

# MIP-Nets: A Compositional Model of Multiagent Interaction

Sea Ling and Seng Wai Loke

School of Computer Science and Software Engineering  
Monash University, P O Box 197, VIC 3145, Australia  
{sling,swloke}@csse.monash.edu.au

**Abstract.** We show how to translate interaction protocols in AUML to equivalent Petri net specifications. A novelty of our approach is that the Petri nets are modular, clearly separating the protocol from the interaction behaviour of agents induced by their participation in the protocol, yet compositional. Our model can serve at least two purposes in multiagent systems engineering: firstly, specification and verification, and secondly, as a basis for synthesising skeleton code of interacting agents from specifications in the spirit of interaction-oriented programming.

## 1 Introduction

A number of interaction protocols have been proposed by FIPA,<sup>1</sup> whose purpose is to provide software standards for interacting agents. It has been proposed that each interaction protocol can be viewed as a pattern to be used as a “reusable aggregate of processing” [5]. In different problem domains, the pattern becomes a template that can be reused in such a way that the basic interaction and message sequencing remain the same while the agent roles and the message details will be modified to adapt to a different scenario. Such templates can be used by programmers as a guide when building their multiagent system.

However, it often requires clever and careful programming to ensure that agents built do implement a particular protocol. How can we build agents that conform to a given interaction protocol? The challenge grows when the agent has to interact with different parties using different protocols or sub-protocols. An approach is to start from specifications whose correctness have been verified, and then carefully derive (perhaps automatically) code from the specifications. Further debugging, if needed, can then be done as an additional step.

In this paper, we apply the Petri net formalism [14] to model interaction protocols, and show how our model can be used for analysing agent behaviours for correctness and for building agents that interact correctly. The use of Petri nets to model multiagent systems have been done in other recent work such as [7, 15, 12, 17]. However, we believe our approach is novel in the use of compositional Petri nets that are analogous to workflow nets [3], in clearly separating the pattern of interaction from agent behaviour that is induced by the interaction,

---

<sup>1</sup> <http://www.fipa.org/>

and in our aim of engineering multiagent systems from patterns of interaction. By separating out the agent interaction behaviour, we hope to create specifications from which skeleton code can be created for agents that correctly work with the protocol. The skeleton code derived can then be fleshed out with application-specific code.

Our work begins with the simple observation that a given interaction protocol imposes particular constraints on the behaviour of participating agents. Informally but intuitively, an interaction protocol represented in AUMML can be translated into a Petri net modelling the pattern of interaction [12], and each participating agents' (or agent roles') behaviour can be represented by a Petri net. The Petri net for each agent (or agent role) acts as a specification of the aspect of the agent's behaviour that is induced by their involvement in the interaction protocol. Such a specification is in the spirit of the agent skeletons proposed in [16], where a skeleton captures the interaction aspect of an agent (with respect to that protocol). Since Petri nets are used, it is natural to use existing Petri net analysis methods [14] to analyse and verify the correctness of specifications. Because we start from the interaction protocol, we can view our approach as being *interaction-oriented* [16], i.e. a multiagent system is constructed based on the interactions among agents.

The paper is organized as follows. Section 2 provides some background on Petri nets, defines a Petri net model of multiagent interactions and shows how interaction protocols can be represented in such models. Section 3 shows how the resulting Petri net model can be subject to Petri net analysis methods and discusses our approach of developing multiagent applications based on Petri net specifications obtained from AUMML interaction protocol descriptions. Section 4 is the conclusion and future work.

## 2 Modelling Interactions with Petri Nets

### 2.1 Petri Net Preliminaries

This section provides some Petri net preliminaries which are used in subsequent definitions.

Petri nets [14] have been widely used for process specification and verification. A standard Petri net graph consists of *places* (represented by circles), *transitions* (represented by rectangles) and *arcs* (represented by arrows). Given a set of identifiers  $U$ , a *Petri net structure* or simply *net*  $N$  is a triple  $(P, T, F)$ , where  $P \subseteq U$  and  $T \subseteq U$  are non-empty, finite disjoint sets of places and transitions respectively, and  $F \subseteq (P \times T) \cup (T \times P)$ , is the set of arcs (flow relation). The components of a net  $N$  are also denoted by  $P_N$ ,  $T_N$  and  $F_N$ . A *path* of a net is a nonempty sequence  $x_1 x_2 \dots x_k (k \in \mathbb{N})$  of net elements which satisfies  $(x_1, x_2), \dots, (x_{k-1}, x_k) \in F$ . It is *strongly connected* iff for every two net elements  $x, y$  there is a path leading from  $x$  to  $y$ . For some  $x \in P \cup T$ , the set  $\bullet x = \{y \mid (y, x) \in F\}$  is the *preset* of  $x$  and the set  $x^\bullet = \{y \mid (x, y) \in F\}$  is the *postset* of  $x$ .

