T-Compound Interaction and Overhearing Agents

Eric Platon[1], Nicolas Sabouret[2], and Shinichi Honiden[1]

[1] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda, 101-8430 Tokyo
[2] Laboratoire d'Informatique de Paris 6, 8, Rue du Capitaine Scott, 75015 Paris
{platon, honiden}@nii.ac.jp, nicolas.sabouret@lip6.fr

Abstract. Overhearing is an indirect interaction type that enacts agents to listen to direct interactions among other agents without taking explicit part in the exchanges. In this paper, we propose a formal model of overhearing named T-compound and a methodology to describe generalised interaction networks in Multi-Agent Systems. The compound is defined with the π-calculus as an interaction composite. It is handled as an interaction primitive distinct from the traditional point-to-point one, so that our methodology can treat both cases homogeneously.

1 Introduction

Multi-agent systems (MAS) rely on the interactions among their agents, either human, hardware or software. Our work highlights the design of interactions in MAS, and we think relevant to exploit the recent concept of overhearing [1, 2] in addition to traditional direct interactions. This notion refers to one type of indirect interactions that occurs frequently in natural systems, typically when two agents are interacting, say discussing, and a third one is just a passive audience *that could however intervene* if required. This presentation of overhearing presents the positive aspects of the paradigm. The counterpart named 'eavesdropping' in the MAS community implies various concerns including security and reliability. In the frame of this paper, we focus on interaction requirements for *cooperative agents* to leverage overhearing advantages. We let eavesdropping as peculiar issue to be addressed separately.

Present work on overhearing exploited this mechanism in various scenarii such as monitoring agent systems and group formation. These applications show the relevance of the concept and its generality. However, it results from these systems that overhearing relies on an unusual interaction infrastructure, since current technologies only exploit direct links among agents. In this paper, we propose a formal model named the T-compound, based on the π–calculus [3], and a methodology to enable and encapsulate systematic use of overhearing interactions, when required. Exploitation of both direct and indirect interactions leads to new perspectives on systems, and we will show some situations whereby this double usage is even necessary.

This paper begins in Section 2 with a presentation of the concept of overhearing, its interest for MAS, and the relevance of formal modeling with the π–calculus. Then, we propose in Section 3 our model and a methodology to describe the interaction dimension in MAS. In Section 4, we develop an example with the proposed method. Finally, we relate our approach to other activities in modeling overhearing in Section 5, before concluding in Section 6.

2 MAS, Overhearing, and π–Calculus

2.1 MAS and Overhearing

The notion of overhearing was recently introduced in the MAS community as reported by Gutnik *et al.* [4]. It was originally proposed to endow agents with monitoring abilities to reason about the apparent behaviour of other agents, *i.e.* their interactions. This concept endows one agent with the capacity to capture information from the interaction of two or more other agents. Fig.1 depicts such a situation in its simplest case.

MAS presently exploit *direct interactions*, such as the discussion link between agents A and B on Fig.1. Overhearing is an instance of *indirect interaction* that might be relevant in the MAS paradigm. In fact, research in the field of natural sciences show that numerous MAS based on complex societies exploit meaningful forms of communication without explicit receiver. For instance, termites build their hills by working together, but they do not exchange any information directly. They rely on the notion of stigmergy whereby they determine their behaviours according to the current state of the environment. One termite puts a piece of material for the hill, and other termites (including the first one) will then pile on top [5].

Indirect interactions already leveraged relevant results in MAS with stigmergy and other techniques [6]. Overhearing has now an increasing number of applications. Original work refers to monitoring agent systems [7], large group communication [1], dynamic group formation [2], conversation recognition [4], and some forms of coordinations [8, 9, 10]. In addition, the phenomenon often appears in agent systems that focus on a variety of concerns. Hence, the Helper Agent reacts to *silences* in an instant messaging discussion between humans to suggest common topics and increase the interest of the participants [11]. The M

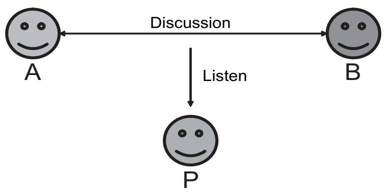

Fig. 1. Overhearing Situation

system surveys the actions of people in a virtual meeting room to optimise their workspaces at run-time [12]. COLLAGEN observes the flight ticket reservation process of the user to propose alternatives when no solution can be found for a given request [13]. In the remainder of this paper, we attempt to make more systematic the usage of overhearing in such systems, and novel applications.

2.2 Interaction and Formal Model

In order to exploit different kinds of interactions, MAS developers need a representation scheme. Formal models consist in proper representations of phenomena and remain mostly neutral regarding implementation details. Also, they can clarify the view of the system and reduce the effects of complexity by using mathematical compact formulae, based for instance on sets or recursivity.

In consequence, we propose in this paper a formal model that relies on the π-calculus from Milner [3] to leverage its interaction- and dynamism-oriented syntax, mechanisms, and expressive power that 'can *in principle* model (...) any computational aspect of agents' [14]. This heritage provides a robust model of traditional interactions and our extension enacts an instance of overhearing.

In addition to the grounds provided with the calculus, Milner developed a set of techniques to study concurrent systems, including equivalence relations. We expect the comparison between interaction structures and between apparent behaviours will enable advanced reasoning capabilities in agents exploiting overhearing. In particular, one agent may hear a conversation and try to match the stream to a known protocol [4]. As it is unlikely to perfectly match an existing models, the notion of equivalence allows more flexibility. In the remainder of this paper, we focus first on the syntax and semantics of the formal model, whereas the exploitation of equivalence belongs to our future work.

