

A multi-facet approach to dynamic agent systems

Marek A. Bednarczyk, Wiesław Pawłowski

Institute of Computer Science, Polish Academy of Sciences, Gdańsk, Poland

Institute of Informatics, University of Gdańsk, Poland

m.bednarczyk@ipipan.gda.pl, w.pawlowski@ipipan.gda.pl

Luca Bernardinello, Lucia Pomello

Dipartimento di Informatica, Sistemistica e Comunicazione

Università degli Studi di Milano Bicocca, Milano, Italy

bernardinello@disco.unimib.it, pomello@disco.unimib.it

Tomasz Borzyszkowski

Institute of Informatics, University of Gdańsk, Poland

t.borzyszkowski@univ.gda.pl

Abstract. We are concerned with the problem of defining a complex, hybrid, agent based discrete system in a modular way. The modularity results from looking at the system from a number of different perspectives, each dealing with a specific aspect of the system. As a solution a synchronization operator is proposed which glues *agent aware systems* on shared agents and transitions. The construction turns out to be a categorical product. We also show that a logic to talk about the temporal and the structural properties of the product can be obtained by gluing suitable logical frameworks from the components.

Keywords: dynamic agents; synchronization; categorical semantics; models of concurrent, distributed and mobile computing, system specification.

1. Introduction

We are interested in complex dynamic systems composed of agents. We assume that the distinguishing feature of agents is their identity, which allows us to keep track of individual agents during the dynamic evolution of the system.

We would like to build such systems in a modular way. The modularity should result from looking at the system from a number of different perspectives, each dealing with a specific aspect of the system. It is expected that every such *view* should be easier to grasp. The behavior of the entire system should then be recovered by combining all these specific views. The above is of particular interest when the systems are *agent based*.

The paper presents a solution to the problem of modular system construction based on a simple generalization of a *synchronous product* of automata. We start with a running example of a flip-flop message exchange system in which *messages* flow around, are sent by and delivered to *communicators*. The system is described from two perspectives, called *communication view* and *logistic view*, respectively. The communication view describes how a collection of agents representing the messages and the communicators is distributed among physical locations. The primary concern of this view is to capture the agents' mobility. The logistic view is intended to capture the process of addressing the messages. This is done by saying who sends a message to whom. Clearly, there are activities restricted to just one dimension, e.g., moving the messages from one physical location to another. Others, like delivery of a message to the authorized agent require cooperation in both perspectives. This intuitive idea is further explained in Sec. 2.

Sec. 3 provides a formalization of the abstract model called *agent aware systems*. In Sec. 4 we show how a family of agent aware systems can be composed by synchronizing on common agents and transitions.

In Sec. 5 a more complex example presenting public key cryptography is considered. In this case the complete system is presented via five perspectives. We also demonstrate on the basis of this example that it is possible to glue together logics which express the properties of individual views to obtain a logic with which the properties of the complete system can be captured. The paper is concluded with discussion of future work.

Acknowledgments. The authors would like to thank the referees for insightful remarks.

2. Example — Flip-flop message exchange system

The first example is fairly simple. We intend to model a system of *message* exchange between *communicators*. Thus, it is natural that there are at least two sets of agents involved: *Comm*, the set of communicators, and *Mess*, the set of messages.

The basic properties of the system that we want to model are the following.

1. There is some infrastructure that makes the messages flow around.
2. Each message carries the information identifying its sender and its intended receiver.
3. Only the intended receiver may receive the message
4. Upon receiving the message the receiver may send it back to the sender.

From the above informal description it should be clear that there are at least *two* dimensions involved. One, let us call it the *communication view*, describes how the messages are sent and received by communicators, and how they physically move between locations. The other, let us call it the *logistic view*, deals with message addressing.

Some activities of the system may involve just one dimension, e.g., moving messages between locations. Certain actions clearly involve synchronized multi-dimensional cooperation, e.g., the physical act of delivering a message to the legal addressee.

In the rest of the section the two dimensions are described in a more rigorous way.

2.1. The communication view

To describe the communication view of the system the formalism of *Petri hypernets*, see [3, 4], is used. Hypernets offer a framework suitable for modeling dynamic agents operating in hierarchically structured environments. In hypernets an agent can manipulate other agents as resources while, perhaps, being manipulated by other agent at the same time.

Like in ambients, see [6], and unlike in other models of mobility based on Petri nets, e.g., [9], the hierarchical structure of hypernets is dynamic. The hierarchy changes when two agents exchange some of their resources, i.e., other agents.

Hypernets are used here as a visual formalism which naturally captures issues related to the mobility of agents. We do keep in mind the existence of natural logical formalisms to talk about the structural as well as the dynamic properties of a system of agents represented as a hypernet, see [5].

A simple¹ architecture of the communication layer of our flip-flop message exchange example is depicted on the left of Fig. 1.

The transitions (boxes) and locations (circles) on Fig. 1 are either white or grey. The grey locations may store the messages. The grey transitions, all labeled² `pass_on`, are responsible for moving the messages around in a circular fashion. The white circles are locations in which the communicators reside.