We can view places as describing the possible local system states or conditions, transitions as events which may modify the system state and arcs simply link a place to a transition or a transition to a place. In other words, an arc linking a place to a transition indicates from which local state can the event occur next, and an arc linking a transition to a place indicates the local state transformation induced by the event occurrence, if any.

At any time, a place contains zero or more *tokens*, drawn as black dots. The state  $M$ , referred to as *marking*, is the distribution of tokens over places. It is a mapping  $M : P \rightarrow \mathbb{N}$ . It will be represented as follows: for example,  $2p_1 + p_2 + 3p_3$  denotes the state with 2 tokens in place  $p_1$ , 1 token in place  $p_2$  and 3 tokens in  $p_3$ . If a place (condition) is marked with a token, the condition is true. A *Petri net system* is a pair  $(N, M_0)$  where  $N$  is a net structure and  $M_0$  is a marking called the initial marking (initial state) of the system.

The dynamic behaviour of a Petri net is controlled by the firing rule. A transition can *fire* or an event can occur if there is at least one token in each of the transition's input places (i.e. the event's pre-conditions are true). This transition is then said to be *enabled*. An enabled transition fires by removing one token from all of its input places (pre-conditions) and depositing one token in each of its output places (post-conditions). This movement of tokens from place(s) to place(s) indicates a change of system states after the occurrence of the event. The notation  $M_1 \xrightarrow{t} M_2$  denotes a transition  $t$  enabled in state  $M_1$  and firing  $t$  in  $M_1$  results in a new state  $M_2$ . If  $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} M_n$  are transition occurrences, then  $\sigma = t_1 t_2 t_3 \dots t_n$  is an occurrence sequence leading from  $M$  to  $M_n$  and we write  $M \xrightarrow{\sigma} M_n$ .  $M_n$  is *reachable* from  $M$  (we write  $M \rightarrow^* M_n$ ) iff there is some firing sequence  $\sigma$  such that  $M \xrightarrow{\sigma} M_n$ .  $[M]$  represents the set of all reachable markings from  $M$ .

Some useful system properties have also been defined for Petri nets. A Petri net system is *live* if, for every marking  $M$  and every transition  $t$ , there is a marking  $M' \in [M]$  which enables  $t$ . The system is *bounded* if for every place  $p$ , there is a natural number  $n$  such that  $M(p) \leq n$  for every reachable marking  $M$ . The system is *1-bounded* or *safe* if  $n = 1$ . Petri net analysis methods [14, 8] have been devised to check for these properties in system modelling.

## 2.2 Agent Nets

The behaviour of an agent can be modelled as a business process consisting of a set of coordinated tasks to be performed from start to finish. Specifically, a class of Petri nets called workflow net (WF-net) [2], used in business process modelling has been adapted for agent behaviour modelling. We call a Petri net that models an agent's behaviour an Agent net (A-net).

**Definition 1.** A Petri net  $A = (P, T, F)$  is an **Agent net (A-net)** iff:

1.  $A$  has two special places *in* and *out*, where  $\bullet in = \emptyset$  and  $out \bullet = \emptyset$ ; and
2. By adding a transition  $t^*$  to  $A$ , the short-circuited net  $(P, T \cup \{t^*\}, F \cup \{(out, t^*), (t^*, in)\})$  must be strongly connected.

In the context of agent systems, the first requirement states that the agent has a *start* state and an *end* state. The second requirement ensures that all places and transitions (or events) contribute to the overall behaviour of the agent. There are no “dangling” places and transitions. In Figure 2(a), there is one agent net on the right and one on the left, describing the behaviours of an initiator agent and a participant agent respectively.

