 = \overline{y}x.P_1$ and $Q_2 = y(z).P_2$, then Q_1 will send x to Q_2 through channel y.

- $Q_1 + Q_2$: Agent P executes either Q_1 or Q_2, but not both.
- $v(x)Q$: Agent P does not change except for that x in Q becomes private to Q. That means any outside communication through channel x will be prohibited.
- $[x = y]Q$: Is called match operator. Agent P behaves like Q if $x = y$, otherwise it behaves like 0.
- $!Q$: Is called replication. Agent P can be thought of as infinite composition of Q, e.g., $Q \mid Q \mid Q \mid \ldots$. Replication is the operator that makes it possible to express infinite behaviours. For example, $!x(z).\overline{y}z.0$ can repeatedly receive name via x and send via y the name that it received.

The semantics of pi-calculus is defined by the deduction rules which formalise the system evolution. The reduction rules for system transitions are listed in Table 1 (Milner, 1992a, 1992b).

Table 1 Pi-calculus reduction rules

TAU-ACT:	$\dfrac{-}{\tau.P \xrightarrow{\tau} P}$	OUTPUT-ACT:	$\dfrac{-}{\overline{x}y.P \xrightarrow{\overline{x}y} P}$
INPUT-ACT:	$\dfrac{-}{x(z).P \xrightarrow{x(w)} P} w \notin fn((z)P)$	MATCH:	$\dfrac{P \xrightarrow{\alpha} P'}{[x = x]P \xrightarrow{\alpha} P'}$
SUM:	$\dfrac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$		
IDE:	$\dfrac{P\{\tilde{y} / \tilde{x}\} \xrightarrow{\alpha} P'}{A(\tilde{y}) \xrightarrow{\alpha} P'} A(\tilde{x}) \overset{def}{=} P$		
PAR:	$\dfrac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} bn(\alpha) \cap fn(Q) = \varphi$		
COM:	$\dfrac{P \xrightarrow{\overline{x}y} P' \quad Q \xrightarrow{x(z)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'\{y / z\}}$		
RES:	$\dfrac{P \xrightarrow{\alpha} P'}{(y)P \xrightarrow{\alpha} (y)P'} y \notin n(\alpha)$		
CLOSE:	$\dfrac{P \xrightarrow{\overline{x}(w)} P' \quad Q \xrightarrow{x(w)} Q'}{P \mid Q \xrightarrow{\tau} (w)(P' \mid Q')}$		
OPEN:	$\dfrac{P \xrightarrow{\overline{x}y} P' \qquad y \neq x}{(y)P \xrightarrow{\overline{x}(w)} P'\{w / y\}} w \notin fn((y)P')$		

4 Pi-calculus modelling of DPS coordination strategies

This section briefly summarises our previous work (Khorasani et al., 2009) on modelling a coordination strategy in pi-calculus. While pi-calculus is extremely powerful and flexible for defining (distributed) processes, the downside of this flexibility is the

complexity of defining strategies in basic pi-calculus. To simplify specifying coordination strategies in pi-calculus, we categorised typical actions in DPS strategies and modelled them in pi-calculus. We also considered different communication scenarios and modelled the communication flow. Using these typical DPS actions, or 'abstractions', which we have created will ease the burden of defining strategies with pi-calculus. For derailed information on pi-calculus actions and reduction rules one can refer to (Milner, 1992a, 1992b). Basic pi-calculus assumes that communication between agents is synchronous. This assumption is not valid in DPS systems where the sending agent can accomplish other tasks, while the message is still in transfer. Thus, asynchronous pi-calculus appears to be more appropriate for modelling DPS coordination strategies. Asynchronous pi-calculus is a sub-calculus of pi-calculus in which message emission is non-blocking (Amadio et al., 1996; Honda and Tokoro, 1991; Sangiorgi and Walker, 2001).

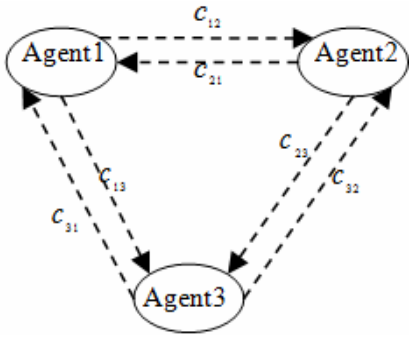
A DPS coordination strategy, *DPSCS*, can be represented in pi-calculus as a number of *agent* processes that run in parallel:

$$DPSCS = Agent_1 \mid Agent_2 \mid \ldots \mid Agent_n$$

Each agent process is defined in terms of a set of *actions*. The actions that agents accomplish can be categorised as follows:

- *Computational:* computational actions are the internal non-communication actions of agents, which are represented by (τ) in pi-calculus.
- *Communication:* the input/output actions in pi-calculus can be used to model communication in different abstraction levels, from the physical communication channels (Ethernet, WiFi) to logical connection between agents. The physical model depends on the agents' network architecture, so the logical connection model is used in this paper to show the communication flow between the agents. Each pair of agents, $Agent_i$ and $Agent_j$, share two free names: c_{ij} and c_{ji}, for communicating with each other: one for sending, and the other one for receiving messages. Figure 1 shows the communication model for a DPS of three agents.

Figure 1 The communication model for DPS of three agents



Communication actions are modelled by simple output and input actions in pi-calculus. $Agent_i$ uses free name c_{ij} to send message m_{ij} to $Agent_j$:

$$Agent_i = \overline{c}_{ij} m_{ij} \,|\, P$$
$$Agent_j = c_{ij}\left(m_{ij}\right) \,|\, Q$$

(P and Q represent the rest of the actions of $Agent_i$ and $Agent_j$, respectively)

The communication between agents may also take place upon a request, when the receiver agent requests information from the sender. In this case, the receiver agent first needs to initiate a new communication session with the sender agent by sending a bound name to it. This is called *scope extrusion* in pi-calculus (Milner, 1992a), because the scope of a *bound name* is extruded to the sender agent. This bound name can be used by the receiver to send request and receive reply back from the sender agent. After receiving a request, the sender agent can decide whether to answer to this request or not, based on some internal decision parameters. The above process can be modelled as follows:

$$Agent_i = \upsilon(d)\left(c_{ji}(x).x(y).\tau_{req}.[d = True]\overline{x}p\right)$$
$$Agent_j = \overline{c}_{ji}(r) \,|\, \overline{r}req \,|\, r(rep).\tau_{rep}$$

(r is a bound name used for establishing a communication session. τ_{req}, τ_{rep} are the computational actions for processing the request and reply, respectively. d is a bound name indicating the decision variable for replying to the request. d can hold the constant values *true* or *false* and is instantiated by τ_{req}.)

- *Conditional:* because most DPS domains are stochastic, agent strategies will typically have alternative branches for different situations in the domain. For example, in SI, it might be that the strategy for $Agent_i$ is to send all its local data to $Agent_j$ if $Agent_i$ finds that its first sensor has returned a value greater than 40, otherwise it should send only that sensor value. Actions like these that are done under only certain conditions are referred to as conditional actions. In order to evaluate the time performance of strategies that contain this sort of non-determinism, the tool must know the probability that each condition will occur in the domain. Conditional actions can be represented by a match operation. For example, if $Agent_i$ sends its local results to $Agent_j$ with a certain probability, pi_j, then this can be represented by:

  $$Agent_i = \nu(p)\left[\, p = p_{ij} \,\right]\overline{c}_{ij} m_{ij}$$
  $$Agent_j = c_{ij}\left(m_{ij}\right).$$

  (p is a bound name indicating the probability of the conditional action. We assume that this probability is provided by the strategy designer).

The agent processes must express ordering and/or mutual exclusion relations among the agent's actions. For each pair of actions, t_i and t_j, one of the following relations might hold:

- *Precedence relation:* the precedence relation between t_i and t_j is modelled with the prefix operation: $t_i.t_j$, which means t_j cannot be executed before t_i has been completed. t_j is called the *successor* of t_i. Because communication is asynchronous,

the precedence relation also holds between corresponding communication actions: the receiving action cannot be active unless the corresponding sending action has been completed.

- *Parallel relation:* the parallel relation between t_i and t_j is modelled as $t_i \mid t_j$, which means t_i and t_j can be executed in any order.
- *Probabilistic mutual relation:* the probabilistic mutual relation between t_i and t_j is modelled as: $[p = p_i]t_i + [p = p_j]t_j$ which means at each time, either t_i or t_j can be executed with probability p_i or p_j respectively, but not both. In this case, $p_i + p_j = 1$.

An example of a pi-calculus model of a DPS coordination strategy is shown in Figure 2. We will also use this example strategy to demonstrate our deadlock detection and prevention approach. The strategy is based on a sensor interpretation application for a simulated sensor network. This system consists of four agents, each with direct access to only a subset of the sensors and thus only a subset of the global data. Each agent has the goal of determining whether some event has occurred or not. In order to accomplish their goals, each agent needs to process its own local data and also obtain the processing results of some of the other agents to integrate with its local results. Which of the other agents results are needed from, will depend on the characteristics of an agent's data, i.e., this is a *stochastic* strategy. In this example, we assume that each agent must send its local results to two other agents, and that agents can determine which two other agents by themselves (by examining the characteristics of their own local sensor data). For simplicity, we have the agents either send to the two other lowest numbered agents or the two other highest numbered agents. In the pi-calculus specification in Figure 2, c_{ij} represents the channel through which $Agent_i$ sends messages to $Agent_j$, m_{ij} is message that $Agent_i$ sends to $Agent_j$, τ_{ij} is an internal action for preparing m_{ij}, τ_{rij} is the internal action for processing the message received from another agent, and p_{ij} is the probability that $Agent_i$ sends its message to $Agent_j$.