The nature of white transitions is more complex, and their true structure and intended meaning is better explained on the right of Fig. 1. What we want to model here are two processes. One concerns the delivery of a message from the network to a communicator. The other concerns the dispatch of a message by a communicator to the network.

Let us concentrate on the delivery process. Intuitively, the act of delivery should involve the following steps.

1. A message and a recipient agent are chosen in the respective locations.
2. The message is taken out of the location and delivered to the recipient agent.
3. The recipient agent takes care of the message

The above protocol is graphically represented on Fig. 1 as follows. Transition `receive` consists in fact of two synchronized actions. One action selects a recipient agent from the location in which the communicators reside. The other takes a message from the grey location and puts it into the first action. This idea, that the message agent is delivered to the recipient agent is

¹The simplest architecture would be obtained by assuming that all messages are living in a single pool, available to all the communicators all the time.

²With addition of labeling we extend the formalism introduced in [3] to ease the modeling effort. This extension does not, however, increase the expressive power of hypernets.

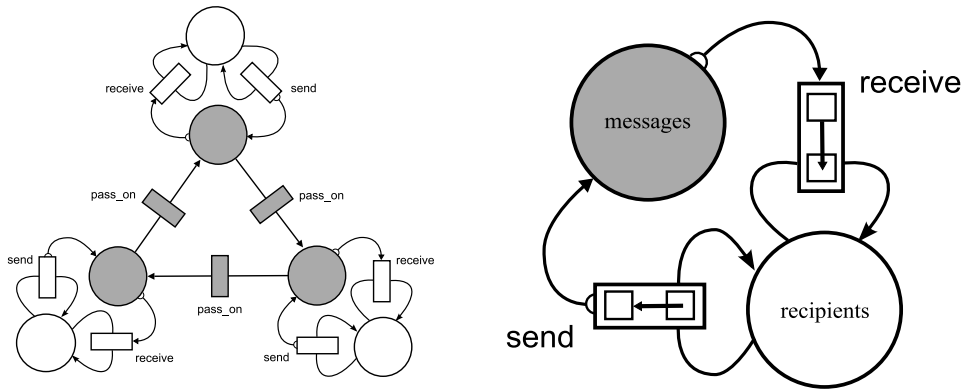


Figure 1. The communication infrastructure and a close-up of the message-receiver

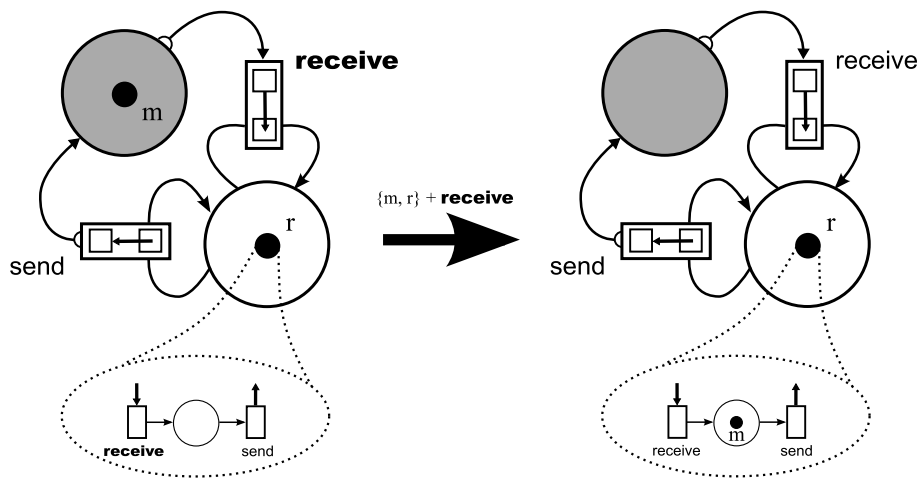


Figure 2. An example of a transaction and the result of its firing

visualized here as an arrow that leaves the border of the top box, and enters the interior of the box at the bottom.

For the above protocol to work the recipient agent should also be prepared to take care of the message. An agent r of that kind residing in the white location is depicted on the left side of Fig. 2. It contains a transition labeled *receive* with an input arrow leading from nowhere — this is a way to say that the transition expects an input from the *current supervisor* of the agent, the communication layer in this case. Whence, it is possible to *fire* the *receive* transition in all the agents involved, and thus moving the agents between locations. The result is depicted in the right side of Fig. 2. Note that the message m has been taken away from the communication layer, and ended inside the agent r .

A detailed presentation of the formalism of hypernets can be found in [3, 4], although the graphical notation is somewhat changed. The important thing now is that the semantics of hypernets given in [3, 4] can be given a more abstract presentation. This is achieved by

defining three sets: A^κ , S^κ and T^κ — the set of agents, the set of states and the set of actions of the communication view, respectively.

The agents in the communication view are grouped in three classes.

- $A^\kappa = \{ \text{layer} \} \cup \text{Comm} \cup \text{Mess}$.