### 2.3 Interaction Protocol Net (IP-net)

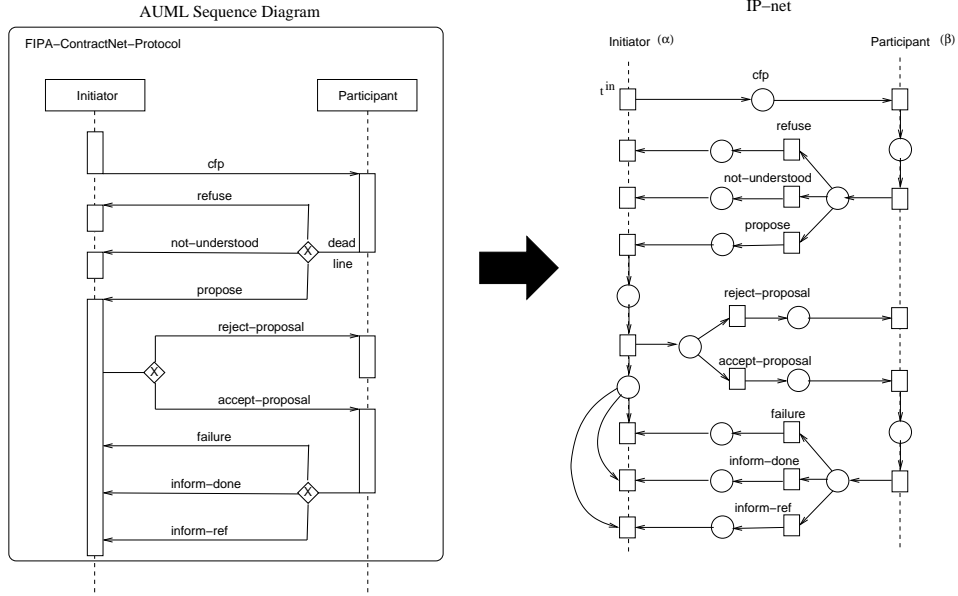
On the left side of Figure 1 is a (chronological) sequence diagram which reproduces the FIPA specification of the Contract Net Interaction Protocol. The dotted box in the upper right corner indicates that this is a template with unbound parameters divided into three categories: role parameters, constraints and communication acts. The communication acts are represented by the arrows labelled with the names of the messages (instead of object-oriented style events in UML). In the protocol, a call-for-proposal (**cfp**) is sent by the initiator (the manager) to the participant (the contractor). Before reaching some deadline, the participant can either **refuse**, **claim not-understood** or **propose** by generating a proposal to perform a task. If a proposal is received by the initiator, the initiator can either **reject-proposal** or **accept-proposal**. After the proposal is accepted, the participant will perform the task till the end when another one of three message (**failure**, **inform-done** or **inform-ref**) has to be sent. It should be noted that for each agent, there are a number of vertical activation bars in the sequence diagram. Each bar represents a different agent role or a different processing thread of the agent. Intuitively, the separation of these bars allows an agent to have different lifelines. For example, after sending **cfp** at the end of a vertical bar, the initiator starts a new role or lifeline to accept a return message (**refuse**, **not-understood** or **propose**) from the participant. For the participant, the reception of **cfp** must be followed by the sending of one message at the end of the deadline on the same lifeline because these activities are constrained to occur on the same vertical bar. A protocol description in AUML can be translated to a Petri net model, as shown in Figure 1, called an Interaction Protocol net (IP-net), defined below. The translation process is similar to the one proposed in [12].

**Definition 2.** A Petri net  $IP = (P, T, F)$  is an **Interaction Protocol net (IP-net)** if and only if:

1.  $IP$  has a special transition  $t^{in} \in T$  called the protocol input transition, such that  $\bullet t^{in} = \emptyset$ ;
2.  $IP$  has a special set of transitions  $T^{out}$ , such that for each  $t \in T^{out}$ ,  $t^\bullet = \emptyset$  and  $t^{in} \notin T^{out}$ ;
3. there are two disjoint classes of transitions, namely  $T^\alpha \subset T$  and  $T^\beta \subset T$ , such that  $t^{in} \in T^\alpha$  and  $T^{out} \subseteq T^\alpha \cup T^\beta$

In the definition, the protocol input transition  $t^{in}$  effectively starts a protocol with the view that every protocol is activated by firing a single transition  $t^{in}$

first. There are no other net elements before  $t^{in}$ . The IP-net terminates with a set of transitions  $T^{out}$  with no other net elements after each  $t$  in the set. In each IP-net, we can also distinguish two disjoint sets of transitions or events. Each set is related to one of the two agents ( $\alpha$  and  $\beta$ ) involved in the protocol. In Figure 1, intuitively, transitions in  $T^\alpha$  and  $T^\beta$  are to be activated by the respective agents ( $\alpha$  and  $\beta$ ) through synchronisation. Note that  $t^{in}$  must be one of the initiator ( $\alpha$ ) agent, while the transitions in  $T^{out}$  can terminate in either  $\alpha$  agent or  $\beta$  agent. Examples of  $T^{out}$  in the IP-net are those transitions on the dotted lines with no outgoing arcs.