2.3 The π-Calculus in This Paper

The model presented in this paper exploits a subset of the π–calculus, originally from R. Milner [3]. This section aims at detailing the elements we retained and their meaning. The π–calculus features much more elements and advanced notions, but the present notations and mechanisms are sufficient in this paper.

The π–calculus is a modern process algebra for concurrent systems. It serves to represent and reason about interactions among concurrent processes and their dynamics such as mobility and changes in the interaction network (reorganisation, life-cycle). *In the frame of this paper*, we call agents the processes of the π–calculus, in the sense of the MAS community [15].

$\mathcal{P}_\pi$ is the set of agent names denoted by capitalised words. The set of Greek letters $\aleph = (\alpha, \beta, ...)$ represents the interaction channels that can link two agents. Other strings and small characters in the set *Str* label the messages that are sent in the different channels. Finally, I is an interval of integers [0,1,2,...].

Syntax. We now define the well-formed formulae (wff) of our restricted π–calculus.

- 0 is the agent termination, in addition to $\mathcal{P}_\pi$
  - $\rightarrow$ It is usually omitted at the end of definitions, i.e. $P = \alpha.Q$ is written instead of $P = \alpha.Q.0$
- $\langle . \rangle$ and $(.)$ represent the sending and reception operators
  - $\rightarrow$ They accept the same syntax for any channel α and message x: $\alpha\langle x\rangle$ and $\alpha(x)$. The operator omission denotes any of them is applied.
- '.' (dot) is the successor operator
  - $\rightarrow$ Given the channels α, β and the agent P, well-formed formulae are $\alpha.P$ and $\alpha.\beta.P$ (it is also generalised to n agents).
- 'new' is the restriction operator
  - $\rightarrow$ Given the channels α, β and the agent P, well-formed formulae are new(α)P and new(α)new(β)P. In the second case, we also write new($\alpha\beta$)P to have more compact formulae.
- $+$ represents the choice operator
  - $\rightarrow$ It allows writting $P + Q$ for any agent P and Q. The generalisation for n of agents is: $\sum_{i=1}^{n} A_i = (A_1 + ... + A_n)$
- $|$ is the concurrent operator
  - $\rightarrow$ It accepts the formula $P \mid Q$ for any agent P and Q, and the generalisation: $\prod_{i=1}^{n} A_i = (A_1 \mid ... \mid A_n)$

Finally, the well-formed agents verify the following equation. An agent P is either of the items in this formula, and also their compositions with recursive definitions.

$$P ::= \quad 0 \quad \vee \quad \alpha.P_0 \quad \vee \quad new(\alpha)P \quad \vee \quad \sum_{i \in I} P_i \quad \vee \quad \prod_{i \in I} P_i \tag{1}$$

Semantics. First, the agent termination 0 is a constant that means no activity, neither internal nor interactive. It is a final state that represents the termination (end of life) of agents. Along interaction channels, two complementary actions can occur, namely the sending and reception of messages. The formulae $\alpha\langle x\rangle$ and $\beta(y)$ respectively mean that the message x is sent through α and y is received through β. The successor operator '.' defines sequences of channels. The agent $\alpha.P$ means α is used to send or receive messages, and then the behaviour is P. Also, $\alpha.\beta$ denotes a sequence of two channels leading to the termination. The 'new' operator allows controlling the scope of an interaction, in a similar way as local variables in functions of programming languages. new(α)P means that α can only be used in the formula of P. Out of the scope of P, the name α refers to *another link*. In particular, the restricted α cannot be used to link P to external agents. The sum of agents relies on the usual choice operator. P can behave as any member of the sum. For instance, $P = \alpha.P_0 + \beta.P_1$ will evolve as P_0 if α is used, and as P_1 if β is triggered. Similarly, the parallel operator '|' represents the composed execution of P_0 and P_1. The focus on interactions of the calculus composes agents by communication channels as presented hereafter in the system evolution.

System Evolution. The π-calculus defines how systems evolve. The basic mechanism is the reaction between two composed agents (parallel execution) along a common channel; one sending a message and the other one receiving. Let's illustrate how this is run.

$$P \overset{def}{=} (\alpha\langle x\rangle.P_0 + \beta\langle y\rangle.P_1) \quad | \quad \alpha(x).0 \tag{2}$$

According to the reaction rule, $\alpha(x).0$ reacts with the first element $\alpha\langle x\rangle.P_0$ of the sum, so that x is passed through α. The second element of the sum is discarded (choice) and the system becomes (we use the intuitive property that A in parallel with 0 is equivalent to A [3]):

$$(P_0) \quad | \quad 0 \quad = \quad P_0 \tag{3}$$

3 T-Compound Model

3.1 T-Compound Formula

Informally, the T-compound is depicted on Fig.1, page 91; the shape of the interaction justifying the name of the composite. The T-compound is formally a 5-tuple $(A, B, P, \alpha, h_\alpha) \in \mathcal{P}_\pi^3 \times \aleph^2$ that verifies structural properties.

Let us consider 3 different agents $(A, B, P) \in \mathcal{P}_\pi^3$ and 2 distinct channels $(\alpha, h_\alpha) \in \aleph^2$ for the communications (A, B) and (A, P) respectively. This configuration is depicted on Fig.2.