Above, *layer* stands for the agent describing the communication layer. Its Petri net-like structure is explained on the left of Fig. 1.

In this view each communicator agent $c \in \text{Comm}$ has the structure of the agent r described on Fig. 2. Thus, each communicator contains its own individual location to store the received messages. The message agents have no structure. We do assume, however, that one can distinguish them.

Let the set $L = L^c \cup L^m$ be the set of all locations in the hypernet. It is the union of (disjoint) sets L^c and L^m of locations dedicated to store communicator agents and message agents, respectively. In our case, see Fig. 1, the communication layer is very simple, so the following notation is used in the sequel.

$\ell_0^c, \ell_1^c, \ell_2^c$ — the locations of the communication layer in which the communicators can be stored. Note that there are no means for them to move between these locations.

$\ell_0^m, \ell_1^m, \ell_2^m$ — the locations of the communication layer to store the messages

ℓ_c^m — one location for each communicator agent c in Comm in which it can temporarily store its private messages.

Thus, $L^c = \{ \ell_0^c, \ell_1^c, \ell_2^c \}$ and $L^m = \{ \ell_0^m, \ell_1^m, \ell_2^m \} \cup \{ \ell_c^m \mid c \in \text{Comm} \}$.

Then the *state* of the system represented by the hypernet, called *hypermarking* in [3], is uniquely characterized by mapping the agents to locations.

- $S^\kappa = (\text{Mess} \rightarrow L^m) \times (\text{Comm} \rightarrow L^c)$

We use $@$, with some decorations perhaps, to range over the states of a hypernet. The infix notation $a @ \ell$ rather than $@ a = \ell$ is used. If $a @ \ell$ holds we say that the agent a is *at* the location ℓ .

Change of the current state in this perspective may result from performing one of the actions from the set T^κ of actions.

- $T^\kappa = \{ \text{send}, \text{receive}, \text{pass_on} \}$.

Intuitively, *send* is responsible for *sending* a message agent by a communicator agent, *receive* is responsible for *receiving* a message agent by a communicator agent, and *pass_on* is responsible for moving a message agent around the ring. These intuitions are formally captured by the next-state function δ^κ of the following type.

- $\delta^\kappa : T^\kappa \rightarrow \mathcal{P}(A^\kappa) \rightarrow (S^\kappa \rightarrow S^\kappa)$.

Perhaps the idea is easier to grasped at the informal level while studying Fig. 2. For the sake of completeness an explicit formal definition of δ^κ is provided. δ^κ applied to action receive and set of agents α is a partial function denoted $\delta_{\text{receive},\alpha}^\kappa$. In the sequel we often use conditional expressions like $(\text{cond} ? \text{if true} : \text{otherwise})$, and write \uparrow to denote undefinedness.

$$\delta_{\text{receive},\alpha}^\kappa @ = (\exists \langle m, c, i \rangle \alpha = \{ m, c \} \wedge m @ \ell_i^m \wedge c @ \ell_i^c ? @' : \uparrow)$$

where $m' @' \ell'$ iff $m' = m \wedge \ell' = \ell_c^m$ otherwise $m' @' \ell'$ if $m' @ \ell'$, and $c' @' \ell'$ iff $c' @ \ell'$.

$$\delta_{\text{send},\alpha}^\kappa @ = (\exists \langle m, c \rangle \alpha = \{ m, c \} \wedge m @ \ell_c^m ? @' : \uparrow)$$

where $m' @' \ell'$ iff $m' = m \wedge \ell' = \ell_i^m \wedge c @ \ell_i^c$ otherwise $m' @' \ell'$ iff $m' @ \ell'$, and $c' @' \ell'$ iff $c' @ \ell'$.

$$\delta_{\text{pass.on},\alpha}^\kappa @ = (\exists \langle m, i \rangle \alpha = \{ m \} \wedge m @ \ell_i^m ? @' : \uparrow)$$

where $m' @' \ell'$ iff $m' = m \wedge \ell' = \ell_{(i+1) \bmod 3}^m$ otherwise $m' @' \ell'$ iff $m' @ \ell'$, and $c' @' \ell'$ iff $c' @ \ell'$.

Let us call $S^\kappa = \langle S^\kappa, A^\kappa, T^\kappa, \delta^\kappa \rangle$ the *communication view*, or *communication perspective* of the system.

2.2. The logistic view

The logistic view is fully defined by the sets A^λ , S^λ and T^λ of logistic agents, states and actions, respectively, and the transition function δ^λ .

The logistic perspective talks about *message* and *communicator* agents.

- $A^\lambda = \text{Comm} \cup \text{Mess}$.

From the logistic point of view the *state* of the system is captured by labeling each message with the name of its sender and receiver. We let $@$ range over S^λ .

- $S^\lambda = \text{Mess} \rightarrow \text{Comm} \times \text{Comm}$

Here, $@ m = (c_1, c_2)$ captures that in state $@$ message m is being sent by agent c_1 to agent c_2 .

Change of the current state may result from performing one of the following actions.