**Fig. 1.** The Contract Net Protocol in AUML and in Petri net

In the protocol, an agent can have multiple lifelines or processing threads denoted by separate vertical bars. Messages can be sent and received within the same thread or in different threads independently according to the specification. We model such independence by having individual send and receive transitions in the IP-net. The initiator sends *cfp* via  $t^{in}$  independently of other events. However, if a proposal is received from the participant, the initiator must send *reject-proposal* or *accept-proposal* and wait to receive the message of *failure*, *inform-done* and *inform-ref* in the same thread (within the same vertical bar). This is the interpretation of the interaction protocol and it is reflected in the IP-net by enforcing these initiator transitions to be on the same processing thread.

## 2.4 Multiagent Interaction Protocol Net (MIP-net)

A multiagent system consists of a number of agents and a number of interaction protocols. Every pair of agents communicate with each other via one interaction protocol. This can be modelled by a Multiagent Interaction Protocol Net (MIP-net) which is a combination of A-nets and IP-nets using some synchronising elements. The following MIP-net definition is adapted from the definition of Interorganizational Workflow (IOWF) [3]. It should be noted that IOWF essentially combines several WF-nets describing processes from different organizations. In our work, we treat agents as individual processes and combine them with their interaction protocols to form a MIP-net.

**Definition 3.** *A Multiagent Interaction Protocol Net (MIP-net) is a tuple  $MIP = (A_1, A_2, \dots, A_n, IP_1, IP_2, \dots, IP_k, T_{SC}, SC)$ , such that:*

1.  $n, k \in \mathbb{N}$ , where  $n$  is the number of A-nets and  $k$  is the number of IP-nets and  $n - 1 \leq k \leq n(n - 1)/2$ ;
2. for each  $i \in \{1, \dots, n\}$  :  $A_i$  is an A-net with a start place  $in_{A_i}$  and an end place  $out_{A_i}$ ;
3. for each  $i \in \{1, \dots, k\}$  :  $IP_i$  is an IP-net with a protocol input transition  $t_{IP_i}^{in}$  and the transition sets  $T_{IP_i}^\alpha$  and  $T_{IP_i}^\beta$ ;
4.  $T_{SC}$  is the set of synchronous communication elements (fusion sets);
5.  $SC \subseteq T_{SC} \times T^\circ \times T^\square$  is the synchronous communication relationship, where  $T^\square = \bigcup_{i \in \{1, \dots, n\}} T_{A_i}$ ,  $T^A = \bigcup_{i \in \{1, \dots, k\}} T_{IP_i}^\alpha$ ,  $T^B = \bigcup_{i \in \{1, \dots, k\}} T_{IP_i}^\beta$ ,  $T^\circ = T^A \cup T^B$ ;
6. for all  $t \in T_{SC}$ ,  $\{(t', x, y) \in SC \mid t' = t\}$  is a singleton;
7. for all  $(t_1, x_1, y_1), (t_2, x_2, y_2) \in SC$ : if  $t_1 \neq t_2$ , then  $x_1 \neq x_2 \wedge y_1 \neq y_2$ .
8. for all  $(t_1, x_1, y_1), (t_2, x_2, y_2) \in SC$ : if  $y_1$  and  $y_2$  are from the same A-net, then  $x_1 \in T_{IP}^\alpha \Rightarrow x_2 \notin T_{IP}^\beta$  for a particular IP-net  $IP$ .

Requirement 1 states the relationship between the number of protocols and the number of agents in the multiagent system, assuming that all agents must take part in communication with at least one other agent in the system. Requirements 2 and 3 defines all the A-nets and the IP-nets respectively.

A transition named  $t_{sc}$  in the set  $T_{SC}$  is the result of synchronising or “fusing” two transitions - one from the IP-net and one from the A-net - called a synchronous communication element. The relationship is defined by  $SC$  which is a set of triples, each consisting of the synchronous communication element and a pair of “fused” transitions. Requirement 6 and 7 state that for every synchronous communication element, there is only one unique element in  $SC$  and the pair of fused transitions are also unique. In other words, no already fused transition can be involved in other synchronous relationship. Requirement 8 states that every A-net will only synchronise with either the left ( $\alpha$ ) or the right side ( $\beta$ ) of a particular IP-net.