The fundamental case where A sends messages to B and P overhears is formally written as follows:

$$T(A,B,P) \overset{def}{=} new(\alpha h_\alpha)(A|B|P) \quad where \quad \begin{cases} A \overset{def}{=} \alpha\langle x\rangle.h_\alpha\langle x\rangle.A \\ B \overset{def}{=} \alpha(x).B \\ P \overset{def}{=} h_\alpha(x).P \end{cases} \tag{4}$$

The definition of the compound means a T is the parallel execution of three agents, each of them playing specific interactions. Agent definitions are recursive to represent the interaction cycle of agents as expected along their lives, that is each agent chooses one of its eligible actions, performs it, and then recovers the capability to choose from its initial action set. The two interaction names used in the T-compound are restricted to these three agents to enforce a proper

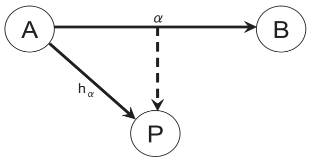

Fig. 2. T-compound Interaction Infrastructure

mechanism of overhearing. Restriction in the π–calculus makes the compound as a coherent interaction entity. It allows the *composition* of T-compounds with other interaction instances as it avoids conflicts among channels (unicity of names and proper scopes).

In the agent formulae, A can first send x through α and *has to* send it through h_α to return to its initial state (otherwise the π-calculus semantics blocks the agent, waiting for triggering h_α). Thus, B receives the message x as a direct interaction through α (the primary intention of A) and P receives a copy of x through h_α representing the overheard event. Fig.2 shows the correspondence with the formula. In the π-calculus, we represent overhearing as a constraint on a direct channel, i.e. the dashed arrow is formally h_α as copy of α.

Note that a channel between two agents does not mean they are actually interacting, since this depends on their intentions or imposed protocols. Instead, *channels represent which interactions are possible* at this level of modeling, *i.e.* the system infrastructure. This is typical for overhearing, where agents may have this ability and use it only in specific situations (possibly ordered by the user).

3.2 Interaction Design Elements

With the T-compound formal model, we now define agent interactions as *two primitives*, and we build a collection of interaction composites relevant in most practical cases. This collection allows constructing a methodology unfolding steps to define generalised MAS interaction infrastructures.

Given $S \subseteq \mathcal{P}_\pi$ an agent set (the system to be modeled), we first define two interaction primitives over the subsets of S, namely $MONO$ and T_0. Then, frequent compositions of these two basic elements allow defining three practical cases named $DUPLEX$, T_1, and T_2. We compiled these interaction elements in Table 1 and detail their syntax and semantics hereafter.

In the following equations, $X \models \phi$ means that the set of agents X satisfies the interaction type ϕ.

$$\forall S_0 \subseteq S: \quad S_0 \models MONO(A,B) \Leftrightarrow \begin{cases} |S_0| = 2, (A,B) \in S_0^2,\ A \neq B, \\ \exists \alpha \in \aleph\ so\ that \\ A = \alpha\langle x \rangle.A\ and \\ B = \alpha(x).B \end{cases} \tag{5}$$

Table 1. Interaction Design Elements

Formal Elements (* primitives)	Meaning
MONO(A,B) *	A sends messages to B.
DUPLEX(A,B)	A and B converse.
T_0(A,B,P) *	A sends messages to B and P hears them.
T_1(A,B,P)	A and B converse and P hears A's talks.
T_2(A,B,P)	A and B converse and P hears both talks.

$$\forall S_0 \subseteq S : \; S_0 \models T_0(A, B, P) \Leftrightarrow \begin{cases} |S_0| = 3, \\ P \in S_0 \; P \neq A, \; P \neq B \\ (A, B, P) \text{ verifies formula (4) for } x \end{cases} \tag{6}$$

MONO represents the fundamental direct interaction, *i.e.* the usual π–calculus channel from one agent to another encapsulated in this interaction compound. T_0 corresponds to the definition (4) and represents the basic case of overhearing.

These two primitives are sufficient to describe MAS interaction infrastructures extended with systematic overhearing. This is due to the fine-grained approach of these interaction compounds. However, MAS interactions shall require coarser-grained elements for practical designs. The simplest and most frequent example is the conversation between two agents, that must be defined with two *MONO*. Therefore, we combine the two primitives into relevant patterns useful for MAS interaction design.

$$\forall S_0 \subseteq S : \quad S_0 \models DUPLEX(A, B) \Leftrightarrow \begin{cases} MONO(A, B) \\ MONO(B, A) \end{cases} \tag{7}$$

$$\forall S_0 \subseteq S : \; S_0 \models T_1(A, B, P) \Leftrightarrow \begin{cases} T_0(A, B, P) \\ MONO(B, A) \end{cases} \tag{8}$$

$$\forall S_0 \subseteq S : \; S_0 \models T_2(A, B, P) \Leftrightarrow \begin{cases} T_0(A, B, P) \\ T_0(B, A, P) \end{cases} \tag{9}$$

DUPLEX describes usual agent conversations. It is built from two symmetrical and complementary *MONO* that define the utterances from A to B and B to A respectively. T_1 corresponds to situations where agent A and B converse and P can only overhear the messages from A. In practice, this case has been demonstrated relevant to reduce the complexity of conversation recognition [4]. In addition, it allows modeling a case typical to overhearing. Human agents A and P are in the same room and A calls B on the phone (in another room). In this scenario and with normal conditions, P can only listen to A. Finally, T_2 refers to the full case of overhearing where P can hear both A and B. This is the most frequent situation when the three agents share the same 'space'.

3.3 MAS Interactions with Our Model

From this collection of interaction elements, we propose two views of interaction infrastructures, namely the system-level $\mathcal{I}_\mathcal{S}$ and agent-centred $\mathcal{I}_\mathcal{A}$ interaction sets. These two tools can be of use in the design of MAS interactions, as they provide points of view orthogonal to the traditional interaction protocols. Our sets aim at a comprehensive description of system interactions, while interaction protocols form a library of scenarii played in part or whole of the system. Interaction design can be thought of as a common exploitation of the three views. In the remainder of this section, we describe our methodology to build these two views from static system specifications (that is, the procedure must be re–run if the specifications change).