- $T^\lambda = \{ \text{send}, \text{receive} \}$

The act of receiving a message m should involve the message and a communicator c . Such action may only be performed in states in which the the communicator c is the valid addressee of m . This is captured by the following clause.

$$\delta_{\text{receive},\alpha}^\lambda @ = (\exists \langle m, c, s \rangle \alpha = \{ m, c \} \wedge @ m = (s, c) ? @ : \uparrow) \quad (1)$$

Clearly, transition receive acts as a *filter* which tests if the current state allows for delivery.

Sending a message requires a feasibility test, and also updates the address of the message.

$$\delta_{\text{send},\alpha}^\lambda @ = (\exists \langle m, c, s \rangle \alpha = \{ m, c \} \wedge @ m = (s, c) ? @ [m \mapsto (c, s)] : \uparrow) \quad (2)$$

Notation $@ [m \mapsto (c, s)]$ is used above to denote a function that is like $@$ except that at the argument m it yields (c, s) . Thus, sending the message m by the agent c can take place, provided the message is addressed to c in the current state $@$, and then the new addressee will be the former sender of m , while c is nominated as the sender.

Let us call $S^\lambda = \langle S^\lambda, A^\lambda, T^\lambda, \delta^\lambda \rangle$ the *logistic view*, or *logistic perspective* of the system.

3. Agent aware transition systems

The following definition captures the key features of the flip-flop message exchange example.

Definition 3.1. (Agent aware system)

An *agent aware (transition) system*, abbr.: *aas*, is a tuple $\mathcal{S} = \langle A, T, S, \delta \rangle$ where A , T and S are (finite) sets of *agents*, *transitions* and *states* respectively, and $\delta : T \rightarrow \mathcal{P}(A) \rightarrow (S \rightarrow S)$ is a *transition function*.

A *pointed* or *initialized agent aware system*, abbr.: *paas*, is an agent aware system $\mathcal{S} = \langle A, T, S, \delta \rangle$ together with an *initial state* $s_0 \in S$.

Note that the above definition is a straightforward generalization of the notion of deterministic transition system. Namely, any deterministic transition system $\langle T, S, \delta \rangle$, where $\delta : T \rightarrow S \rightarrow S$, can be identified with an agent aware transition system $\langle \emptyset, T, S, \delta' \rangle$ by letting $\delta'_{t,\emptyset} = \delta_t$.

Definition 3.2. (Morphism)

Given aas's $\mathcal{S} = \langle A, T, S, \delta \rangle$ and $\mathcal{S}' = \langle A', T', S', \delta' \rangle$ such that $A \supseteq A'$ and $T \supseteq T'$ their morphism $\sigma : \mathcal{S} \rightarrow \mathcal{S}'$ is a function $\sigma : S \rightarrow S'$ such that for any $t \in T$, $\alpha \subseteq A$ and $@ \in S$

1. if $t \in T'$ and $\delta_{t,\alpha}@$ is defined then $\delta'_{t,\alpha \cap A'}(\sigma(@)) = \sigma(\delta_{t,\alpha}@)$
2. if $t \notin T'$ and $\delta_{t,\alpha}@$ is defined then $\sigma(@) = \sigma(\delta_{t,\alpha}@)$

Morphisms of pointed systems are required to preserve the initial states as well.

We use the strong interpretation of equality involving the potentially undefined term $\delta'_{t,\alpha \cap A'}(\sigma(@))$, which means that we require it to be defined.

The two conditions in Def. 3.2 explain how a *step* from $@$ to $\delta_{t,\alpha}@$ induced by $t \in T$ and $\alpha \subseteq A$ in the source aas of a morphism is simulated in the target aas. Namely, if t is observable in the target aas, then the transition is preserved, see Def. 3.2.1. If, however, $t \notin T'$, then the morphism glues the beginning and the end states of the transition, see Def. 3.2.2. Similar conditions on morphisms of transition systems, Petri nets, and other models of parallel computations can be found already in [10].

Proposition 3.1. Agent aware transition systems and their morphisms form a category. □

The category of agent aware transition systems is denoted \mathbb{AAS} in the sequel.

4. Synchronizing agent aware systems

We have stopped discussion of the flip-flop message exchange system by defining two views: agent aware system \mathcal{S}^λ captures the logistic perspective, while \mathcal{S}^κ captures the communication perspective.

The question arises: *How does the complete system work?* The idea explained below is to view the behavior of the entire system as a kind of *product* of its views \mathcal{S}^λ and \mathcal{S}^κ . We follow an analogy to the operation of synchronization in CSP [7], and so called *synchronous product* known from transition systems and many other models (see e.g., [8, 11, 2]).

Hence, we say that each view contributes its local state to the state of the complete system. Thus, the states of the complete system are vectors of states, one taken from each view.

The agents and the transitions of the complete system should include the corresponding sets from each view — this assumption follows from the idea that each view contains just those components of the complete system which are relevant from the given perspective. To say this differently — all agents and transitions irrelevant to a given perspective are omitted. For instance, the logistic view includes neither the `pass_on` transition, nor the `layer` agent.