The MIP-net in Figure 2(a) combines two A-nets with the IP-net in Figure 1 by synchronising pairs of appropriate transitions illustrated by the dotted lines. Note that each agent participates only with one side of the protocol. Note also

that we consider only synchronous communications between transitions because it is more natural to assume that a task activated by an agent will immediately and synchronously trigger a corresponding activity (transition) in the protocol.

### 3 Verification of MIP-nets

Like other existing Petri net work on multiagent systems [7, 12, 15, 17], one purpose of modelling multiagent systems in Petri nets is to make full use of the well-established analysis methods proposed for Petri nets. These methods are commonly used to detect the liveness and the boundedness properties of systems modelled [14]. We want to use Petri nets to provide a notion of correctness for multiagent systems and then to be able to verify and analyse multiagent systems.

Since A-nets are essentially modified WF-nets, the correctness of an A-net is defined similar to WF-net correctness, which is called *soundness*. It is defined based on the following behaviour. Informally, the behaviour of an agent modelled by an A-net is **sound** if and only if:

1. the agent is able to complete its tasks starting from *in* state and finishing at *out* state;
2. after completion, there is no more tasks left waiting to be completed in the A-net; and
3. every defined task in the A-net must have a chance to be activated and completed.

It has been shown that workflow soundness is related to liveness and boundedness properties [2]. We can use the same theorem for agent soundness. Briefly, to decide whether  $A = (P, T, F)$  is sound, we build an extended net  $\bar{A}$  by adding an extra transition which connects the *out* place to the *in* place. Then  $\bar{A}$  will be subject to traditional analysis methods for checking liveness and boundedness properties. Effectively, the theorem states that *an agent (or A-net A) is sound if and only if the Petri net system  $(\bar{A}, in)$  is live and bounded*.

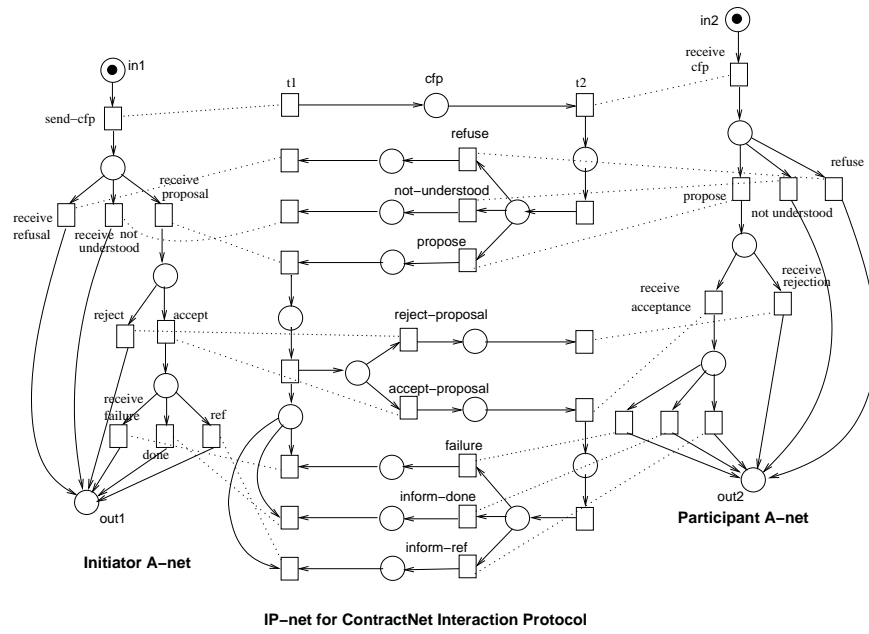
The two A-nets in Figure 2(a) are sound. However, the combination of these agents with an interaction protocol might be subject to protocol synchronization errors. The resulting MIP-net may not be “correct”.

To verify whether a MIP-net is “correct”, we modify the “unfolding” technique proposed for IOWF [3].