Figure 2 Pi-calculus model of a four-agent coordination strategy

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4 \mid$$

$$Agent_1 = \nu(p)\begin{pmatrix} \tau_{13}.\bar{c}_{13}m_{13} \mid \left([p = p_{12}]\tau_{12}.\bar{c}_{12}m_{12} + [p = p_{14}]\tau_{14}.\bar{c}_{14}m_{14}\right) \mid \\ c_{21}(m_{21}).\tau_{R21} \mid c_{31}(m_{31}).\tau_{R31} \mid c_{41}(m_{41}).\tau_{R41} \end{pmatrix}$$

$$Agent_2 = \nu(p)\begin{pmatrix} \tau_{23}.\bar{c}_{23}m_{23} \mid \left([p = p_{21}]\tau_{21}.\bar{c}_{21}m_{21} + [p = p_{24}]\tau_{24}.\bar{c}_{24}m_{24}\right) \mid \\ c_{12}(m_{12}).\tau_{R12} \mid c_{32}(m_{32}).\tau_{R32} \mid c_{42}(m_{42}).\tau_{R42} \end{pmatrix}$$

$$Agent_3 = \nu(p)\begin{pmatrix} \tau_{32}.\bar{c}_{32}m_{32} \mid \left([p = p_{31}]\tau_{31}.\bar{c}_{31}m_{31} + [p = p_{34}]\tau_{34}.\bar{c}_{34}m_{34}\right) \mid \\ c_{13}(m_{13}).\tau_{R13} \mid c_{23}(m_{23}).\tau_{R23} \mid c_{43}(m_{43}).\tau_{R43} \end{pmatrix}$$

$$Agent_4 = \nu(p)\begin{pmatrix} \tau_{42}.\bar{c}_{42}m_{42} \mid \left([p = p_{43}]\tau_{43}.\bar{c}_{43}m_{43} + [p = p_{41}]\tau_{41}.\bar{c}_{41}m_{41}\right) \mid \\ c_{24}(m_{24}).\tau_{R24} \mid c_{34}(m_{34}).\tau_{R34} \mid c_{14}(m_{14}).\tau_{R14} \end{pmatrix}$$

5 Deadlock detection in the DPS coordination strategy

One of the advantages of using pi-calculus is that its rigorous definition allows assessment of various system properties, such as performance and deadlock. This section describes how pi-calculus specification may be used to detect deadlock in a DPS coordination strategy, at the design stage. For this purpose, we adapt Kobayashi's definition for deadlock (Kobayashi, 2002), to the case of *asynchronous pi-calculus*:

Definition 1: (Kobayashi, 2002) (deadlock, deadlock-freedom). A process is in deadlock if

1 $P = (\nu\ \tilde{y})\left(x(\tilde{z})^{C}.Q \mid R\right)$ or $P = (\nu\ \tilde{y})(\overline{x}\tilde{z}^{c}.Q \mid R)$

2 there is no P' such that $P \rightarrow P'$. A process P is deadlock-free if there exist no Q such that $P \rightarrow {}^{*}Q$ and Q is in deadlock.

This definition is outlined in the scope of Kobayashi's language, which is a subset of the polyadic pi-calculus. The polyadic pi-calculus (Milner, 1993) is defined the same way as the monadic pi-calculus except that the output/input actions are allowed to send/receive vectors of names at a single reduction step. Accordingly, the input and output actions in this calculus are given by: $x(\tilde{z})$ and $\overline{x}\langle\tilde{y}\rangle$, where $\tilde{z}$ and $\tilde{y}$ represent vectors. Kobayashi's definition states that a process P is in deadlock if it contains a receiving or a sending action that must proceed, but P is locked and cannot be further reduced to another process P'. The annotation c indicates that the respective task is expected to succeed and the value is either read or sent through the channel. Kobayashi's definition ensures that, at least one thread will be executed to perform the annotated task. Thus in $P = (\nu\tilde{y})\left(x(\tilde{z})^{c}.Q \middle| R\right)$, a message will be received through channel x as long as the sender agent does not diverge. Although this is not the semantic used in the context of our research, we found the definition valid to search for possible deadlocks. In what follows, we define the deadlock situation in a DPS coordination strategy based on Kobayashi's definition. However, unlike Definition 1, we do not use the notion of annotation of actions, as some actions in DPS are probabilistic and may not necessarily succeed. Moreover, the case where an agent has a pending sending action does not occur in our model as asynchronous pi-calculus does not allow for output prefixes (i.e., no action is performed after writing in a channel).

Definition 2: A DPS coordination strategy is in deadlock condition if it reaches a state in which at least one agent contains an input prefix which will never succeed. That is:

$$DPSCS = Agent_1 \mid Agent_2 \mid \ldots \mid Agent_n$$

$$Agent_i = C_i\left(m_i\right).Q \mid R$$

And there is no p such that $Agent_i \rightarrow P$.

Hence, to detect a deadlock condition in a coordination strategy the pi-calculus reduction rules are applied to the model until a state is reached in which there is an agent with an input prefix, which will never succeed. If no such state is found in a scenario of a coordination strategy then the scenario is said to be deadlock free. If all the scenarios of a coordination strategy are deadlock free, then the strategy itself is deadlock free.

To demonstrate the deadlock detection procedure, consider, as an example, the coordination strategy modelled in Figure 2. One can observe that this strategy could lead to a deadlock in some scenarios. For instance, if the four agents in the model send messages to the two other lowest numbered agents as described in Figure 3, then $Agent_3$ and $Agent_4$ would end up with input prefixes which will not succeed.

Figure 3 A deadlock scenario from the coordination strategy in Figure 2

Scenario

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4 \mid$$

$$Agent_1 = \tau_{13}.\overline{c}_{13}m_{13} \mid \tau_{12}.\overline{c}_{12}m_{12} \mid c_{21}(m_{21}).\tau_{R21} \mid c_{31}(m_{31}).\tau_{R31} \mid c_{41}(m_{41}).\tau_{R41}$$

$$Agent_2 = \tau_{23}.\overline{c}_{23}m_{23} \mid \tau_{21}.\overline{c}_{21}m_{21} \mid c_{12}(m_{12}).\tau_{R12} \mid c_{32}(m_{32}).\tau_{R32} \mid c_{42}(m_{42}).\tau_{R42}$$

$$Agent_3 = \tau_{32}.\overline{c}_{32}m_{32} \mid \tau_{31}.\overline{c}_{31}m_{31} \mid c_{13}(m_{13}).\tau_{R13} \mid c_{23}(m_{23}).\tau_{R23} \mid c_{43}(m_{43}).\tau_{R43}$$

$$Agent_4 = \tau_{42}.\overline{c}_{42}m_{42} \mid \tau_{41}.\overline{c}_{41}m_{41} \mid c_{24}(m_{24}).\tau_{R24} \mid c_{R34}(m_{34}).\tau_{R34} \mid c_{14}(m_{14}).\tau_{R14}$$

To prove the deadlock condition in the above scenario, the pi-calculus reduction rules are applied to the model as follows. A more detailed version of the derivations is provided in Appendix A.

In the first reduction step, the pair $\tau_{13}.\overline{c}_{13}\,m_{13}$ and $c_{13}(m_{13})$ are executed and the scenario is reduced to:

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$

$$Agent_1 = 0 \mid (\tau_{12}.\overline{c}_{12}m_{12}) \mid c_{21}(m_{21}).\tau_{R21} \mid c_{31}(m_{31}).\tau_{R31} \mid c_{41}(m_{41}).\tau_{R41}$$

$$Agent_2 = \tau_{23}.\overline{c}_{23}m_{23} \mid (\tau_{21}.\overline{c}_{21}m_{21}) \mid c_{12}(m_{12}).\tau_{R12} \mid c_{32}(m_{32}).\tau_{R32} \mid c_{42}(m_{42}).\tau_{R42}$$

$$Agent_3 = \tau_{32}.\overline{c}_{32}m_{32} \mid (\tau_{31}.\overline{c}_{31}m_{31}) \mid 0.\tau_{R13} \mid c_{23}(m_{23}).\tau_{R23} \mid c_{43}(m_{43}).\tau_{R43}$$

$$Agent_4 = \tau_{42}.\overline{c}_{42}m_{42} \mid (\tau_{41}.\overline{c}_{41}m_{41}) \mid c_{24}(m_{24}).\tau_{R24} \mid c_{34}(m_{34}).\tau_{R34} \mid \mid c_{14}(m_{14}).\tau_{R14}$$

Similarly the other pairs of input/output actions are executed in parallel, namely: $\tau_{12}.\overline{c}_{12}m_{13}/c_{12}(m_{12})$, $\tau_{21}.\overline{c}_{21}m_{21}/c_{21}(m_{21})$, $\tau_{23}.\overline{c}_{23}m_{23}/c_{23}(m_{23})$, $\tau_{31}.\overline{c}_{31}m_{31}/c_{31}(m_{31})$ $\tau_{32}.\overline{c}_{32}m_{32}/c_{32}(m_{32})$, and $\tau_{42}.\overline{c}_{42}m_{42}/c_{42}(m_{42})$. As a result, the scenario is reduced to:

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$

$$Agent_1 = \nu(p)\left(0 \mid 0.\tau_{R21} \mid 0.\tau_{R31} \mid 0.\tau_{R41}\right)$$

$$Agent_2 = \nu(p)\left(0 \mid 0 \mid 0.\tau_{R12} \mid 0.\tau_{R32} \mid 0.\tau_{R42}\right)$$

$$Agent_3 = \nu(p)\left(0|0|0|0.\tau_{R23}|c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(0|0|c_{24}\left(m_{24}\right).\tau_{R24}|c_{34}\left(m_{34}\right).\tau_{R34}|c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

In the next reduction step, the computational tasks: τ_{r21}, τ_{r31}, τ_{r41}, τ_{r12}, τ_{r32}, τ_{r42} and τ_{r23} are executed and the scenario arrives at the following state:

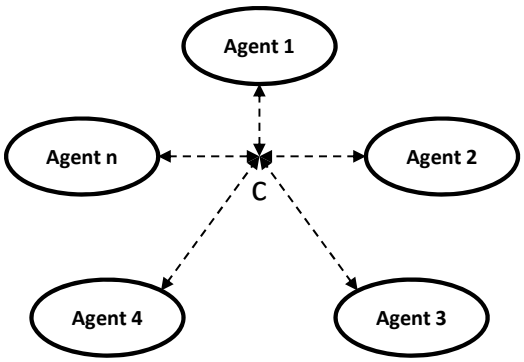
$$DPSCS = Agent_3|Agent_4$$
$$Agent_3 = \nu(p)\left(c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(c_{24}\left(m_{24}\right).\tau_{R24}|c_{34}\left(m_{34}\right).\tau_{R34}|c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

In this state, both $Agent_3$ and $Agent_4$ are left with pending receiving actions and input prefixes which can never succeed. Thus, based on Definition 2, the coordination strategy modelled in Figure 2, may lead into a deadlock condition.