Whence, the set of agents and the set of transitions of the complete system should be unions of the corresponding sets from each view.

Finally, the transition function of the complete system should agree with the transition functions of each view. More specifically, a step induced by a transition t of the complete system should result from a collection of steps in all views containing t . Those states of the vector which come from components not containing t should not be changed by the step.

The following formalization is intended to capture the above intuitions.

Definition 4.1. (Synchronization of agent aware systems)

Consider a collection $\{S_i \mid i \in I\}$ of agent aware systems with $S_i = \langle A_i, T_i, S_i, \delta^i \rangle$. The *synchronous product* of the collection, notation $\prod_{i \in I} S_i$, is defined as $\langle \bigcup_{i \in I} A_i, \bigcup_{i \in I} T_i, \prod_{i \in I} S_i, \Delta \rangle$ with

$$\Delta_{t,\alpha} @ \hat{=} ((\exists k \in I \ t \in T_k \wedge \delta_{t,\alpha \cap A_k}^k @_k = \uparrow) \vee \alpha \neq \bigcup_{i \in I_t} \alpha \cap A_i \ ? \ \uparrow : \langle @'_i \rangle_{i \in I})$$

where I_t denotes the set $\{k \in I \mid t \in T_k\}$, $@'_i = \delta_{t,\alpha \cap A_i}^i @_i$ when $t \in T_i$ and $@'_i = @_i$ when $t \notin T_i$, for any $i \in I$, $t \in \bigcup_{i \in I} T_i$, $\alpha \subseteq \bigcup_{i \in I} A_i$ and $@ = \langle @_i \rangle_{i \in I} \in \prod_{i \in I} S_i$.

The above construction is extended to pointed agent aware systems by taking the vector of initial states as the initial state of the product.

The synchronization of several aas's by means of the product construction defined above gives us an aas. In fact, we can show more.

Theorem 4.1. The synchronous product of a family of agent aware transition systems is a categorical product of the family in \mathbb{AAS} . □

Although the above result is quite elementary, it is encouraging to know that the synchronization operator turns out to be a universal construction. As an immediate consequence one obtains that the construction is associative and commutative (up to a canonical isomorphism).

5. Public key cryptography

Let us demonstrate the usefulness of the proposed approach by discussing a multi-facet description of a public key cryptography system. It turns out that to some extent this example is just an elaboration of the flip-flop message exchange system discussed in Section 1.

Let us first introduce the sets of transition and agent names of interest.

- $A = \text{Comm} \cup \text{Mess} \cup \text{PubKey} \cup \text{SecKey} \cup \{ \text{layer} \}$

There are four classes of agents: *communicators* ranged over by $c : \text{Comm}$, *messages* ranged over by $m : \text{Mess}$, *public keys* ranged over by $pk : \text{PubKey}$, and *secret keys* ranged over by $sk : \text{SecKey}$, respectively. There is also agent layer responsible for describing the communication system.

We assume that the four classes of agents are disjoint.

- $T = \{ \text{receive}, \text{send}, \text{pass_on} \}$.

The complete cryptography system is described below from five perspectives.

5.1. The communication view

The communication perspective is the same as described in Sec. 2.1. Let us keep notation S^k for this view.

Let us recall that in this perspective a message may be received by a communicator whenever it is close. As a consequence of the assumption that only the message and the communicator agents are considered, together with the layer agent, one obtains that no exchange of the keys among communicators is envisaged in this layer.

5.2. New logistic view — messages' encryption

The logistic view of flip-flop message exchange example is modified here to describe the encryption of messages. In the sequel A^* is the free monoid over A , notation ε is used to denote the empty sequence, $x \cdot y$ denotes concatenation of sequences. Let $\text{tail}(x) = y$ and $\text{head}(x) = a$ if $x = a \cdot y$, and undefined otherwise.

- $A^\lambda = \text{Mess} \cup \text{PubKey}$, $T^\lambda = \{ \text{receive}, \text{send} \}$, $S^\lambda = \text{Mess} \rightarrow \text{PubKey}^*$

The state in the logistic view represents how each message is wrapped up, i.e., what public keys have been used to prepare it. Intuitively, for $s \in S^\lambda$, $s(m) = \varepsilon$ if the message m is not encrypted. If $s(m) = k \cdot x$ than one should start reading m by "stripping" the public key k . Of course a communicator will not always be able to "strip" a public key from the message. This aspect will be covered by the next two views. Here, we just describe the *wrapping-up-stripping* mechanism.

- A message m may be received provided the right key is applied.

$$\delta_{\text{receive}, \alpha}^\lambda @ = (\alpha = \{ m, k \} \wedge \text{head}(@m) = k \ ? \ @ [m \mapsto \text{tail}(@m)] : \uparrow)$$

As a result the outer encoding is removed.

- Sending a message is simpler. It may involve encoding.

$$\delta_{\text{send}, \alpha}^\lambda @ = (\alpha = \{ m, k \} \ ? \ @ [m \mapsto k \cdot (@m)] : \uparrow)$$

But does not have to: $\delta_{\text{send}, \{ m \}}^\lambda @ = @$.