$\mathcal{I}_\mathcal{S}$ represents all system interactions in a single view. It is a set of interaction elements from the collection in Table 1 and thus allows explicitly showing overhearing cases. This feature is important so that designers keep track of this interaction pattern and can better avoid unexpected situations leading to potential eavesdropping breaches. Also, it treats direct interactions and overhearing with equal importance, so that system representations are homogeneous. Formally, $\mathcal{I}_\mathcal{S}$ is defined by the following equation, where '*' denotes that the corresponding interaction element can appear zero or several times:

$$\mathcal{I}_\mathcal{S} = \{MONO^*,\ DUPLEX^*,\ T_0^*,\ T_1^*,\ T_2^*\} \tag{10}$$

$\mathcal{I}_\mathcal{A}$ is equivalent to $\mathcal{I}_\mathcal{S}$, though it represents interactions per agent. Each agent appears together with its set of interactions in the system. This view implies various consequences, such as highlighting overloaded agents that perform too many interactions, defining groups and roles, and aligning the infrastructure with interaction protocols (roles can be assigned to agents in their context). The formal description of $\mathcal{I}_\mathcal{A}$ is a π–calculus expression showing the concurrent execution of system agents $(A_i)_{i\leq|S|}$ and their respective interactions $(\sum_{j\leq I_i}\alpha_j)_{i\leq|S|}$.

$$\mathcal{I}_\mathcal{A} = \prod_{i\leq|S|}\sum_{j\leq I_i}\alpha_j.A_i \tag{11}$$

Algorithm 1 describes our methodology that takes in input the raw system interaction specifications I and the empty sets $\mathcal{I}_\mathcal{S}$ and $\mathcal{I}_\mathcal{A}$. Outputs are optimised interaction sets compiled from I, without specification redundancy and improper interaction compounds.

Algorithm 1 Interaction Description Methodology

```
1: I={raw interaction element list ($MONO$, $DUPLEX$, etc.)}, $\mathcal{I}_\mathcal{S}$=∅, $\mathcal{I}_\mathcal{A}$=∅
2: remove_redundant_elements(I)
3: compose_element_types(I)
4: minimize(I)
5: $\mathcal{I}_\mathcal{S}$=I
6: rewrite($\mathcal{I}_\mathcal{S}$,$\mathcal{I}_\mathcal{A}$)
```

The method first removes from the specification set I any obvious redundant interaction element with the procedure *remove_redundant_elements* on line 2. This algorithm is not detailed as it merely compares elements and eliminates repeating ones (note that $DUPLEX$ and T_2 feature a 'symmetry', so $T_2(A, B, P)$ and $T_2(B, A, P)$ are redundant). Then, *compose_element_types* on line 3 combines elements according to the properties (7), (8), and (9) (see Appendix). Finally, *minimize* produces $\mathcal{I}_\mathcal{S}$ by comparing and removing elements that contain common features. Typically, $MONO(A, B)$ and $T_0(A, B, P)$ can be produced by the specification to outline two different aspects of the interactions between A and B. However, $T_0(A, B, P)$ is enough in terms of interaction infrastructure, while the other element is redundant (see Appendix). Our hypothesis is that if an overhearing case has been explicitly defined, it overrides a matching direct interaction. This hypothesis is appropriate since an overhearing situation must be explicitly decided by the designer and it is a stronger constraint on agent behaviours (T_0 above 'includes' the $MONO$ structural information).

Algorithm 2 rewrite(Input: $\mathcal{I}_\mathcal{S}$, Input–Output: $\mathcal{I}_\mathcal{A}$)

```
1: for all Interaction element in $\mathcal{I}_\mathcal{S}$ do
2:     Develop the $\pi$-calculus formula
3:     if compound agent $A$ in $\mathcal{I}_\mathcal{A}$ then
4:         Complete the interaction formula of $A$:
5:             $\mathcal{I}_\mathcal{A}=(\mathcal{I}_\mathcal{A} \setminus \{\sum_i a_i^{old}.A\}) \cup \{\sum_i a_i^{old}.A + \sum_i a_i^{new}.A\}$
6:     else
7:         $\mathcal{I}_\mathcal{A}=\mathcal{I}_\mathcal{A} \cup \{\sum_i a_i.A\}$
8:     end if
9: end for
```

Once $\mathcal{I}_\mathcal{S}$ is finished, the procedure builds $\mathcal{I}_\mathcal{A}$ by calling *rewrite* on line 6. This sub-procedure shown on Algorithm 2 browses $\mathcal{I}_\mathcal{S}$ and develops each encountered interaction element according to its π–calculus formula. Then it extracts the agents contained in the current expression, together with the interactions in which they are involved. If an agent is not part of $\mathcal{I}_\mathcal{A}$, it is added with its interactions (line 7). Otherwise, the formula of this agent in $\mathcal{I}_\mathcal{A}$ is completed with the new interaction links (line 4–5). The procedure terminates and the two aimed interaction sets are completed. The next section now illustrates an execution of this procedure with an example.

4 Example: The Board of Directors

This example models a meeting among the head of a company and its division directors. In other words, our system targets a user and its software advisor agents. In the following, we first suppose all agents can listen to all discussions, and the user is put aside to receive the final advice from the completed debate. We run the methodology for this simple specification. Then, we suppose that the user agent can also send messages to the assistants (to give new orders, *etc.*). We consequently modify the initial scenario and apply once more the methodology to adapt the interaction sets.