6 Introducing a new deadlock-free model

In the previous section, we demonstrated how the pi-calculus model of a coordination strategy is used for deadlock detection and, as an example, we identified a potential deadlock condition in the coordination strategy of Figure 2. In this section, we show how deadlock could be prevented by imposing certain constraints on the pi-calculus model. For example, in the strategy of Figure 2, if we modify the model such that every agent is forced to receive at least one message, then we would be able to avoid possible deadlock scenarios. For this purpose, we have to slightly change the communications model. Unlike the previous model, where each pair of agents used two names for communication, in the new model a single name is used for all communications. All messages are sent and received through name c at any time. This model is shown in the following figure.

Figure 4 A single name communication model for DPS agents



Using this communication model, all the agent's actions defined in Section 4 remain the same except for the communication action. Since all agents use the same name for communication, the input/output actions should be modified to send and receive the source ID and the destination ID along with the message:

$$Agent_i = \overline{c}\left(s_{id}, d_{id}, m\right) \mid P$$
$$Agent_j = c\left(s_{id}, d_{id}, m\right) \mid Q$$

where s_{id}, and d_{id} denote the id of the source and the destination agents respectively. As was seen in the strategy of Figure 2, the input action is commonly followed by a computational task that is performed by the destination agent to process the message it has received. However, in the single name communication model, before doing any processing on the received message, the destination agent needs to know the source of the message in order to perform the computations accordingly. Hence, it is convenient to define an abstract read action for each agent as shown in Figure 5. For each agent, A_i, a read process is created that listens on channel c until it receives three names: s_{id} which encodes the message sender identifier, d_{id} which is the message destination identifier, and m which carries the actual message. At this point, if the message destination is agent A_j itself, i.e., ($[d_{id} = A_j]$) then it executes a computational action according to the source of the message, i.e., ($\Sigma[s_{id} = A_k]\tau_{Rkj}$).

Figure 5 Agents read operations in the single communication model

$$Read_{Agent1}(m) = c\left(s_{id}, d_{id}, m\right).\left(\left[d_{id} = A_1\right]\left(\begin{array}{l}\left[s_{id} = A_2\right]\tau_{R21} + \\ \left[s_{id} = A_3\right]\tau_{R31} + \left[s_{id} = A_4\right]\tau_{R41}\end{array}\right)\right)$$
$$Read_{Agent2}(m) = c\left(s_{id}, d_{id}, m\right).\left(\left[d_{id} = A_2\right]\left(\begin{array}{l}\left[s_{id} = A_1\right]\tau_{R12} + \\ \left[s_{id} = A_3\right]\tau_{R32} + \left[s_{id} = A_4\right]\tau_{R42}\end{array}\right)\right)$$
$$Read_{Agent3}(m) = c\left(s_{id}, d_{id}, m\right).\left(\left[d_{id} = A_3\right]\left(\begin{array}{l}\left[s_{id} = A_1\right]\tau_{R13} + \\ \left[s_{id} = A_2\right]\tau_{R23} + \left[s_{id} = A_4\right]\tau_{R43}\end{array}\right)\right)$$
$$Read_{Agent4}(m) = c\left(s_{id}, d_{id}, m\right).\left(\left[d_{id} = A_4\right]\left(\begin{array}{l}\left[s_{id} = A_1\right]\tau_{R14} + \\ \left[s_{id} = A_2\right]\tau_{R24} + \left[s_{id} = A_3\right]\tau_{R34}\end{array}\right)\right)$$

In order to avoid deadlock scenarios, we are now able to precisely incorporate in our model the restriction that every agent receives at least one message and executes a computational task accordingly. To this end, we force the reading tasks to succeed in this new model. We impose two restrictions on the pi-calculus model to assure that every agent receives at least one message.

First, we let the read operation be performed either once, twice, or n times, where n is the number of agents. This means that an agent might receive messages from one, two, or all other agents.

Second, we enforce the constraint that the reading group of actions should be eventually reduced to 0; in other words, when more than one reduction is possible for the pi-calculus model, we keep the ones that reduce the reading group of actions to 0 and discard the rest.

Consider again our example coordination strategy of four agents. For $Agent_1$ the group of reading actions would be:

$$\left(\left(Read_{Agent1}\right) + \left(Read_{Agent1} \mid Read_{Agent1}\right) + \left(Read_{Agent1} \mid Read_{Agent1} \mid Read_{Agent1}\right)\right)$$

where one of the three possibilities must be satisfied during reduction to avoid deadlock. If the same restriction is applied for the other agents, the new model, illustrated in Figure 6, complies with the deadlock free requirement.

Figure 6 An example of modelling a deadlock free coordination strategy

$$DPSCS = Agent_1 \,|\, Agent_2 \,|\, Agent_3 \,|\, Agent_4$$

$$Agent_1 = \nu(p)\begin{pmatrix} \tau_{13}.\overline{c}\langle A_1, A_3, m_{13}\rangle \,| \\ \left([p = p_{12}]\tau_{12}.\overline{c}\langle A_1, A_2, m_{12}\rangle + [p = p_{14}]\tau_{14}.\overline{c}\langle A_1, A_4, m_{14}\rangle\right) \,| \\ \left(\begin{pmatrix} (Read_{Agent1}) + (Read_{Agent1} \,|\, Read_{Agent1}) + \\ (Read_{Agent1} \,|\, Read_{Agent1} \,|\, Read_{Agent1}) \end{pmatrix}\right) \end{pmatrix}$$

$$Agent_2 = \nu(p)\begin{pmatrix} \tau_{23}.\overline{c}\langle A_2, A_3, m_{23}\rangle \,| \\ \left([p = p_{21}]\tau_{21}.\overline{c}\langle A_2, A_1, m_{21}\rangle + [p = p_{24}]\tau_{24}.\overline{c}\langle A_2, A_4, m_{24}\rangle\right) \,| \\ \left(\begin{pmatrix} (Read_{Agent2}) + (Read_{Agent2} \,|\, Read_{Agent2}) + \\ (Read_{Agent2} \,|\, Read_{Agent2} \,|\, Read_{Agent2}) \end{pmatrix}\right) \end{pmatrix}$$

$$Agent_3 = \begin{pmatrix} \nu(p)(\tau_{32}.\overline{c}\langle A_3, A_2, m_{32}\rangle \,| \\ \left([p = p_{31}]\tau_{31}.\overline{c}\langle A_3, A_1, m_{31}\rangle + [p = p_{34}]\tau_{34}.\overline{c}\langle A_3, A_4, m_{34}\rangle\right) \,| \\ \left(\begin{pmatrix} (Read_{Agent3}) + (Read_{Agent3} \,|\, Read_{Agent3}) + \\ (Read_{Agent3} \,|\, Read_{Agent3} \,|\, Read_{Agent3}) \end{pmatrix}\right) \end{pmatrix}$$

$$Agent_4 = \begin{pmatrix} \nu(p)(\tau_{42}.\overline{c}\langle A_4, A_2, m_{42}\rangle \,| \\ \left([p = p_{43}]\tau_{43}.\overline{c}\langle A_4, A_3, m_{43}\rangle + [p = p_{41}]\tau_{41}.\overline{c}\langle A_4, A_1, m_{41}\rangle\right) \,| \\ \left(\begin{pmatrix} (Read_{Agent4}) + (Read_{Agent4} \,|\, Read_{Agent4}) + \\ (Read_{Agent4} \,|\, Read_{Agent4} \,|\, Read_{Agent4}) \end{pmatrix}\right) \end{pmatrix}$$

As can be seen in the figure, after execution of internal tasks each agent sends a triad of names via channel *c* and then the reading actions are executed.

One concern might be that, if the reduction of one of the reading processes is performed at agent *i* but the destination of the message is not agent *i*, then the reduction leads to an invalid state where the message is not delivered to its destination. However, in that case, the group of reading actions is not reduced to 0, and this situation is prevented by enforcing the second constraint above.

Another concern could be that the three (or two) exclusive read actions in the new model may cause the same message to be read three (or two) times and execute the same task three times; for example τ_{r21} in $Read_{Agent1}$. However, this is not the case as only one instance of each message is sent during the occurrence of each scenario. This avoids

executing multiple times the same task. The designer should be aware of these details to avoid unexpected model outputs.

In what follows, we shall prove the deadlock freedom of the strategy in Figure 6. In this proof we show that given *DPSCS* there is no scenario *S* such that *DPCS* → *S* and *S* is in deadlock condition. By reduction ad absurdum, we first assume that there exist such a scenario and proceed to find a *S* which satisfies the conditions in Definition 2. In order to keep it short, we collapse many reductions in one step as in the previous section.

$$DPSCS = Agent_1 \,|\, Agent_2 \,|\, Agent_3 \,|\, Agent_4$$

$$Agent_1 = \nu(p)\left(\begin{array}{l}\tau_{13}.\overline{c}\langle A_1, A_3, m_{13}\rangle \,|\, \tau_{12}.\overline{c}\langle A_1, A_2, m_{12}\rangle \,| \\ \left(\left(Read_{Agent1}\right) + \left(Read_{Agent1} \,|\, Read_{Agent1}\right) + \right. \\ \left.\left(Read_{Agent1} \,|\, Read_{Agent1} \,|\, Read_{Agent1}\right)\right)\end{array}\right)$$

$$Agent_2 = \nu(p)\left(\begin{array}{l}\tau_{23}.\overline{c}\langle A_2, A_3, m_{23}\rangle \,|\, \tau_{21}.\overline{c}\langle A_2, A_1, m_{21}\rangle \,| \\ \left(\left(Read_{Agent2}\right) + \left(Read_{Agent2} \,|\, Read_{Agent2}\right) + \right. \\ \left.\left(Read_{Agent2} \,|\, Read_{Agent2} \,|\, Read_{Agent2}\right)\right)\end{array}\right)$$

$$Agent_3 = \nu(p)\left(\begin{array}{l}\tau_{32}.\overline{c}\langle A_3, A_2, m_{32}\rangle \,|\, \tau_{31}.\overline{c}\langle A_3, A_1, m_{31}\rangle \,| \\ \left(\left(Read_{Agent3}\right) + \left(Read_{Agent3} \,|\, Read_{Agent3}\right) + \right. \\ \left.\left(Read_{Agent3} \,|\, Read_{Agent3} \,|\, Read_{Agent3}\right)\right)\end{array}\right)$$

$$Agent_4 = \nu(p)\left(\begin{array}{l}\tau_{42}.\overline{c}\langle A_4, A_2, m_{42}\rangle \,|\, \tau_{41}.\overline{c}\langle A_4, A_1, m_{41}\rangle \,| \\ \left(\left(Read_{Agent4}\right) + \left(Read_{Agent4} \,|\, Read_{Agent4}\right) + \right. \\ \left.\left(Read_{Agent4} \,|\, Read_{Agent4} \,|\, Read_{Agent4}\right)\right)\end{array}\right)$$