5.3. Public keys versus secret keys

This view defines a correspondence between the public and the secret keys.

- $A^S = \text{PubKey} \cup \text{SecKey}$, $T^S = \{\text{receive}\}$, $S^S = \text{PubKey} \rightarrow \text{SecKey}$
Every public key is associated with a unique secret key.
- The process of receiving involves a pair of matching keys.

$$\delta_{\text{receive},\alpha}^S(@) = (\alpha = \{ \text{pk}, \text{sk} \} \wedge @(\text{pk}) = \text{sk} \ ? \ @ : \uparrow)$$

Note that the clauses define δ^S as partial identities on states. Thus, performing a transition in this perspective amounts to verifying that if some key agents are involved then they are dual to each other.

5.4. Epistemic layers

This two views contain information concerning the knowledge of communicators. The state spaces explain who knows which keys, public or secret. Below we shall only present one of the epistemic layers (concerned with secret keys), as the other one can be defined analogously.

- $A^{\mu_s} = \text{Comm} \cup \text{SecKey}$, $T^{\mu_s} = \{\text{receive}\}$, $S^{\mu_s} = \text{Comm} \rightarrow \mathcal{P}(\text{SecKey})$,
- The process of receiving messages involves secret keys

$$\delta_{\text{receive},\alpha}^{\mu_s}(@) = (\alpha = \{ c, k \} \wedge k \in @(c) \cap \text{SecKey} \ ? \ @ : \uparrow)$$

Again, the clause defines δ^{μ_s} as partial identity, so the action here is a guard. The clause above says that receiving an encrypted message requires participation of two agents—a communicator and a secret key, which is known to the communicator.

5.5. Synchronization and its properties

Above, five views of a complex system have been defined. Each aims to cope only with a specific feature of the cooperation between the agents involved. Although the nature of the views is quite different all conform to the very general and abstract formalism of agent aware transition systems. Hence, using the construction from the Definition 4.1, one can compose all of them to obtain equally abstract view of the complete system.

Clearly, the behavior of the complete system is a restriction of the behaviors of its views. As a result some properties enjoyed by a component may not be valid in the synchronous product, and vice versa. This prompts a question: *What are the properties of the complete system?*

Here we want to address more basic issue related to the above question. Namely, it is natural to expect that the properties of each view would be described in a formalism specific to this view. In [5], for instance, a logic was proposed in which two sets of modal operators were introduced: one to deal with the temporal evolution of a hypernet, another to describe the subsumption of some agents by other agents. In order to answer the above question

we have to solve the fundamental problem: *How can we merge (logical) specification formalisms associated to different views to obtain a uniform formalism capable to express the properties of the complete system?*

We have tried, and failed, to define such a merger of logics under assumption that each of the views is described by a logic proposed in [5]. More specifically, we have found it difficult to reuse the modal operators dedicated to the structural properties of agents in each view. Here, on a simple example, we show how such a merger can be achieved under the following assumptions.

- In each view the structural properties of agents are described by means of predicates as atomic formulae. It is assumed here that each predicate is specific to a single view.
- The temporal properties of each view are described in the same temporal formalism.

The merger is then done in a straightforward manner. The logic to express the properties of the synchronization of views would inherit the atomic predicates from each view, as well as the common temporal set of temporal modalities. Instead of a formal presentation we offer here an example of such merger and demonstrate how one can mix descriptions from different views as a result.

Let us first note that an agent aware system $\mathcal{S} = \langle A, T, S, \delta \rangle$ is also a deterministic transition system $\langle S, T \times \mathcal{P}(A), \delta \rangle$ where $\delta : T \times \mathcal{P}(A) \rightarrow (S \rightarrow S)$. Thus, it admits many modal logics to talk about the dynamic evolution of the system. Such logics would include various extensions of LTL, CTL, μ -calculus, possibly capable of expressing various *structural properties* of the agents involved.

For instance [5] investigates how to define such a logic to express:

- The dynamic evolution of hypernets, i.e., the properties of the *current state* and some/all states reachable from it.
- The the structural properties of the agents, i.e., the properties of the *current agent* and some/all agents connected with it.
- Any mixture of the temporal and structural properties.

In terms of the communication view of the public key cryptography case study we can express for example the following structural properties.

- The current agent is a communicator.
- The current agent contains a *sub-agent*.
- The current agent is contained in a communicator *super-agent*.
- Agents denoted m_1 and m_3 are sub-agents of a communicator agent.
- All sub-agents of the current agent will eventually end up as sub-agents of an agent called r .

The last property mixes the temporal evolution of the state with the structural position of agents. In the current paper, for simplicity, we shall only consider structural properties in the form of *atomic statements*.

Interpreted Agent Aware Systems

In what follows we want to model a situation where each *temporal state* of the agent aware system (describing the synchronization of views) has an associated set of *relations on agents*.