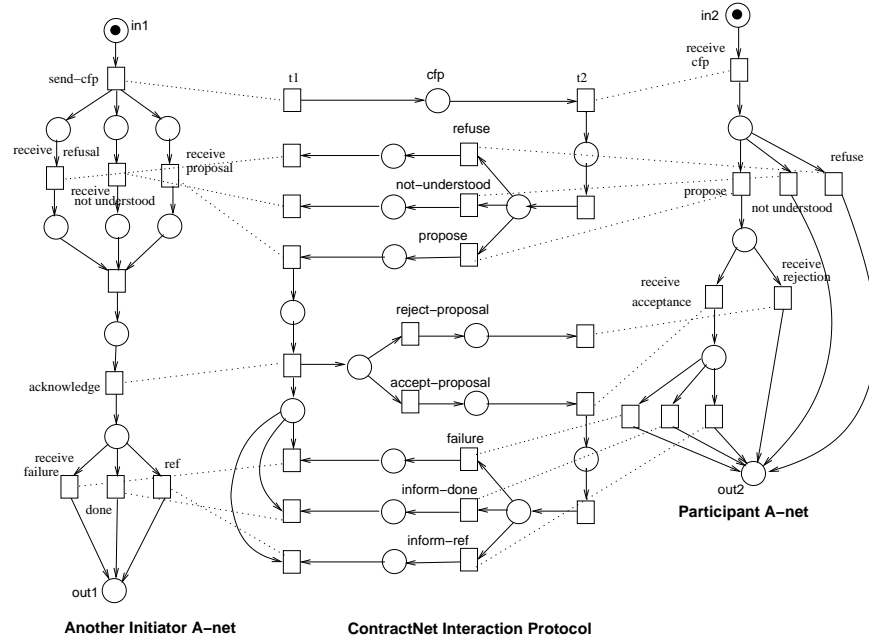
**Definition 4.**

Let  $MIP = (A_1, A_2, \dots, A_n, IP_1, IP_2, \dots, IP_k, T_{SC}, SC)$  be a MIP-net.  $U(MIP) = (P^U, T^U, F^U)$  is the **unfolding** of MIP where:

1.  $P^U = P_{A_1} \cup P_{A_2} \cup \dots \cup P_{A_n} \cup P_{IP_1} \cup P_{IP_2} \cup \dots \cup P_{IP_k} \cup \{i, o\}$ ;
2.  $T^U = r(T^\square \sqcup T^\diamond) \cup T_{SC} \cup (\bigcup_{j \in \{1, \dots, k\}} T_{IP_j} \setminus T^\diamond) \cup \{t_i, t_o\}$ ;
3.  $\{i, o, t_i, t_o\} \cap \{P_{A_1} \cup \dots \cup P_{A_n} \cup P_{IP_1} \cup \dots \cup P_{IP_k} \cup T_{SC} \cup (\bigcup_{j \in \{1, \dots, k\}} T_{IP_j} \setminus T^\diamond)\} = \emptyset$ ;



(a) Original



(b) New

**Fig. 2.** MIP-nets, each formed by combining two A-nets and one IP-net



4.  $r$  is the renaming function:  $r(x) = t_{sc}$  if there is a  $t_{sc} \in T_{SC}$  and  $x, y \in \{T^\square \cup T^\diamond\}$  such that either  $(t_{sc}, x, y) \in SC$  or  $(t_{sc}, y, x) \in SC$ . Otherwise,  $r(x) = x$ ;
5.  $F^\square = F_{A_1} \cup \dots \cup F_{A_n} \cup F_{IP_1} \cup \dots \cup F_{IP_k}$ .  
 $F' = F^\square \cup \{(i, t_i), (t_o, o)\} \cup \{(t_i, in_{A_j}) \mid j \in \{1, \dots, n\}\} \cup \{(out_{A_j}, t_o) \mid j \in \{1, \dots, n\}\}$ ; and
6.  $F^U = \{(r(x), r(y)) \mid (x, y) \in F'\}$ .

The unfolding means to create a single, global A-net from the MIP-net by introducing two new transitions ( $t_i$  and  $t_o$ ) and two new places ( $i$  and  $o$ ). Effectively, place  $i$  and place  $o$  becomes the *in* and *out* places respectively of the global A-net. In Definition 4, requirements 1, 2 and 3 introduce the four new net elements. Requirements 4, 5 and 6 state that all the transition pairs undergoing IP-net and A-net synchronisation will be merged and given a new name by using the renaming function. The result of unfolding is a single Petri net which can then be subject to analysis, as mentioned previously. We define the soundness property of MIP-nets similar to IOWF-soundness [3].

**Definition 5.** *A MIP-net is sound if and only if every A-net in the system is sound and the unfolding of the MIP-net is also sound.*

As mentioned earlier, soundness can be verified by using Petri net methods to check for liveness and boundedness properties. After the MIP-net in Figure 2(a) is unfolded, the unfolding is found to be live and bounded. Since the two A-nets are also sound, the MIP-net is therefore sound, according to Definition 5.