After τ_{13} and τ_{23} are executed and names A_1, A_3, m_{13} and A_2, A_3, m_{23} are sent via channel c and the two corresponding parallel reading actions are also performed, we have:

$$DPSCS = Agent_1 \,|\, Agent_2 \,|\, Agent_3 \,|\, Agent_4$$

$$Agent_1 = \nu(p)\left(0 \,|\, \tau_{12}.\overline{c}\langle A_1, A_2, m_{12}\rangle \,|\left(\begin{array}{l}\left(Read_{Agent1}\right) + \left(Read_{Agent1} \,|\, Read_{Agent1}\right) + \\ \left(Read_{Agent1} \,|\, Read_{Agent1} \,|\, Read_{Agent1}\right)\end{array}\right)\right)$$

$$Agent_2 = \nu(p)\left(0 \,|\, \tau_{21}.\overline{c}\langle A_2, A_1, m_{21}\rangle \,|\left(\begin{array}{l}\left(Read_{Agent2}\right) + \left(Read_{Agent2} \,|\, Read_{Agent2}\right) + \\ \left(Read_{Agent2} \,|\, Read_{Agent2} \,|\, Read_{Agent2}\right)\end{array}\right)\right)$$

$$Agent_3 = \nu(p)\left(\tau_{32}.\overline{c}\langle A_3, A_2, m_{32}\rangle \,|\, \tau_{31}.\overline{c}\langle A_3, A_1, m_{31}\rangle \,|\, 0\right)$$

$$Agent_4 = \nu(p)\left(\begin{array}{l}\tau_{42}.\overline{c}\langle A_4, A_2, m_{42}\rangle \,|\, \tau_{41}.\overline{c}\langle A_4, A_1, m_{41}\rangle \,| \\ \left(\left(Read_{Agent4}\right) + \left(Read_{Agent4} \,|\, Read_{Agent4}\right) + \right. \\ \left.\left(Read_{Agent4} \,|\, Read_{Agent4} \,|\, Read_{Agent4}\right)\right)\end{array}\right)$$

Similar reductions are performed for τ_{12}, τ_{32}, τ_{42} and the sending/receiving of A_1, A_2, m_{12}, A_3, A_2, m_{32}, and A_4, A_2, m_{42}. The scenario is reduced to:

$$DPSCS = Agent_1 \,|\, Agent_2 \,|\, Agent_3 \,|\, Agent_4$$

$$Agent_1 = \nu(p)\left(0 \mid 0 \mid \left(\begin{array}{l}\left(Read_{Agent1}\right) + \left(Read_{Agent1} \mid Read_{Agent1}\right) + \\ \left(Read_{Agent1} \mid Read_{Agent1} \mid Read_{Agent1}\right)\end{array}\right)\right)$$

$$Agent_2 = \nu(p)\left(0 \mid \tau_{21}.\overline{c}\left\langle A_2, A_1, m_{21}\right\rangle \mid 0\right)$$

$$Agent_3 = \nu(p)\left(0 \mid \tau_{31}.\overline{c}\left\langle A_3, A_1, m_{31}\right\rangle \mid 0\right)$$

$$Agent_4 = \nu(p)\left(0 \mid \tau_{41}.\overline{c}\left\langle A_4, A_1, m_{41}\right\rangle \mid \left(\begin{array}{l}\left(Read_{Agent4}\right) + \left(Read_{Agent4} \mid Read_{Agent4}\right) + \\ \left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\end{array}\right)\right)$$

Similar reductions are applied to τ_{21}, τ_{31}, τ_{41} and the sending/receiving of A_2, A_1, m_{21}, A_3, A_1, m_{31}, and A_4, A_1, m_{41} through c:

$$DPSCS = Agent_1 \,|\, Agent_2 \,|\, Agent_3 \,|\, Agent_4$$

$$Agent_1 = \nu(p)(0 \mid 0 \mid 0)$$

$$Agent_2 = \nu(p)(0 \mid 0 \mid 0)$$

$$Agent_3 = \nu(p)(0 \mid 0 \mid 0)$$

$$Agent_4 = \nu(p)\left(0 \mid 0 \mid \left(\begin{array}{l}\left(Read_{Agent4}\right) + \left(Read_{Agent4} \mid Read_{Agent4}\right) + \\ \left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\end{array}\right)\right)$$

or

$$DPSCS = 0 \mid 0 \mid 0 \mid \left(\begin{array}{l}\left(Read_{Agent4}\right) + \left(Read_{Agent4} \mid Read_{Agent4}\right) + \\ \left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\end{array}\right)$$

The last state in the above derivation contradicts the constraint that the group of reading actions must be reduced to 0. $Agent_4$ is left with some reading actions which cannot be further reduced to 0. Hence, the assumption of the existence of a scenario S with a deadlock condition is not valid and the strategy is deadlock free.

7 Summary and future work

The primary goal of the research reported on here was to establish the basis for a verification system of a DPS coordination, using pi-calculus. In a previous research (Khorasani et al., 2009) we proposed a tool for modelling and analysing the time performance of DPS coordination strategies. The chosen language for defining coordination strategies was pi-calculus as its formal structure provides the basis for mathematical analysis of various properties such as performance and deadlock.

In this paper we further demonstrated the advantages of using pi-calculus for modelling DPS coordination strategies. In particular, we showed how pi-calculus model could be used for deadlock detection and prevention. We illustrated the deadlock

detection procedure via an example and we showed how certain restriction could be imposed on the model to assure a deadlock free strategy. Our approach asserts deadlock freedom, as long as the communication between agents succeeds. Nonetheless, analysis under limited communication is still needed. These features validate the pi-calculus suitability for DPS coordination modelling and verification.

It is worth noting that any methodology developed over formal modelling tools such as pi-calculus suffers from the complexity of the formalisation and difficulty of performing the reductions and verification of the objectives. To address this problem, authors are in the process of simplifying utilisation of pi-calculus by implementing a comprehensive visual toolbox for pi-calculus that includes the following components:

- A visualisation tool for visually modelling MAS in pi-calculus by performing drag-and-drop actions instead of dealing with complexity of pi-calculus (Ahmad and Rahimi, 2008).
- An expert system-based automated reduction module that applies pi-calculus reduction rules on the modelled system automatically (Rahimi and Dillards, 2008).
- An automated bi-directional verification module pi-calculus (build over the reduction system) that compares the outcomes of the modelled system against a set of defined objectives (Rahimi and Ahmad, 2011).
- A performance evaluation methodology that adds new formulation to pi-calculus to make performance evaluation possible (Rahimi, 2006). Implementation of a performance evaluation module for the toolbox will begin in close future.

Using such toolbox, the system designers will be able to write the pi-calculus specification of DPS systems effortlessly, visually evaluate the actions taken by the system (based on the reduction rules), and automatically assess the deadlock freedom.

References

Ahmad, R. and Rahimi, S. (2008) 'ACVisualizer: a visualization tool for API–calculus', *Multi-Agent and Grid Systems, an International Journal*, Vol. 4, No. 3, pp.271–291.

Amadio, R.M., Castellani, I. and Sangiorgi, D. (1996) 'On bisimulations for the asynchronous pi-calculus', *Proceedings of the Proceedings of the 7th International Conference on Concurrency Theory*, Springer-Verlag.

Becker, R., Zilberstein, S. and Lesser, V. (2004) 'Decentralized Markov decision processes with event-driven interactions', *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*', Vol. 1, New York, New York, IEEE Computer Society.

Carver, N. and Lesser, V. (2003) 'Domain monotonicity and the performance of local solutions strategies for CDPS-based distributed sensor interpretation and distributed diagnosis', *Autonomous Agents and Multi-Agent Systems*, Vol. 6, pp.35–76.

Durfee, E.H. (2001) 'Distributed problem solving and planning', *Multi-agents Systems and Applications*, Springer-Verlag, Inc., New York, pp.118–149.

Honda, K. and Tokoro, M. (1991) 'An object calculus for asynchronous communication', *Proceedings of ECOOP'91*, LNCS 512, Berlin, Springer-Verlag, pp.133–147.

Khorasani, E.S, Carver, N. and Rahimi, S. (2009) 'Performance evaluation of DPS coordination strategies modeled in pi-calculus', *International Journal Intelligent Information and Database Systems*, Vol. 3, No. 4, pp.419–439.

Kobayashi, N. (2002) 'A type system for lock-free processes', *Information and Computation*, Vol. 177, pp.122–159.

Lesser, V.R. (2002) 'Evolution of the GPGP/TÆMS domain-independent coordination framework', *Proceedings of the Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1*, Bologna, Italy, ACM.

Milner, R. (1993) 'The polyadic pi-calculus: a tutorial', F.L. Bauer, W. Brauer and H. Schwichtenberg (Eds.): *Logic and Algebra of Specification*, Springer-Verlag.

Milner, R., Parrow, J. and Walker, D. (1992a) 'A calculus of mobile processes, I', *Inf. Comput.*, Vol. 100, pp.1–40.

Milner, R., Parrow, J. and Walker, D. (1992b) 'A calculus of mobile processes, II', *Inf. Comput.*, Vol. 100, pp.41–77.

Rahimi, S. and Ahmad, R. (2011) 'An open bisimilarity based automated verification tool for pi-calculus family of process calculi', to appear in *Scalable Computing: Practice and Experience, an International Journal*.

Rahimi, S. (2006) 'Toward a performance evaluation methodology for pi-calculus family of formal modeling languages', *Scalable Computing: Practice and Experience*, Editorial, Vol. 7, No. 1, pp.i–vi.

Rahimi, S. and Dillards, J. (2008) 'An expert system for pi-calculus and a pi-calculus automated reduction', *Proceedings of the International Conference of Software Engineering Research and Practice* (*SER 08*), pp.462–468.

Sangiorgi, D. and Walker, D. (2001) *The π-Calculus: A Theory of Mobile Processes*, Cambridge University Press, Cambridge, UK.

Shen, J., Lesser, V. and Carver, N. (2003) 'Minimizing communication cost in a distributed Bayesian network using a decentralized MDP', *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, Melbourne, Australia, ACM.