Definition 5.1. (Interpreted agent aware system)

An interpreted agent aware system, abbr.: *iaas*, is a 6-tuple: $M = \langle A, T, S, \delta, \Sigma, \xi \rangle$ such that $\langle A, T, S, \delta \rangle$ is an agent aware system, Σ is a *ranked alphabet*, and $\xi : S \rightarrow \text{RelStr}(\Sigma, A)$, where $\text{RelStr}(\Sigma, A)$ denotes the set of all Σ -relational structures having the set of agents A as their carrier.

The function ξ interprets the *temporal states* (elements of S) in terms of relations between agents. The repertoire of relations i.e., their names and arities, stays fixed but the interpretation of the relational symbols may of course change from state to state.

Agent terms and valuations

To define the set of formulae of our logic we shall need a notion of an *agent term*. An agent term over the set of agents A will be either an *agent name* or an *agent variable*:

$$\text{Term}(A) ::= \{ [a] \mid a \in A \} \cup X$$

where X denotes the set of all agent variables.

An interpretation of *agent variables* will be given by a *valuation* $v : \text{Val} = X \rightarrow A$. Each valuation v in a natural way extends to a function $v^\sharp : \text{Term}(A) \rightarrow A$ interpreting the whole set of *agent terms*: $v^\sharp([a]) \hat{=} a$.

Formulae and satisfaction relation

The set of formulae of our logic is defined as follows

$$\varphi ::= \text{false} \mid r(t_1, \dots, t_n) \mid \neg \varphi \mid \varphi \vee \psi \mid \text{E}\varphi\text{U}\psi$$

where $r \in \Sigma$ is an arbitrary n -ary relational symbol and t_1, \dots, t_n are agent terms over A .

$$\begin{array}{lll} s, v \models \text{false} & \text{never} & s, v \models \varphi \vee \psi \quad \text{if } s, v \models \varphi \text{ or } s, v \models \psi \\ s, v \models \neg \varphi & \text{if } s, v \not\models \varphi & s, v \models r(t_1, \dots, t_n) \quad \text{if } \langle v^\sharp(t_1), \dots, v^\sharp(t_n) \rangle \in r_{\xi(s)} \\ s, v \models \text{E}\varphi\text{U}\psi & \text{if there is a path } \pi \text{ from } s \text{ and some } k \text{ such that } \pi_k, v \models \psi, \text{ and } \pi_i, v \models \varphi, \\ & \text{for all } i, 0 < i < k. \text{ Above } \pi_n \text{ denotes the } n\text{-th element of the path } \pi. \end{array}$$

Definition 5.2. (IAAS Morphism)

Given two iaas's $M = \langle A, T, S, \delta, \Sigma, \xi \rangle$ and $M' = \langle A', T', S', \delta', \Sigma', \xi' \rangle$ their morphism $\langle \sigma, \rho \rangle : M \rightarrow M'$ is given by an agent aware system morphism $\sigma : \langle A, T, S, \delta \rangle \rightarrow \langle A', T', S', \delta' \rangle$ and

a ranked alphabet morphism $\rho : \Sigma' \rightarrow \Sigma$ such, that the following diagram commutes (in Set)

$$\begin{array}{ccc}
 S & \xrightarrow{\xi} & \text{RelStr}(\Sigma, A) \\
 \sigma \downarrow & & \downarrow \cdot |_{\rho} \\
 S' & \xrightarrow{\xi'} & \text{RelStr}(\Sigma', A')
 \end{array}$$

where $\cdot |_{\rho} : \text{RelStr}(\Sigma, A) \rightarrow \text{RelStr}(\Sigma', A')$ denotes the usual ρ -reduct operation on relational structures, combined with the obvious carrier restriction.

Proposition 5.1. Interpreted agent aware systems and their morphisms form a category, denoted \mathbb{IAAS} in the sequel. \square

It turns out that the extension of the synchronization operation on agent aware systems, as defined in Section 4, to the case of interpreted systems is quite straightforward.

Definition 5.3. (Synchronisation of iaas's)

Let $\{\mathcal{S}_i \mid i \in I\}$ be a family of interpreted agent aware systems with $\mathcal{S}_i = \langle A_i, T_i, S_i, \delta^i, \Sigma_i, \xi_i \rangle$. Its *synchronous product*, denoted $\prod_{i \in I} \mathcal{S}_i$, is defined as $\langle A, T, S, \Delta, \coprod_{i \in I} \Sigma_i, \xi \rangle$ where

- $\langle A, T, S, \Delta \rangle$ is the product of the family $\{A_i, T_i, S_i, \delta^i \mid i \in I\}$ in the category of agent aware systems \mathbb{AAS} , see Section 4,
- $\coprod_{i \in I} \Sigma_i$ (together with injections $\iota_j : \Sigma_j \rightarrow \coprod_{i \in I} \Sigma_i$, for $j \in I$) is the coproduct of $\{\Sigma_i \mid i \in I\}$ in the category of ranked alphabets, i.e., up to an isomorphism a disjoint union of the families of symbols from Σ_i , for $i \in I$,
- for any $\langle s_i \rangle_{i \in I}$ in $S = \prod_{i \in I} S_i$, and each n -ary symbol $r \in \coprod_{i \in I} \Sigma_i$ the interpretation of r in the $\coprod_{i \in I} \Sigma_i$ -relational structure $\xi(\langle s_i \rangle_{i \in I})$ is given by $r_{\xi(\langle s_i \rangle_{i \in I})} \hat{=} r'_{\xi_k(s_k)} \subseteq A_k^n \subseteq A^n$, where $k \in I$ and $r' \in \Sigma_k$ are the unique index and the unique n -ary relational symbol respectively, such that $\iota_k(r') = r$.