So far, we have described a Petri net method to verify multiagent systems in which multiple agents are engaged in multiple interaction protocols. To summarise, the steps are: (1) Construct an A-net for each agent and an IP-net for each interaction protocol. (2) Combine the A-nets and the IP-nets to form a MIP-net. (3) Unfold the MIP-net. (4) Check the soundness of each A-net and of the unfolded MIP-net. If they are sound, then the system is sound.

To illustrate a particular use of the verification technique, we create the scenario that the initiator agent in Figure 2(a) decides to leave the interaction. A new initiator agent intends to take over the role of the previous agent. The result is a new MIP-net as shown in Figure 2(b). It can be shown that the A-nets are sound, but the unfolded MIP-net is not. Therefore, the MIP-net is not sound (Definition 5).

The reason why it is not sound is that the new initiator (on the left) implements its behaviour by creating three parallel threads to **receive refusal**, **receive not understood** and **receive proposal**, possibly to work in other systems, prior to joining the current protocol. The current system is no longer sound because the interaction protocol requires the joining initiator agent to accept only one of the three messages (a choice of three). To make the system sound, the new agent will have to adapt by changing its behaviour and therefore, needs to be re-programmed.

Given an interaction protocol, it is interesting to note that we can construct *minimal* A-nets to satisfy the protocol. The A-nets in Figure 2(a) are minimal (and have been derived by observing the structure of the IP-net). In other words, they provide the minimal number of receive/send transitions to synchronise correctly with the IP-net in Figure 1 and still ensure system soundness. From the IP-net, the initiator is expected to have a transition (**send-cfp**) to synchronise with  $t^{in}$ , followed by three free-choice transitions (**receive refusal**, **receive not understood** and **receive proposal**) to accept one of the three possible messages. If a proposal is received (the third choice), it must be followed by an acknowledgement (**acknowledge**) and another one of three free-choice transitions (**receive failure**, **done** and **ref**) on the same lifeline. This is exactly as implemented for the initiator A-net and can be considered as the basis for the skeleton code of the initiator agent. Note that the A-net can be viewed as representing the agent role, rather than only the individual agent assuming the role. There might be multiple participant agents in a concrete realization of a multiagent systems using the Contract Net protocol, and the participant A-net can be viewed as representing the class of participant agents.

We can modify the behaviour of the agent (and thus the skeleton code) by changing or enhancing the A-net with more transitions and places. Appropriate transformation rules must be followed to ensure that these changes will not affect the soundness of the agent and the entire system. An approach described in [4] for workflows and another describing agent subtyping relationship [10] can be adapted for changes in agent behaviour. In these approaches, the notion of behavioural equivalence and inheritance-preserving transformation rules have been defined. In A-nets and MIP-nets, we also need to preserve the correct order of activities. As an example, the new A-net in Figure 2(b) has not preserved the intended behaviour of the original A-net to ensure MIP-net soundness.

A-nets and IP-nets are also useful when reasoning with protocol enhancements and protocol compositions. One could model an enhancement of the Contract Net protocol with an IP-net and show that it is essentially equivalent (with some notion of bisimilarity) to the basic version. Domain independent enhancements to protocols can be similarly modelled - for example, to increase robustness when communicating over wireless environments. A composition of several protocols can be modelled by composing the associated IP-nets, after which the corresponding minimal A-nets can be derived.

Combining several A-nets and several IP-nets can be very complex, considering the size of the final MIP-net. Complexity issues have also been addressed for WF-nets and IOWF [1]. It has concluded that by restricting the Petri net models to free-choice nets, the analysis will be more efficient and the soundness property can be checked in polynomial time. These results can also be applied to A-nets and MIP-nets.

## 4 Conclusion

Earlier work [13, 9] on modelling multiagent systems using Petri nets exists in the literature. The general idea is to add intelligent elements and knowledge bases to existing formalisms. In [13], a class of object-oriented Petri nets is enhanced with other concepts to model intelligence in order to achieve the notion of agent-oriented programming.

More recent work focusing on agent interactions through interaction protocols has seen a type of Petri net called G-net extended to support agent modelling [17], whose extension is based on the BDI model. In [12], interaction protocols are translated to equivalent Petri net models so that they can be validated and analysed. Poutakidis et al. [15] used a different Petri net translation of the protocols to debug multiagent systems. In [10], the focus is on behavioural substitutability of agents participating in interaction protocols.

Our work on agent interactions has a different emphasis in that we intend to use Petri nets to develop minimal agent specifications using interaction protocols. Such a specification is in the spirit of the agent skeletons proposed in [16], where a skeleton captures the interaction aspect of an agent (with respect to that protocol). However, Singh's work [16] does not use Petri nets or AUML descriptions. We view IP-nets as template components which can be reused in different environments. In order to use or reuse the templates, agents must satisfy the requirements as stipulated in the interaction protocol and must adhere to (at least) the minimal expected behaviour.