Appendix A

Proof 1 detail: deadlock detection for the previous model

$$DPSCS = Agent_1 \,|\, Agent_2 \,|\, Agent_3 \,|\, Agent_4$$

$$Agent_1 = \nu(p)\begin{pmatrix} \tau_{13}.\bar{c}_{13}m_{13} \,|\, \left([p = p_{12}]\tau_{12}.\bar{c}_{12}m_{12} + [p = p_{14}]\tau_{14}.\bar{c}_{14}m_{14}\right) \\ c_{21}(m_{21}).\tau_{R21} \,|\, c_{31}(m_{31}).\tau_{R31} \,|\, c_{41}(m_{41}).\tau_{R41} \end{pmatrix}$$

$$Agent_2 = \nu(p)\begin{pmatrix} \tau_{23}.\bar{c}_{23}m_{23} \,|\, \left([p = p_{21}]\tau_{21}.\bar{c}_{21}m_{21} + [p = p_{24}]\tau_{24}.\bar{c}_{24}m_{24}\right) \\ c_{12}(m_{12}).\tau_{R12} \,|\, c_{32}(m_{32}).\tau_{R32} \,|\, c_{42}(m_{42}).\tau_{R42} \end{pmatrix}$$

$$Agent_3 = \nu(p)\begin{pmatrix} \tau_{32}.\bar{c}_{32}m_{32} \,|\, \left([p = p_{31}]\tau_{31}.\bar{c}_{31}m_{31} + [p = p_{34}]\tau_{34}.\bar{c}_{34}m_{34}\right) \\ c_{13}(m_{13}).\tau_{R13} \,|\, c_{23}(m_{23}).\tau_{R23} \,|\, c_{43}(m_{43}).\tau_{R43} \end{pmatrix}$$

$$Agent_4 = \nu(p)\begin{pmatrix} \tau_{42}.\bar{c}_{42}m_{42} \,|\, \left([p = p_{43}]\tau_{43}.\bar{c}_{43}m_{43} + [p = p_{41}]\tau_{41}.\bar{c}_{41}m_{41}\right) \\ c_{24}(m_{24}).\tau_{R24} \,|\, c_{34}(m_{34}).\tau_{R34} \,|\, c_{14}(m_{14}).\tau_{R14} \end{pmatrix}$$

We start by selecting the scenario of interest.

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$
$$Agent_1 = \nu(p)\left(\tau_{13}.\overline{c}_{13}m_{13} \mid (\tau_{12}.\overline{c}_{12}m_{12}) \mid c_{21}(m_{21}).\tau_{R21} \mid c_{31}(m_{31}).\tau_{R31} \mid c_{41}(m_{41}).\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(\tau_{23}.\overline{c}_{23}m_{23} \mid (\tau_{21}.\overline{c}_{21}m_{21}) \mid c_{12}(m_{12}).\tau_{R12} \mid c_{32}(m_{32}).\tau_{R32} \mid c_{42}(m_{42}).\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(\tau_{32}.\overline{c}_{32}m_{32} \mid (\tau_{31}.\overline{c}_{31}m_{31}) \mid c_{13}(m_{13}).\tau_{R13} \mid c_{23}(m_{23}).\tau_{R23} \mid c_{43}(m_{43}).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(\tau_{42}.\overline{c}_{42}m_{42} \mid (\tau_{41}.\overline{c}_{41}m_{41}) \mid c_{24}(m_{24}).\tau_{R24} \mid c_{34}(m_{34}).\tau_{R34} \mid c_{14}(m_{14}).\tau_{R14}\right)$$

After τ_{13} is executed and name m_{13} is sent through channel $\overline{c}_{13}$, and the corresponding reading is also performed, we have:

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$
$$Agent_1 = \nu(p)\left(\overline{c}_{13}m_{13} \mid (\tau_{12}.\overline{c}_{12}m_{12}) \mid c_{21}(m_{21}).\tau_{R21} \mid c_{31}(m_{31}).\tau_{R31} \mid c_{41}(m_{41}).\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(\tau_{23}.\overline{c}_{23}m_{23} \mid (\tau_{21}.\overline{c}_{21}m_{21}) \mid c_{12}(m_{12}).\tau_{R12} \mid c_{32}(m_{32}).\tau_{R32} \mid c_{42}(m_{42}).\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(\tau_{32}.\overline{c}_{32}m_{32} \mid (\tau_{31}.\overline{c}_{31}m_{31}) \mid c_{13}(m_{13}).\tau_{R13} \mid c_{23}(m_{23}).\tau_{R23} \mid c_{43}(m_{43}).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(\tau_{42}.\overline{c}_{42}m_{42} \mid (\tau_{41}.\overline{c}_{41}m_{41}) \mid c_{24}(m_{24}).\tau_{R24} \mid c_{34}(m_{34}).\tau_{R34} \mid c_{14}(m_{14}).\tau_{R14}\right)$$

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$
$$Agent_1 = \nu(p)\left(0 \mid (\tau_{12}.\overline{c}_{12}m_{12}) \mid c_{21}(m_{21}).\tau_{R21} \mid c_{31}(m_{31}).\tau_{R31} \mid c_{41}(m_{41}).\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(\tau_{23}.\overline{c}_{23}m_{23} \mid (\tau_{21}.\overline{c}_{21}m_{21}) \mid c_{12}(m_{12}).\tau_{R12} \mid c_{32}(m_{32}).\tau_{R32} \mid c_{42}(m_{42}).\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(\tau_{32}.\overline{c}_{32}m_{32} \mid (\tau_{31}.\overline{c}_{31}m_{31}) \mid 0.\tau_{R13} \mid c_{23}(m_{23}).\tau_{R23} \mid c_{43}(m_{43}).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(\tau_{42}.\overline{c}_{42}m_{42} \mid (\tau_{41}.\overline{c}_{41}m_{41}) \mid c_{24}(m_{24}).\tau_{R24} \mid c_{34}(m_{34}).\tau_{R34} \mid c_{14}(m_{14}).\tau_{R14}\right)$$

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$
$$Agent_1 = \nu(p)\left((\tau_{12}.\overline{c}_{12}m_{12}) \mid c_{21}(m_{21}).\tau_{R21} \mid c_{31}(m_{31}).\tau_{R31} \mid c_{41}(m_{41}).\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(\tau_{23}.\overline{c}_{23}m_{23} \mid (\tau_{21}.\overline{c}_{21}m_{21}) \mid c_{12}(m_{12}).\tau_{R12} \mid c_{32}(m_{32}).\tau_{R32} \mid c_{42}(m_{42}).\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(\tau_{32}.\overline{c}_{32}m_{32} \mid (\tau_{31}.\overline{c}_{31}m_{31}) \mid 0 \mid \tau_{R13} \mid c_{23}(m_{23}).\tau_{R23} \mid c_{43}(m_{43}).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(\tau_{42}.\overline{c}_{42}m_{42} \mid (\tau_{41}.\overline{c}_{41}m_{41}) \mid c_{24}(m_{24}).\tau_{R24} \mid c_{34}(m_{34}).\tau_{R34} \mid c_{14}(m_{14}).\tau_{R14}\right)$$

After reduction of τ_{12} and sending of m_{12} through $\overline{c}_{12}$:

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$
$$Agent_1 = \nu(p)\left(0.\overline{c}_{12}m_{12} \mid c_{21}(m_{21}).\tau_{R21} \mid c_{31}(m_{31}).\tau_{R31} \mid c_{41}(m_{41}).\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(\tau_{23}.\overline{c}_{23}m_{23} \mid (\tau_{21}.\overline{c}_{21}m_{21}) \mid c_{12}(m_{12}).\tau_{R12} \mid c_{32}(m_{32}).\tau_{R32} \mid c_{42}(m_{42}).\tau_{R42}\right)$$

$$Agent_3 = \nu(p)\left(\tau_{32}.\overline{c}_{32}m_{32}\middle|\left(\tau_{31}.\overline{c}_{31}m_{31}\right)\middle|0\middle|\tau_{R13}\middle|c_{23}\left(m_{23}\right).\tau_{R23}\middle|c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(\tau_{42}.\overline{c}_{42}m_{42}\middle|\left(\tau_{41}.\overline{c}_{41}m_{41}\right)\middle|c_{24}\left(m_{24}\right).\tau_{R24}\middle|c_{34}\left(m_{34}\right).\tau_{R34}\middle|c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

$$DPSCS = Agent_1\middle|Agent_2\middle|Agent_3\middle|Agent_4$$
$$Agent_1 = \nu(p)\left(0\middle|c_{21}\left(m_{21}\right).\tau_{R21}\middle|c_{31}\left(m_{31}\right).\tau_{R31}\middle|c_{41}\left(m_{41}\right).\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(\tau_{23}.\overline{c}_{23}m_{23}\middle|\left(\tau_{21}.\overline{c}_{21}m_{21}\right)\middle|0.\tau_{R12}\middle|c_{32}\left(m_{32}\right).\tau_{R32}\middle|c_{42}\left(m_{42}\right).\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(\tau_{32}.\overline{c}_{32}m_{32}\middle|\left(\tau_{31}.\overline{c}_{31}m_{31}\right)\middle|0\middle|\tau_{R13}\middle|c_{23}\left(m_{23}\right).\tau_{R23}\middle|c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(\tau_{42}.\overline{c}_{42}m_{42}\middle|\left(\tau_{41}.\overline{c}_{41}m_{41}\right)\middle|c_{24}\left(m_{24}\right).\tau_{R24}\middle|c_{34}\left(m_{34}\right).\tau_{R34}\middle|c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