4.1 First Specifications

Given n$\geq$3 agents $(A_i)_{i\leq n}$ and the integers i and j, α_{ij} is the communication channel from A_i to A_j. The agent U represents the user interface that compiles the final report from the board and c_{iu} the corresponding channel from agent i. Consequently, the complete system is $S = \{(A_i)_{i\leq n},\ U\}$. Hereafter is the raw

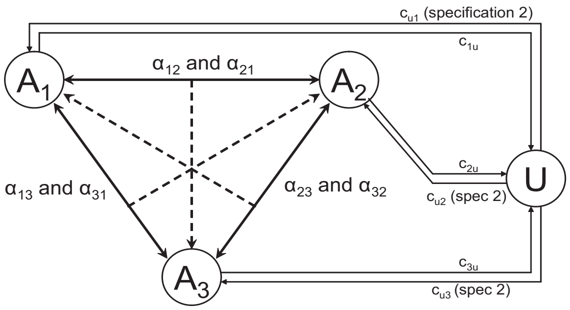

Fig. 3. Board of 3 Directors and the User

interaction set I issued from the specifications for n agents, and we then detail the case $n = 3$ to illustrate the methodology.

$$\mathrm{I} \;=\; \{(T_2(A_i, A_j, A_k))_{i\neq j\neq k}, (MONO(A_i, U))_{i\leq n}\} \tag{12}$$

The first term represents the discussions among the advisors and their ability to overhear conversations in the meeting room, even if they are not active participants. The second term is the final report from each agent to the user interface U. Let us now study in more details the case $n = 3$. Fig.3 shows the interactions that must appear according to the scenario specifications.

The application of the methodology based on the specifications yields the following. The input I is processed, and the output is just $\mathcal{I}_{\mathcal{S}}$=I and the corresponding $\mathcal{I}_{\mathcal{A}}$.

```
1: I=$\{(T_2(A_i, A_j, A_k))_{i\neq j\neq k}, (MONO(A_i, U))_{i\leq 3}\}$, $\mathcal{I}_{\mathcal{S}}=\emptyset$, $\mathcal{I}_{\mathcal{A}}=\emptyset$
2: remove_redundant_elements(I) does not change I (no redundant element)
3: compose_element_types(I) does not change I ($T_2$ and $MONO$ do not combine)
4: minimize(I) does not change I ($T_2$ and $MONO$ involve different agents)
5: $\mathcal{I}_{\mathcal{S}}$=I
6: rewrite($\mathcal{I}_{\mathcal{S}}$,$\mathcal{I}_{\mathcal{A}}$) as described hereafter for the two first iterations.
```

Rewrite procedure: iteration 1 First element of I:

$$\begin{aligned}
&T_2(A_1, A_2, A_3) \;= (\\
&\quad P_1 = \alpha_{12}\langle x_{12}\rangle.\alpha_{13}\langle x_{12}\rangle.A_1 + \alpha_{21}(x_{21}).A_1 \mid\\
&\qquad A_1 \text{ talks to } A_2 \text{ and allows } A_3 \text{ to overhear, or } A_1 \text{ receives from } A_2\\
&\quad P_2 = \alpha_{21}\langle x_{21}\rangle.\alpha_{23}\langle x_{21}\rangle.A_2 + \alpha_{12}(x_{12}).A_2 \mid\\
&\qquad A_2 \text{ talks to } A_1 \text{ and allows } A_3 \text{ to overhear, or } A_2 \text{ receives from } A_1\\
&\quad P_3 = \alpha_{13}(x_{12}).A_3 + \alpha_{23}(x_{21}).A_3\\
&\qquad A_3 \text{ receives overheard messages from } A_1 \text{ or from } A_2)\\
&Consequently: \mathcal{I}_{\mathcal{A}(iteration1)} = (P_1 \mid P_2 \mid P_3)
\end{aligned} \tag{13}$$

Rewrite procedure: iteration 2 Idem with the second element of I:

$$\begin{array}{ll} T_2(A_2, A_3, A_1) = (& Q_1 = \alpha_{23}\langle x_{23}\rangle.\alpha_{21}\langle x_{23}\rangle.A_2 + \alpha_{32}(x_{32}).A_2 \mid \\ & Q_2 = \alpha_{32}\langle x_{32}\rangle.\alpha_{31}\langle x_{32}\rangle.A_3 + \alpha_{23}(x_{23}).A_3 \mid \\ & Q_3 = \alpha_{21}(x_{23}).A_1 + \alpha_{31}(x_{32}).A_1) \\ \mathit{Consequently} : \mathcal{I}_{\mathcal{A}(iteration2)} = (P_1 + Q_3 \mid P_2 + Q_1 \mid P_3 + Q_2) & \end{array} \tag{14}$$

In the end, $\mathcal{I}_\mathcal{S}$ is equal to the raw specifications, and $\mathcal{I}_\mathcal{A}$ contains four agents with all their individual interactions.

$$\mathcal{I}_\mathcal{S} = \{(T_2(A_i, A_j, A_k)_{i\neq j\neq k}, (MONO(A_i, U))_{i\leq 3}\} \quad \mathcal{I}_\mathcal{A} = (\prod_{i=1}^{3} A_i)|cU \tag{15}$$

where $cU = \sum_{i\leq 3} c_{iu}(r_{iu}).cU$ is the set of interactions for the user agent U. We detail hereafter the formula of A_1 only, as the other formulae are similar.