In our earlier work [11], we modelled mobile agent itineraries with a type of Petri nets called Agent Itinerary (AI-) nets. An A-net models the interaction aspect of an agent (with respect to an interaction protocol) whereas an AI-net models the mobility aspect of an agent. The different aspects of an agent can be modelled and their consistency checked. The specifications can serve as basis for generating skeleton code for different aspects of agent behaviour (e.g., [6]).

There is no doubt that future work needs to address other issues common in multiagent systems, i.e., the intelligence and autonomous aspects of agent behaviour. The current MIP-net definitions need to be "upgraded" to high-level or coloured Petri nets to encompass true agent characteristics. These upgrades can be inspired from existing work mentioned previously. In addition, we are also working on analysis of non-functional properties such as performance and timing issues and timed commitments, using other advanced Petri nets such as timed Petri nets. Open protocols is also an area of concern where run-time verification of an agent's ability to undertake a protocol might be useful. We also need to extend our work to allow an agent to be engaged in several protocols at the same time, where the agent's interaction behaviour would then be a composition of the A-nets for each protocol engagement - this would also imply a cleaner separation between the concepts of agent and role than we have done here (where we assumed an agent takes up only one role).

## References

1. W. v. Aalst. Structural characterizations of sound workflow nets. Technical report, Computing Science Reports 96/23, Eindhoven University of Technology, Eindhoven, 1996.
2. W. v. Aalst. The application of Petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
3. W. v. Aalst. Interorganizational workflows: An approach based on message sequence charts and Petri nets. *Systems Analysis - Modelling - Simulation*, 34(3):335–67, 1999.
4. W. v. Aalst and T. Basten. Inheritance of workflows: an approach to tackling problems related to change. Technical report, Computing Science Reports 99/06, Eindhoven University of Technology, Eindhoven, 1999.
5. B. Bauer, J. P. Müller, and J. Odell. Agent UML: A formalism for specifying multiagent interaction. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering*, pages 91–103. Springer-Verlay, Berlin, 2001.
6. S. Chachkov and D. Buchs. From formal specifications to ready-to-use software components: The concurrent object oriented petri net approach. In *Proceedings of the 2nd International Conference on Application of Concurrency to System Design (ACSD'01)*, pages 99–110, June 2001.
7. R. Cost, Y. Chen, T. Finin, Y. Labrou, and Y. Peng. Modeling agent conversations with colored petri nets. In *Proc. 9th Int'l Joint Conference on Artificial Intelligence (IJCAI'99)*, 1999.
8. J. Desel and J. Esparza. *Free Choice Petri Nets*. Cambridge University Press, 1995.
9. J. M. Fernandes and O. Belo. Modeling multi-agent systems activities through colored Petri nets: An industrial production system case study. In *16th IASTED Int'l Conference on Applied Informatics (AI'98)*, pages 17–20, 1998.
10. N. Hameurlain. Formal semantics for behavioural substitutability of agent components: Application to interaction protocols. In *2nd Int'l Workshop of Central and Eastern Europe on Multi-Agent Systems (CEEMAS'01)*, pages 131–40, 2002.
11. S. Ling and S. Loke. Advanced petri nets for modelling mobile agent enabled interorganizational workflow. In *Proc. 9th Int'l Conference on Engineering of Computer-Based Systems (ECBS2002)*, pages 245–52, 2002.
12. H. Mazouzi, A. El Fallah-Seghrouchni, and S. Haddad. Open protocol design for complex interactions in multiagent systems. In *1st Int'l Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, pages 517–25, 2002.
13. D. Moldt and F. Wienberg. Multi-agent-systems based on coloured Petri nets. In *18th Int'l Conference on Application and Theory of Petri Nets 1997*, pages 82–101, 1997.
14. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–80, April 1989.
15. D. Poutakidis, L. Padgham, and M. Winikoff. Debugging multi-agent systems using design artifacts: The case of interaction protocols. In *Proc. 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002)*, 2002.
16. M. Singh. Synthesizing coordination requirements for heterogeneous autonomous agents. *Autonomous Agents and Multi-Agent Systems*, 3(2):107–132, June 2000.
17. H. Xu and S. M. Shatz. A framework for modeling agent-oriented software. In *21st Int'l Conference on Distributed Computing Systems (ICDCS-21)*, pages 57–64, 2001.