Reduction of τ_{23} and sending of m_{23} through $\overline{c}_{23}$, τ_{21} and sending of m_{21} through $\overline{c}_{21}$, τ_{32} and sending of m_{32} through $\overline{c}_{32}$, τ_{31} and sending of m_{31} through $\overline{c}_{31}$, τ_{41} and sending of m_{41} through $\overline{c}_{41}$

$$DPSCS = Agent_1\middle|Agent_2\middle|Agent_3\middle|Agent_4$$
$$Agent_1 = \nu(p)\left(0\middle|c_{21}\left(m_{21}\right).\tau_{R21}\middle|c_{31}\left(m_{31}\right).\tau_{R31}\middle|c_{41}\left(m_{41}\right).\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(0.\overline{c}_{23}m_{23}\middle|\left(\tau_{21}.\overline{c}_{21}m_{21}\right)\middle|0.\tau_{R12}\middle|c_{32}\left(m_{32}\right).\tau_{R32}\middle|c_{42}\left(m_{42}\right).\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(\tau_{32}.\overline{c}_{32}m_{32}\middle|\left(\tau_{31}.\overline{c}_{31}m_{31}\right)\middle|0\middle|\tau_{R13}\middle|c_{23}\left(m_{23}\right).\tau_{R23}\middle|c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(\tau_{42}.\overline{c}_{42}m_{42}\middle|\left(\tau_{41}.\overline{c}_{41}m_{41}\right)\middle|c_{24}\left(m_{24}\right).\tau_{R24}\middle|c_{34}\left(m_{34}\right).\tau_{R34}\middle|c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

$$DPSCS = Agent_1\middle|Agent_2\middle|Agent_3\middle|Agent_4$$
$$Agent_1 = \nu(p)\left(0\middle|c_{21}\left(m_{21}\right).\tau_{R21}\middle|c_{31}\left(m_{31}\right).\tau_{R31}\middle|c_{41}\left(m_{41}\right).\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(0\middle|\left(\tau_{21}.\overline{c}_{21}m_{21}\right)\middle|0.\tau_{R12}\middle|c_{32}\left(m_{32}\right).\tau_{R32}\middle|c_{42}\left(m_{42}\right).\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(0.\overline{c}_{32}m_{32}\middle|\left(\tau_{31}.\overline{c}_{31}m_{31}\right)\middle|0\middle|0.\tau_{R23}\middle|c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(\tau_{42}.\overline{c}_{42}m_{42}\middle|\left(\tau_{41}.\overline{c}_{41}m_{41}\right)\middle|c_{24}\left(m_{24}\right).\tau_{R24}\middle|c_{34}\left(m_{34}\right).\tau_{R34}\middle|c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

$$DPSCS = Agent_1\middle|Agent_2\middle|Agent_3\middle|Agent_4$$
$$Agent_1 = \nu(p)\left(0\middle|c_{21}\left(m_{21}\right).\tau_{R21}\middle|c_{31}\left(m_{31}\right).\tau_{R31}\middle|c_{41}\left(m_{41}\right).\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(0\,|\,\left(0.\overline{c}_{21}m_{21}\right)\middle|0.\tau_{R12}\middle|c_{32}\left(m_{32}\right).\tau_{R32}\middle|c_{42}\left(m_{42}\right).\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(0.\overline{c}_{32}m_{32}\middle|\left(\tau_{31}.\overline{c}_{31}m_{31}\right)\middle|0\middle|0.\tau_{R23}\middle|c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(\tau_{42}.\overline{c}_{42}m_{42}\middle|\left(\tau_{41}.\overline{c}_{41}m_{41}\right)\middle|c_{24}\left(m_{24}\right).\tau_{R24}\middle|c_{34}\left(m_{34}\right).\tau_{R34}\middle|c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

$$DPSCS = Agent_1\middle|Agent_2\middle|Agent_3\middle|Agent_4$$
$$Agent_1 = \nu(p)\left(0\middle|0.\tau_{R21}\middle|c_{31}\left(m_{31}\right).\tau_{R31}\middle|c_{41}\left(m_{41}\right).\tau_{R41}\right)$$

$$Agent_2 = \nu(p)\left(0\middle|0\middle|0.\tau_{R12}\middle|c_{32}\left(m_{32}\right).\tau_{R32}\middle|c_{42}\left(m_{42}\right).\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(\tau_{32}.\overline{c}_{32}m_{32}\middle|\left(\tau_{31}.\overline{c}_{31}m_{31}\right)\middle|0\middle|0.\tau_{R23}\middle|c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(\tau_{42}.\overline{c}_{42}m_{42}\middle|\left(\tau_{41}.\overline{c}_{41}m_{41}\right)\middle|c_{24}\left(m_{24}\right).\tau_{R24}\middle|c_{34}\left(m_{34}\right).\tau_{R34}\middle|c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

$$DPSCS = Agent_1\middle|Agent_2\middle|Agent_3\middle|Agent_4$$
$$Agent_1 = \nu(p)\left(0\middle|0.\tau_{R21}\middle|c_{31}\left(m_{31}\right).\tau_{R31}\middle|c_{41}\left(m_{41}\right).\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(0\middle|0\middle|0.\tau_{R12}\middle|c_{32}\left(m_{32}\right).\tau_{R32}\middle|c_{42}\left(m_{42}\right).\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(0.\overline{c}_{32}m_{32}\middle|\left(\tau_{31}.\overline{c}_{31}m_{31}\right)\middle|0\middle|0.\tau_{R23}\middle|c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(\tau_{42}.\overline{c}_{42}m_{42}\middle|\left(\tau_{41}.\overline{c}_{41}m_{41}\right)\middle|c_{24}\left(m_{24}\right).\tau_{R24}\middle|c_{34}\left(m_{34}\right).\tau_{R34}\middle|c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

$$DPSCS = Agent_1\middle|Agent_2\middle|Agent_3\middle|Agent_4$$
$$Agent_1 = \nu(p)\left(0\middle|0.\tau_{R21}\middle|c_{31}\left(m_{31}\right).\tau_{R31}\middle|c_{41}\left(m_{41}\right).\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(0\middle|0\middle|0.\tau_{R12}\middle|0.\tau_{R32}\middle|c_{42}\left(m_{42}\right).\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(0.\overline{c}_{32}m_{32}\middle|\left(\tau_{31}.\overline{c}_{31}m_{31}\right)\middle|0\middle|0.\tau_{R23}\middle|c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(\tau_{42}.\overline{c}_{42}m_{42}\middle|\left(\tau_{41}.\overline{c}_{41}m_{41}\right)\middle|c_{24}\left(m_{24}\right).\tau_{R24}\middle|c_{34}\left(m_{34}\right).\tau_{R34}\middle|c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

$$DPSCS = Agent_1\middle|Agent_2\middle|Agent_3\middle|Agent_4$$
$$Agent_1 = \nu(p)\left(0\middle|0.\tau_{R21}\middle|0.\tau_{R31}\middle|c_{41}\left(m_{41}\right).\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(0\middle|0\middle|0.\tau_{R12}\middle|0.\tau_{R32}\middle|c_{42}\left(m_{42}\right).\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(0\middle|0\middle|0\middle|0.\tau_{R23}\middle|c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(0.\overline{c}_{42}m_{42}\middle|\left(\tau_{41}.\overline{c}_{41}m_{41}\right)\middle|c_{24}\left(m_{24}\right).\tau_{R24}\middle|c_{34}\left(m_{34}\right).\tau_{R34}\middle|c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

$$DPSCS = Agent_1\middle|Agent_2\middle|Agent_3\middle|Agent_4$$
$$Agent_1 = \nu(p)\left(0\middle|0.\tau_{R21}\middle|0.\tau_{R31}\middle|c_{41}\left(m_{41}\right).\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(0\middle|0\middle|0.\tau_{R12}\middle|0.\tau_{R32}\middle|0.\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(0\middle|0\middle|0\middle|0.\tau_{R23}\middle|c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(0\middle|\left(\tau_{41}.\overline{c}_{41}m_{41}\right)\middle|c_{24}\left(m_{24}\right).\tau_{R24}\middle|c_{34}\left(m_{34}\right).\tau_{R34}\middle|c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

$$DPSCS = Agent_1 \,|\, Agent_2 \,|\, Agent_3 \,|\, Agent_4$$
$$Agent_1 = \nu(p)\left(0 \,|\, 0.\tau_{R21} \,|\, 0.\tau_{R31} \,|\, c_{41}(m_{41}).\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(0 \,|\, 0 \,|\, 0.\tau_{R12} \,|\, 0.\tau_{R32} \,|\, 0.\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(0 \,|\, 0 \,|\, 0 \,|\, 0.\tau_{R23} \,|\, c_{43}(m_{43}).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(0 \,|\, (0.\overline{c}_{41}m_{41}) \,|\, c_{24}(m_{24}).\tau_{R24} \,|\, c_{34}(m_{34}).\tau_{R34} \,|\, c_{14}(m_{14}).\tau_{R14}\right)$$

$$DPSCS = Agent_1 \,|\, Agent_2 \,|\, Agent_3 \,|\, Agent_4$$
$$Agent_1 = \nu(p)\left(0 \,|\, 0.\tau_{R21} \,|\, 0.\tau_{R31} \,|\, 0.\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(0 \,|\, 0 \,|\, 0.\tau_{R12} \,|\, 0.\tau_{R32} \,|\, 0.\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(0 \,|\, 0 \,|\, 0 \,|\, 0.\tau_{R23} \,|\, c_{43}(m_{43}).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(0 \,|\, 0 \,|\, c_{24}(m_{24}).\tau_{R24} \,|\, c_{34}(m_{34}).\tau_{R34} \,|\, c_{14}(m_{14}).\tau_{R14}\right)$$

The following are the reductions for after-input tasks τ_{R21}, τ_{R31}, τ_{R41}, τ_{R12}, τ_{R32}, τ_{R42} and τ_{R23} to reach the final deadlock state:

$$DPSCS = Agent_1 \,|\, Agent_2 \,|\, Agent_3 \,|\, Agent_4$$
$$Agent_1 = \nu(p)\left(0 \,|\, 0 \,|\, 0.\tau_{R31} \,|\, 0.\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(0 \,|\, 0 \,|\, 0.\tau_{R12} \,|\, 0.\tau_{R32} \,|\, 0.\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(0 \,|\, 0 \,|\, 0 \,|\, 0.\tau_{R23} \,|\, c_{43}(m_{43}).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(0 \,|\, 0 \,|\, c_{24}(m_{24}).\tau_{R24} \,|\, c_{34}(m_{34}).\tau_{R34} \,|\, c_{14}(m_{14}).\tau_{R14}\right)$$

$$DPSCS = Agent_1 \,|\, Agent_2 \,|\, Agent_3 \,|\, Agent_4$$
$$Agent_1 = \nu(p)\left(0 \,|\, 0 \,|\, 0 \,|\, 0.\tau_{R41}\right)$$
$$Agent_2 = \nu(p)\left(0 \,|\, 0 \,|\, 0.\tau_{R12} \,|\, 0.\tau_{R32} \,|\, 0.\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(0 \,|\, 0 \,|\, 0 \,|\, 0.\tau_{R23} \,|\, c_{43}(m_{43}).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(0 \,|\, 0 \,|\, c_{24}(m_{24}).\tau_{R24} \,|\, c_{34}(m_{34}).\tau_{R34} \,|\, c_{14}(m_{14}).\tau_{R14}\right)$$

$$DPSCS = Agent_1 \,|\, Agent_2 \,|\, Agent_3 \,|\, Agent_4$$
$$Agent_1 = \nu(p)\left(0 \,|\, 0 \,|\, 0 \,|\, 0\right)$$
$$Agent_2 = \nu(p)\left(0 \,|\, 0 \,|\, 0.\tau_{R12} \,|\, 0.\tau_{R32} \,|\, 0.\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(0 \,|\, 0 \,|\, 0 \,|\, 0.\tau_{R23} \,|\, c_{43}(m_{43}).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(0 \,|\, 0 \,|\, c_{24}(m_{24}).\tau_{R24} \,|\, c_{34}(m_{34}).\tau_{R34} \,|\, c_{14}(m_{14}).\tau_{R14}\right)$$