$$\begin{array}{rcll} A_1 & = & (\alpha_{12}\langle x_{12}\rangle.\alpha_{13}\langle x_{12}\rangle.A_1 + \alpha_{13}\langle x_{13}\rangle.\alpha_{12}\langle x_{13}\rangle.A_1 + & //A_1 \text{ talks,} \\ & & & //\text{others overhear} \\ & & \alpha_{21}(x_{21}).A_1 + \alpha_{31}(x_{31}).A_1 + & //\text{One talk to } A_1 \\ & & \alpha_{21}(x_{23}).A_1 + \alpha_{31}(x_{32}).A_1 + & //A_1 \text{ overhears} \\ & & c_{1u}(r_{1u}).A_1) & //A_1 \textit{ reports} \end{array} \tag{16}$$

This example shows how $\mathcal{I}_\mathcal{S}$ represents all system interactions in a compact syntax, and how $\mathcal{I}_\mathcal{A}$ allows handling interactions individually for each agent.

4.2 Second Specifications

In this second case, the specification revision expands the interactions of U (see Fig.3). Our methodology solves the inconsistencies that potentially appear, so that we only need to add the new intended interactions to I. There are two means to extend I and let U be able to engage conversations with assistants. Some designers could add explicit $DUPLEX(U, A_i)$; others would complete the initial reports from assistants to user with symmetrical $MONO(U, A_i)$. Our methodology accepts both cases and computes the same result. We now unfold the procedure twice with the two possible extensions of I, namely J_1 and J_2, and we show it yields the same expected sets.

$$J_1 = I \cup \{(DUPLEX(A_i, U))_{i\leq n}\} \qquad J_2 = I \cup \{(MONO(U, A_i)_{i\leq n}\} \tag{17}$$

In both cases, the execution of the methodology is similar to the previous section and we will just emphasize the differences.

In the case of J_2, elements are composed on line 3 so that the *MONO* added by the new specifications are combined as expected with the *MONO* already representing the reports from assistants to user. Then, the minimization on line 4 does not influence J_2 as there is no compatible item to match. The composition of *MONO* is performed as follows:

```
1: $J_X$={as defined above}, $\mathcal{I_S}=\emptyset$, $\mathcal{I_A}=\emptyset$
2: remove_redundant_elements($J_X$)                    /*no change*/
3: compose_element_types($J_X$)                    /*only modifies $J_2$*/
4: minimize($J_X$)                                 /*only modifies $J_1$*/
5: $\mathcal{I_S}$=$J_X$
6: rewrite($\mathcal{I_S}$,$\mathcal{I_A}$) is given hereafter
```

compose_element_types(J_2):
$J_2 = J_M \cup J_D \cup J_{T_0} \cup J_{T_1} \cup J_{T_2} = J_M \cup \{\emptyset\} \cup \{\emptyset\} \cup \{\emptyset\} \cup J_{T_2}$ (line 1)
Compose steps: Only one iteration affects the output (line 8):
compose(J_M, J_M, J_D)
For $x \in${1,2,3}:
$MONO(U, A_x)$ matches $MONO(A_x, U)$ (*compose* line 14)
So $J_M = J_M \backslash \{MONO(U, A_x)\}$ and $J_M = J_M \backslash \{MONO(A_x, U)\}$, and
$J_D = J_D \cup \{DUPLEX(U, A_x)\}$ (*compose* line 15)
$\rightarrow J_M$ ends empty and J_D has three new elements
Completion $J_2 = \{\emptyset\} \cup J_D \cup \{\emptyset\} \cup \{\emptyset\} \cup J_{T_2}$ (line 11)
$J_2 = \{(T_2(A_i, A_j, A_k))_{i \neq j \neq k}, (DUPLEX(U, A_i))_{i \leq n}\}$

In the case of J_1, the composition has no effect, and modifications are carried out by the following minimization. As $DUPLEX$ are added to the specifications, the initial $MONO$ representing the reports to the user agent are redundant and will be eliminated.

minimize(J_1):
$J_1 = J_M \cup J_D \cup J_{T_0} \cup J_{T_1} \cup J_{T_2} = J_M \cup J_D \cup \{\emptyset\} \cup \{\emptyset\} \cup J_{T_2}$ (line 1)
Minimize step 1 min(J_M, $J_D \cup J_{T_0} \cup J_{T_1} \cup J_{T_2}$) (line 2)
For $x \in${1,2,3}:
$MONO(A_x, U)$ matches $DUPLEX(U, A_x)$ (*min* line 3)
So $J_M = J_M \backslash \{MONO(U, A_x)\}$ (*min* line 4)
$\rightarrow J_M$ ends empty
Minimize step 2 min(J_D, $J_{T_0} \cup J_{T_1} \cup J_{T_2}$) has no effect (line 3)
Minimize step 3 min(J_{T_0}, $J_{T_1} \cup J_{T_2}$) has no effect (line 4)
Minimize step 4 min(J_{T_1}, J_{T_2}) has no effect (line 5)
Completion $J_1 = \{\emptyset\} \cup J_D \cup \{\emptyset\} \cup \{\emptyset\} \cup J_{T_2}$ (line 6)
$J_1 = \{(T_2(A_i, A_j, A_k))_{i \neq j \neq k}, (DUPLEX(U, A_i))_{i \leq n}\}$

In the end, both approaches lead to the same interaction set $\mathcal{I_S}$=J_1=J_2, and consequently the same $\mathcal{I_A}$ as follows, with the detail for agent A_1 ($n = 3$).

$$\mathcal{I_A} = (\prod_{i=1}^{n} A_i)|cU \qquad \text{where } cU = \sum_{i \leq n} (c_{iu}(r_{iu}).cU + c_{ui}\langle r_{ui}\rangle.cU) \tag{18}$$

$$\begin{aligned} A_1^{specification2} &= (A_1^{specification1} + && //\text{formula (16)} \\ & \quad c_{u1}\langle r_{u1}\rangle.A_1) && //A_1 \text{ gets orders} \end{aligned} \tag{19}$$

This example shows the robustness of the methodology to design choices and how incremental design of interactions with $\mathcal{I_S}$ and $\mathcal{I_A}$ could be exploited, especially for open MAS.

5 Work Related to Overhearing Modeling

Gutnik *et al.* proposed the first formal model dedicated to overhearing for conversation recognition [4]. Their representation embodies conversation notions (roles, states, transitions, speech acts, etc.) and the practical exploitation for their issue of identification. Although they propose a 'comprehensive formal model of the general problem' of overhearing, this first attempt is specialised to a peculiar aspect. Our model aims at describing MAS infrastructures with traditional and overhearing interactions, and it is consequently a complementary approach.

Busetta *et al.* proposed an implementation of overhearing [1]. Albeit this work is not a formal model, it stands close to our proposal. It is a multicast communication among agents in a cooperative group. When taking on a channel identified by a discussion theme, all registered listener agents receive the information. This work shows a conceptual difference between our framework and implementation issues. Our approach requires fine-grained details of interactions, whereas the implementation of Busetta is a single broadcast. Thus, implementing efficiently a model is not trivial, especially in the case of *open* MAS. A corollary of this statement is that our formal model do not scale as the implementation.

6 Conclusion

In this paper, we proposed a formal model of interaction in π-calculus that embodies the recent concept of overhearing, represented here as an interaction composite named the T-compound. The aim of this model is to provide a general description of interactions in MAS, orthogonally to other design issues (agents, environment or organisation). This description is performed by a methodology that compiles two views for the study of MAS interactions. The first representation shows all interactions that can occur in a given system. The second one represents an agent-centred description of all these interactions. These two views of the same system can provide MAS designers with relevant information for analysis and design.

Our current model covers static interactions of MAS. Dynamism is not included yet and we intend to introduce this feature necessary in open systems. It will enact considering agents that have new acquaintances, join or quit dynamically the system, or feature mobility. We also pointed out the scalability of our approach is rather low, considering open or large-scale MAS. Hence, we are working on the agent environment so that overhearing would be relayed through it. In fact, Omicini *et al.* and Mamei *et al.* described two infrastructures to support coordination among agents [9, 8] based on the environment. These approaches do not address explicitly the case of overhearing, but they embody related ideas. Our present endeavours are to study the consequences of such environments on our formal definition, methodology, and the pragmatics (computation and management concerns).

References

1. Busetta, P., Donà, A., Nori, M.: Channelled multicast for group communications. In: Autonomous Agents and Multi-Agent Systems. (2002)
2. Legras, F., Tessier, C.: Lotto: Group formation by overhearing in large teams. In: Autonomous Agents and Multi-Agent Systems. (2003)
3. Milner, R.: Communicating and Mobile Systems: the π-calculus. Cambridge Press (1999)
4. Gutnik, G., Kaminka, G.A.: Towards a formal approach to overhearing: Algorithms for conversation identification. In: Autonomous Agents and Multi-Agent Systems. (2004)
5. Resnick, M.: Learning about life. Artificial Life Journal **1** (1994)
6. Keil, D., Goldin, D.: Modelling indirect interaction in open computational systems. In: WETICE. (2003)
7. Kaminka, G.A., Pynadath, D.V., Tambe, M.: Monitoring teams by overhearing: A multi-agent plan-recognition approach. Journal of Artificial Intelligence Research **17** (2002) 83–135
8. Mamei, M., Zambonelli, F.: Motion coordination in the quake 3 arena environment: a field-based approach. In: Workshop on Environment for Multi-Agent Systems. (2004)
9. Omicini, A., Ricci, A., Viroli, M., Castelfranchi, C., Tummolini, L.: Coordination artifacts: Environment-based coordination for intelligent agents. In: Autonomous Agents and Multi-Agent Systems. (2004)
10. Tummolini, L., Castelfranchi, C., Ricci, A., Viroli, M., Omicini, A.: "Exhibitionists" and "Voyeurs" do it better: A shared environment for flexible coordination with tacit messages. In: Workshop on Environment for Multi-Agent Systems. (2004)
11. Isbister, K., Nakanishi, H., Ishida, T., Nass, C.: Helper agent: Designing an assistant for human-human interaction in a virtual meeting space. In: CHI. (2000)
12. Riecken, D.: M: An architecture of integrated agents. In Bradshaw, J., ed.: Software Agents. AAAI Press (2000) 419
13. Rich, C., Sidner, C.: Collagen: When agents collaborate with people. In: First International Conference on Autonomous Agents. (1997)
14. Esterline, A.C., Rorie, T.: Using the π-calculus to model multiagent systems. In: FAABS. Volume 1871 of LNAI., Springer–Verlag (2001) 164–179
15. Ferber, J.: Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley (1999)

A Algorithm Appendix

Algorithm 3 modifies a set I by combining interaction elements into more complex ones (15 cases) with the sub-procedure *compose* in Algorithm 4.

Algorithm 4 receives three interaction sets I, J, K from the procedure *compose_element_types* in Algorithm 3. Elements of I are matched with elements of J that feature common agents *in reversed order*, and new compounds are produced in K. The procedure applies the composition properties (7), (8), and (9) and solves other to cases such as $T_1(A, B, P)$ and $T_1(B, A, P)$ that, as T_0, breed $T_2(A, B, P)$ in terms of infrastructure (12 more cases).