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$
$$Agent_2 = \nu(p)\left(0 \mid 0 \mid 0 \mid 0.\tau_{R32} \mid 0.\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(0 \mid 0 \mid 0 \mid 0.\tau_{R23} \mid c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(0 \mid 0 \mid c_{24}\left(m_{24}\right).\tau_{R24} \mid c_{34}\left(m_{34}\right).\tau_{R34} \mid c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$
$$Agent_2 = \nu(p)\left(0 \mid 0 \mid 0 \mid 0 \mid 0.\tau_{R42}\right)$$
$$Agent_3 = \nu(p)\left(0 \mid 0 \mid 0 \mid 0.\tau_{R23} \mid c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(0 \mid 0 \mid c_{24}\left(m_{24}\right).\tau_{R24} \mid c_{34}\left(m_{34}\right).\tau_{R34} \mid c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$
$$Agent_2 = \nu(p)\left(0 \mid 0 \mid 0 \mid 0 \mid 0\right)$$
$$Agent_3 = \nu(p)\left(0 \mid 0 \mid 0 \mid 0.\tau_{R23} \mid c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(0 \mid 0 \mid c_{24}\left(m_{24}\right).\tau_{R24} \mid c_{34}\left(m_{34}\right).\tau_{R34} \mid c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$
$$Agent_3 = \nu(p)\left(0 \mid 0 \mid 0 \mid 0 \mid c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(0 \mid 0 \mid c_{24}\left(m_{24}\right).\tau_{R24} \mid c_{34}\left(m_{34}\right).\tau_{R34} \mid c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$
$$Agent_3 = \nu(p)\left(c_{43}\left(m_{43}\right).\tau_{R43}\right)$$
$$Agent_4 = \nu(p)\left(c_{24}\left(m_{24}\right).\tau_{R24} \mid c_{34}\left(m_{34}\right).\tau_{R34} \mid c_{14}\left(m_{14}\right).\tau_{R14}\right)$$

The final state for this model is in deadlock since both $Agent_3$ and $Agent_4$ has read actions that are waiting forever for a message that never arrives.

Appendix B

Proof 2 detail: deadlock-freedom detection for the new model

The scenario of interest for this model is:

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$
$$Agent_1 = \nu(p)\left(\begin{array}{l} \tau_{13}.\overline{c}\left\langle A_1, A_3, m_{13}\right\rangle \mid \tau_{12}.\overline{c}\left\langle A_1, A_2, m_{12}\right\rangle \mid \\ \left(\left(Read_{Agent1}\right) + \left(Read_{Agent1} \mid Read_{Agent1}\right) + \right. \\ \left.\left(Read_{Agent1} \mid Read_{Agent1} \mid Read_{Agent1}\right)\right. \end{array}\right)$$

$$Agent_2 = \nu(p)\left(\begin{array}{l}\tau_{23}.\overline{c}\left\langle A_2, A_3, m_{23}\right\rangle \mid \tau_{21}.\overline{c}\left\langle A_2, A_1, m_{21}\right\rangle \\ \left(\left(Read_{Agent2}\right)+\left(Read_{Agent2} \mid Read_{Agent2}\right)+\right. \\ \left.\left(Read_{Agent2} \mid Read_{Agent2} \mid Read_{Agent2}\right)\right)\end{array}\right)$$

$$Agent_3 = \nu(p)\left(\begin{array}{l}\tau_{32}.\overline{c}\left\langle A_3, A_2, m_{32}\right\rangle \mid \tau_{31}.\overline{c}\left\langle A_3, A_1, m_{31}\right\rangle \\ \left(\left(Read_{Agent3}\right)+\left(Read_{Agent3} \mid Read_{Agent3}\right)+\right. \\ \left.\left(Read_{Agent3} \mid Read_{Agent3} \mid Read_{Agent3}\right)\right)\end{array}\right)$$

$$Agent_4 = \nu(p)\left(\begin{array}{l}\tau_{42}.\overline{c}\left\langle A_4, A_2, m_{42}\right\rangle \mid \tau_{41}.\overline{c}\left\langle A_4, A_1, m_{41}\right\rangle \\ \left(\left(Read_{Agent4}\right)+\left(Read_{Agent4} \mid Read_{Agent4}\right)+\right. \\ \left.\left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\right)\end{array}\right)$$

The following are the reductions for τ_{13} and τ_{23}, and names A_1, A_3, m_{13} and A_2, A_3, m_{23} sent through channel c:

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$

$$Agent_1 = \nu(p)\left(\begin{array}{l}0.\overline{c}\left\langle A_1, A_3, m_{13}\right\rangle \mid \tau_{12}.\overline{c}\left\langle A_1, A_2, m_{12}\right\rangle \\ \left(\left(Read_{Agent1}\right)+\left(Read_{Agent1} \mid Read_{Agent1}\right)+\right. \\ \left.\left(Read_{Agent1} \mid Read_{Agent1} \mid Read_{Agent1}\right)\right)\end{array}\right)$$

$$Agent_2 = \nu(p)\left(\begin{array}{l}\tau_{23}.\overline{c}\left\langle A_2, A_3, m_{23}\right\rangle \mid \tau_{21}.\overline{c}\left\langle A_2, A_1, m_{21}\right\rangle \\ \left(\left(Read_{Agent2}\right)+\left(Read_{Agent2} \mid Read_{Agent2}\right)+\right. \\ \left.\left(Read_{Agent2} \mid Read_{Agent2} \mid Read_{Agent2}\right)\right)\end{array}\right)$$

$$Agent_3 = \nu(p)\left(\begin{array}{l}\tau_{32}.\overline{c}\left\langle A_3, A_2, m_{32}\right\rangle \mid \tau_{31}.\overline{c}\left\langle A_3, A_1, m_{31}\right\rangle \\ \left(\left(Read_{Agent3}\right)+\left(Read_{Agent3} \mid Read_{Agent3}\right)+\right. \\ \left.\left(Read_{Agent3} \mid Read_{Agent3} \mid Read_{Agent3}\right)\right)\end{array}\right)$$

$$Agent_4 = \nu(p)\left(\begin{array}{l}\tau_{42}.\overline{c}\left\langle A_4, A_2, m_{42}\right\rangle \mid \tau_{41}.\overline{c}\left\langle A_4, A_1, m_{41}\right\rangle \\ \left(\left(Read_{Agent4}\right)+\left(Read_{Agent4} \mid Read_{Agent4}\right)+\right. \\ \left.\left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\right)\end{array}\right)$$

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$

$$Agent_1 = \nu(p)\left(\begin{array}{l}0.\overline{c}\left\langle A_1, A_3, m_{13}\right\rangle \mid \tau_{12}.\overline{c}\left\langle A_1, A_2, m_{12}\right\rangle \\ \left(\left(Read_{Agent1}\right)+\left(Read_{Agent1} \mid Read_{Agent1}\right)+\right. \\ \left.\left(Read_{Agent1} \mid Read_{Agent1} \mid Read_{Agent1}\right)\right)\end{array}\right)$$

$$Agent_2 = \nu(p)\left(\begin{array}{l} 0.\overline{c}\left\langle A_2, A_3, m_{23}\right\rangle \mid \tau_{21}.\overline{c}\left\langle A_2, A_1, m_{21}\right\rangle \\ \left(\left(Read_{Agent2}\right) + \left(Read_{Agent2} \mid Read_{Agent2}\right) + \right. \\ \left.\left(Read_{Agent2} \mid Read_{Agent2} \mid Read_{Agent2}\right)\right) \end{array}\right)$$

$$Agent_3 = \nu(p)\left(\begin{array}{l} \tau_{32}.\overline{c}\left\langle A_3, A_2, m_{32}\right\rangle \mid \tau_{31}.\overline{c}\left\langle A_3, A_1, m_{31}\right\rangle \\ \left(\left(Read_{Agent3}\right) + \left(Read_{Agent3} \mid Read_{Agent3}\right) + \right. \\ \left.\left(Read_{Agent3} \mid Read_{Agent3} \mid Read_{Agent3}\right)\right) \end{array}\right)$$

$$Agent_4 = \nu(p)\left(\begin{array}{l} \tau_{42}.\overline{c}\left\langle A_4, A_2, m_{42}\right\rangle \mid \tau_{41}.\overline{c}\left\langle A_4, A_1, m_{41}\right\rangle \\ \left(\left(Read_{Agent4}\right) + \left(Read_{Agent4} \mid Read_{Agent4}\right) + \right. \\ \left.\left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\right) \end{array}\right)$$

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$

$$Agent_1 = \nu(p)\left(0 \mid \tau_{12}.\overline{c}\left\langle A_1, A_2, m_{12}\right\rangle \mid \left(\begin{array}{l} \left(Read_{Agent1}\right) + \left(Read_{Agent1} \mid Read_{Agent1}\right) + \\ \left(Read_{Agent1} \mid Read_{Agent1} \mid Read_{Agent1}\right) \end{array}\right)\right)$$

$$Agent_2 = \nu(p)\left(0 \mid \tau_{21}.\overline{c}\left\langle A_2, A_1, m_{21}\right\rangle \mid \left(\begin{array}{l} \left(Read_{Agent2}\right) + \left(Read_{Agent2} \mid Read_{Agent2}\right) + \\ \left(Read_{Agent2} \mid Read_{Agent2} \mid Read_{Agent2}\right) \end{array}\right)\right)$$

$$Agent_3 = \nu(p)\left(\tau_{32}.\overline{c}\left\langle A_3, A_2, m_{32}\right\rangle \mid \tau_{31}.\overline{c}\left\langle A_3, A_1, m_{31}\right\rangle \mid 0\right)$$

$$Agent_4 = \nu(p)\left(\begin{array}{l} \tau_{42}.\overline{c}\left\langle A_4, A_2, m_{42}\right\rangle \mid \tau_{41}.\overline{c}\left\langle A_4, A_1, m_{41}\right\rangle \\ \left(\left(Read_{Agent4}\right) + \left(Read_{Agent4} \mid Read_{Agent4}\right) + \right. \\ \left.\left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\right) \end{array}\right)$$

The following are the reductions for τ_{12}, τ_{32} and τ_{42} and the corresponding sending actions of A_1, A_2, m_{12}, A_3, A_2, m_{32} and A_4, A_2, m_{42} through c:

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$

$$Agent_1 = \nu(p)\left(\begin{array}{l} 0.\overline{c}\left\langle A_1, A_3, m_{13}\right\rangle \mid \tau_{12}.\overline{c}\left\langle A_1, A_2, m_{12}\right\rangle \\ \left(\left(Read_{Agent1}\right) + \left(Read_{Agent1} \mid Read_{Agent1}\right) + \right. \\ \left.\left(Read_{Agent1} \mid Read_{Agent1} \mid Read_{Agent1}\right)\right) \end{array}\right)$$

$$Agent_2 = \nu(p)\left(\tau_{21}.\overline{c}\left\langle A_2, A_1, m_{21}\right\rangle \mid \left(\begin{array}{l} \left(Read_{Agent2}\right) + \left(Read_{Agent2} \mid Read_{Agent2}\right) + \\ \left(Read_{Agent2} \mid Read_{Agent2} \mid Read_{Agent2}\right) \end{array}\right)\right)$$

$$Agent_3 = \nu(p)\left(\tau_{32}.\overline{c}\left\langle A_3, A_2, m_{32}\right\rangle \mid \tau_{31}.\overline{c}\left\langle A_3, A_1, m_{31}\right\rangle\right)$$

$$Agent_4 = \nu(p)\left(\begin{array}{l}\tau_{42}.\overline{c}\left\langle A_4, A_2, m_{42}\right\rangle \left| \tau_{41}.\overline{c}\left\langle A_4, A_1, m_{41}\right\rangle \right| \\ \left(\left(Read_{Agent4}\right) + \left(Read_{Agent4} \mid Read_{Agent4}\right) + \right. \\ \left.\left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\right)\end{array}\right)$$

$$DPSCS = Agent_1 \left| Agent_2 \right| Agent_3 \left| Agent_4 \right.$$

$$Agent_1 = \nu(p)\left(0.\overline{c}\left\langle A_1, A_3, m_{13}\right\rangle \left| \left(\begin{array}{l}\left(Read_{Agent1}\right) + \left(Read_{Agent1} \mid Read_{Agent1}\right) + \\ \left(Read_{Agent1} \mid Read_{Agent1} \mid Read_{Agent1}\right)\end{array}\right)\right.\right)$$

$$Agent_2 = \nu(p)\left(\tau_{21}.\overline{c}\left\langle A_2, A_1, m_{21}\right\rangle \left| \left(\begin{array}{l}\left(Read_{Agent2}\right) + \left(Read_{Agent2} \mid Read_{Agent2}\right) + \\ \left(Read_{Agent2} \mid Read_{Agent2} \mid Read_{Agent2}\right)\end{array}\right)\right.\right)$$

$$Agent_3 = \nu(p)\left(0.\overline{c}\left\langle A_3, A_2, m_{32}\right\rangle \left| \tau_{31}.\overline{c}\left\langle A_3, A_1, m_{31}\right\rangle \right.\right)$$

$$Agent_4 = \nu(p)\left(\begin{array}{l}\tau_{42}.\overline{c}\left\langle A_4, A_2, m_{42}\right\rangle \left| \tau_{41}.\overline{c}\left\langle A_4, A_1, m_{41}\right\rangle \right| \\ \left(\left(Read_{Agent4}\right) + \left(Read_{Agent4} \mid Read_{Agent4}\right) + \right. \\ \left.\left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\right)\end{array}\right)$$

$$DPSCS = Agent_1 \left| Agent_2 \right| Agent_3 \left| Agent_4 \right.$$

$$Agent_1 = \nu(p)\left(0.\overline{c}\left\langle A_1, A_3, m_{13}\right\rangle \left| \left(\begin{array}{l}\left(Read_{Agent1}\right) + \left(Read_{Agent1} \mid Read_{Agent1}\right) + \\ \left(Read_{Agent1} \mid Read_{Agent1} \mid Read_{Agent1}\right)\end{array}\right)\right.\right)$$

$$Agent_2 = \nu(p)\left(\tau_{21}.\overline{c}\left\langle A_2, A_1, m_{21}\right\rangle \left| \left(\begin{array}{l}\left(Read_{Agent2}\right) + \left(Read_{Agent2} \mid Read_{Agent2}\right) + \\ \left(Read_{Agent2} \mid Read_{Agent2} \mid Read_{Agent2}\right)\end{array}\right)\right.\right)$$

$$Agent_3 = \nu(p)\left(0.\overline{c}\left\langle A_3, A_2, m_{32}\right\rangle \left| \tau_{31}.\overline{c}\left\langle A_3, A_1, m_{31}\right\rangle \right.\right)$$

$$Agent_4 = \nu(p)\left(\begin{array}{l}0.\overline{c}\left\langle A_4, A_2, m_{42}\right\rangle \left| \tau_{41}.\overline{c}\left\langle A_4, A_1, m_{41}\right\rangle \right| \\ \left(\left(Read_{Agent4}\right) + \left(Read_{Agent4} \mid Read_{Agent4}\right) + \right. \\ \left.\left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\right)\end{array}\right)$$

$$DPSCS = Agent_1 \left| Agent_2 \right| Agent_3 \left| Agent_4 \right.$$

$$Agent_1 = \nu(p)\left(0\left| \left(\begin{array}{l}\left(Read_{Agent1}\right) + \left(Read_{Agent1} \mid Read_{Agent1}\right) + \\ \left(Read_{Agent1} \mid Read_{Agent1} \mid Read_{Agent1}\right)\end{array}\right)\right.\right)$$

$$Agent_2 = \nu(p)\left(\tau_{21}.\overline{c}\left\langle A_2, A_1, m_{21}\right\rangle \left| 0\right.\right)$$

$$Agent_3 = \nu(p)\left(0\left| \tau_{31}.\overline{c}\left\langle A_3, A_1, m_{31}\right\rangle \right.\right)$$

$$Agent_4 = \nu(p)\left(0\left| \tau_{41}.\overline{c}\left\langle A_4, A_1, m_{41}\right\rangle \right| \left(\begin{array}{l}\left(Read_{Agent4}\right) + \left(Read_{Agent4} \mid Read_{Agent4}\right) + \\ \left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\end{array}\right)\right.\right)$$

Similarly for reduction of: τ_{21}, τ_{31} and τ_{41} and the corresponding sending actions of A_2, A_1, m_{21} and A_3, A_1, m_{31} and A_4, A_1, m_{41} through c:

$$DPSCS = Agent_1 \left| Agent_2 \right| Agent_3 \left| Agent_4 \right.$$

$$Agent_1 = \nu(p)\left(\begin{pmatrix}\left(Read_{Agent1}\right)+\left(Read_{Agent1} \mid Read_{Agent1}\right)+\\ \left(Read_{Agent1} \mid Read_{Agent1} \mid Read_{Agent1}\right)\end{pmatrix}\right)$$

$$Agent_2 = \nu(p)\left(0.\overline{c}\left\langle A_2, A_1, m_{21}\right\rangle\right)$$

$$Agent_3 = \nu(p)\left(\tau_{31}.\overline{c}\left\langle A_3, A_1, m_{31}\right\rangle\right)$$

$$Agent_4 = \nu(p)\left(\tau_{41}.\overline{c}\left\langle A_4, A_1, m_{41}\right\rangle \left|\begin{pmatrix}\left(Read_{Agent4}\right)+\left(Read_{Agent4} \mid Read_{Agent4}\right)+\\ \left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\end{pmatrix}\right.\right)$$

$$DPSCS = Agent_1 \left| Agent_2 \right| Agent_3 \left| Agent_4 \right.$$

$$Agent_1 = \nu(p)\left(\begin{pmatrix}\left(Read_{Agent1}\right)+\left(Read_{Agent1} \mid Read_{Agent1}\right)+\\ \left(Read_{Agent1} \mid Read_{Agent1} \mid Read_{Agent1}\right)\end{pmatrix}\right)$$

$$Agent_2 = \nu(p)\left(0.\overline{c}\left\langle A_2, A_1, m_{21}\right\rangle\right)$$

$$Agent_3 = \nu(p)\left(0.\overline{c}\left\langle A_3, A_1, m_{31}\right\rangle\right)$$

$$Agent_4 = \nu(p)\left(\tau_{41}.\overline{c}\left\langle A_4, A_1, m_{41}\right\rangle \left|\begin{pmatrix}\left(Read_{Agent4}\right)+\left(Read_{Agent4} \mid Read_{Agent4}\right)+\\ \left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\end{pmatrix}\right.\right)$$

$$DPSCS = Agent_1 \left| Agent_2 \right| Agent_3 \left| Agent_4 \right.$$

$$Agent_1 = \nu(p)\left(\begin{pmatrix}\left(Read_{Agent1}\right)+\left(Read_{Agent1} \mid Read_{Agent1}\right)+\\ \left(Read_{Agent1} \mid Read_{Agent1} \mid Read_{Agent1}\right)\end{pmatrix}\right)$$

$$Agent_2 = \nu(p)\left(0.\overline{c}\left\langle A_2, A_1, m_{21}\right\rangle\right)$$

$$Agent_3 = \nu(p)\left(0.\overline{c}\left\langle A_3, A_1, m_{31}\right\rangle\right)$$

$$Agent_4 = \nu(p)\left(0.\overline{c}\left\langle A_4, A_1, m_{41}\right\rangle \left|\begin{pmatrix}\left(Read_{Agent4}\right)+\left(Read_{Agent4} \mid Read_{Agent4}\right)+\\ \left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\end{pmatrix}\right.\right)$$

$$DPSCS = Agent_1 \mid Agent_2 \mid Agent_3 \mid Agent_4$$

$$Agent_1 = \nu(p)(0)$$

$$Agent_2 = \nu(p)(0)$$

$$Agent_3 = \nu(p)(0)$$

$$Agent_4 = \nu(p)\left(0 \left|\begin{pmatrix}\left(Read_{Agent4}\right)+\left(Read_{Agent4} \mid Read_{Agent4}\right)+\\ \left(Read_{Agent4} \mid Read_{Agent4} \mid Read_{Agent4}\right)\end{pmatrix}\right.\right)$$

This state is not allowed under the new model and the deadlock condition is never reached.