Algorithm 3 compose_element_types(Input–Output: interaction set I)

```
1: I=$I_M \cup I_D \cup I_{T_0} \cup I_{T_1} \cup I_{T_2}$
2: for $(I_1,I_2) \in \{(I_{T_x},I_{T_x}),(I_M,I_{T_2}),(I_D,I_{T_2}),(I_{T_x},I_{T_{y>x}})\}$, $x \leq 3$ and $y \leq 3$ do
3:    compose($I_1$, $I_2$, $I_{T_2}$)    /*This loop handles rule (9) in compose($I_{T_0}$, $I_{T_0}$, $I_{T_2}$)*/
4: end for
5: for $(I_1,I_2) \in \{(I_M,I_{T_{x\leq 1}}),(I_D,I_{T_{x\leq 1}})\}$ do
6:    compose($I_1$, $I_2$, $I_{T_1}$)    /*This loop handles rule (8) in compose($I_M$, $I_{T_0}$, $I_{T_1}$)*/
7: end for
8: for $(I_1,I_2) \in \{(I_X,I_X),(I_M,I_D)\}$, $X \in \{M, D\}$ do
9:    compose($I_1$, $I_2$, $I_D$)    /*This loop handles rule (7) in compose($I_M$, $I_M$, $I_D$)*/
10: end for
11: I=$I_M \cup I_D \cup I_{T_0} \cup I_{T_1} \cup I_{T_2}$
```

Algorithm 4 compose(Input–Output: interaction sets I,J,K)

```
1: for all $X \in I$ do
2:    for all $Y \in J$ do
3:       if $\exists$ agents (A,B,P) so that $X(A,B,P)$ and $Y(B,A,P)$ exist then
4:          I=I$\setminus\{X\}$;  J=J$\setminus\{Y\}$;  K=K$\cup\{T_2(A,B,P)\}$;  Break the loop
5:       end if
6:       if $\exists$ agents (A,B,P) so that $(X(A,B)$ and $Y(B,A,P))$ exist then
7:          I=I$\setminus\{X\}$;  J=J$\setminus\{Y\}$;
8:          if $Y$ is $T_2$ then
9:             K=K$\cup\{T_2(A,B,P)\}$;  Break the loop
10:         else
11:            K=K$\cup\{T_1(A,B,P)\}$;  Break the loop
12:         end if
13:      end if
14:      if $\exists$ agents (A,B) so that $X(A,B)$ and $Y(B,A)$ exist then
15:         I=I$\setminus\{X\}$;  J=J$\setminus\{Y\}$;  K=K$\cup\{DUPLEX(A,B)\}$;  Break the loop
16:      end if
17:   end for
18: end for
```

Algorithm 5 minimize(Input–Output: interaction set I)

```
1: I=$I_M \cup I_D \cup I_{T_0} \cup I_{T_1} \cup I_{T_2}$
2: min($I_M$, $I_D \cup I_{T_0} \cup I_{T_1} \cup I_{T_2}$)    /*Line 2–5 minimize each subset
3: min($I_D$, $I_{T_0} \cup I_{T_1} \cup I_{T_2}$)              relative to subsets of
4: min($I_{T_0}$, $I_{T_1} \cup I_{T_2}$)                        more complex interaction
5: min($I_{T_1}$, $I_{T_2}$)                                        elements*/
6: I=$I_M \cup I_D \cup I_{T_0} \cup I_{T_1} \cup I_{T_2}$
```

Algorithm 6 min(Input–Output: interaction set I, Input: set list J=$\{(J_i)_{i\leq n}\}$

```
1: for all $X \in I$ do
2:   for all $Y$ in a set of $J$ do
3:     if $\exists$ agents (A,B,P) so that $(X(A,B)$ and $Y(A,B))$ or $(X(A,B)$ and
       $Y(A,B,P))$ or $(X(A,B,P)$ and $Y(A,B,P))$ exist then
4:       I=I$\setminus\{X\}$
5:       if $X$ is $DUPLEX(A,B)$ and $Y$ is $T_0(A,B,P)$ then
6:         $J_0$=$J_0\setminus\{T_0(A,B,P)\}$   /*corresponds to $T_0$ interactions in that case*/
7:         $J_1$=$J_1\cup\{T_1(A,B,P)\}$   /*corresponds to $T_1$ interactions in that case*/
8:       end if
9:       Break the loop
10:    end if
11:  end for
12: end for
```

Algorithm 5 modifies a set I by matching interaction elements and keeping only the most constraining ones. For example, $MONO(A,B,P)$ matches $DUPLEX(A,B)$, $T_0(A,B,P)$, $T_1(A,B,P)$, and $T_2(A,B,P)$. As it is less constraining, the procedure will eliminate it if one of the others is found. The minimization is performed by the sub-procedure *min* in Algorithm 6.

Algorithm 6 receives from *minimize* in Algorithm 5 a set I and a set list J, ordered by increasing complexity of interaction elements. The aim is to match elements in I with elements in a set of J that feature common agents *in the same order* (line 3). If a match occurs (we counted 15 cases), the element of I is discarded (line 4) as it is redundant and less complete. In the case of $DUPLEX(A,B)$ and $T_0(A,B,P)$ (line 5), there is an exception. The former is a 'conversation' and the second a single overhearing, so the result of the match is a 'conversation overheard on one side', *i.e.* $T_1(A,B,P)$. In such a case, $T_0(A,B,P)$ is also removed (line 6) and $T_1(A,B,P)$ is created (line 7). The procedure ends with a minimized I, relative to J.