Again, the synchronization of interpreted agent aware systems is a universal operation.

Theorem 5.1. The synchronization of a family of interpreted agent aware systems is a categorical product of the family in \mathbb{IAAS} . \square

Public key cryptography example revisited

Unfortunately, due to the lack of space, we are not able to present the result of composing all the views from Section 5 here. Instead, let us illustrate the construction with an example of a property of the system, which refers to properties coming from the different views. We shall extend the views presented in Section 5 with the appropriate interpretations for the temporal states.

- For the *communication view* let us consider a ranked alphabet containing a binary symbol $\text{SubA}(-, -)$ with the “agent-containment” interpretation given by the open net hierarchy within a hypernet (see Section 2.1).
- In the first *logistic view*, describing encryption of messages let us consider a ranked alphabet containing a binary relational symbol: $\text{LastK}(-, -)$. Its interpretation in a given temporal state $s : \text{Mess} \rightarrow \text{PubKey}^*$ is the set of all pairs $\langle m, \text{pk} \rangle$ such that pk is the first element of the list $s(m)$, whenever the list is non-empty.
- For the view, which establishes the correspondence between public and secret keys, let the ranked alphabet contain a binary symbol: $\text{MatchK}(-, -)$. Its interpretation, for an arbitrary temporal state $s : \text{PubKey} \rightarrow \text{SecKey}$ is the set of all pairs $\langle \text{pk}, \text{sk} \rangle$ such that $s(\text{pk}) = \text{sk}$.
- In the two epistemic views, describing the communicators’ knowledge about keys (public and secret), let the ranked alphabet contain a binary symbol $\text{HasK}(-, -)$, again with the obvious interpretation in any given temporal state of the view.

If, for simplicity, we assume the ranked alphabets of the above views to be disjoint, then, in the logic for the whole system (i.e., synchronization of the views) we can state properties like:

$$\text{SubA}(c, m) \wedge \text{LastK}(m, \text{pk}) \wedge \text{MatchK}(\text{pk}, \text{sk}) \wedge \text{HasK}(c, \text{sk}) \Rightarrow \text{EF}(\neg \text{LastK}(m, \text{pk}))$$

where $\text{EF}\varphi$ stands for $\text{EtrueU}\varphi$. Informally, this property says that whenever a communicator physically possesses a message, and knows a secret key that is “dual” to the last public key the message has been encrypted with, then there is a possible temporal evolution of the system leading to a state where the message has been decrypted using the secret key.

6. Conclusions and future work

We have proposed a synchronization operator that allows us to glue together several agent based dynamic systems, provided they can be recast as *agent aware transition systems*. The construction was shown to be a categorical product. We have argued that the proposed synchronization, here by means of common names of agents and transitions, can be used to enforce cooperation between models of very different nature. In practice, however, one would want to rename actions of one system to allow for synchronization. We plan to follow the ideas laid down in [2] and consider a larger category of morphisms. We hope that in this way the synchronous product considered here would turn also a limit (pullback) in the larger category — as suggested by one of the referees. Interestingly, this forces one to move to semantics in which *true concurrency* is taken seriously, see [2, 1]. The presence of agents in the framework opens quite new interesting areas for investigation.

The categorical studies should also include the problem of synthesis of concrete models, like hypernets, from abstract specifications in terms of agent aware systems.

We have also shown that under certain assumptions one can hope to merge specification formalisms used to describe the properties of components. Initially, our goal was to achieve

such a merger of logics described in [5]. We have found it quite difficult to cope with modal operators introduced in [5] to describe the structural properties of agents in each view. Our example — admittedly, very simple — demonstrates that this can be done when the structural properties are described by means of predicates as atomic formulae. The logics of properties can then be built to discuss the properties of the synchronous product of views by taking as atomic formulae the predicates from all views.

The algorithmic side of model-checking the properties of the resulting logic remains to be investigated.

References

- [1] M. A. Bednarczyk. Limits of modularity. *Fundamenta Informaticae*, 74(2):167–187, 2006.
- [2] M. A. Bednarczyk, L. Bernardinello, B. Caillaud, W. Pawłowski, and L. Pomello. Modular system development with pullbacks. In W. van der Aalst and E. Best, editors, *24th International Conference on Applications and Theory of Petri Nets, Eindhoven, The Netherlands, June 2003*, volume 2679 of LNCS, pages 140–160. Springer-Verlag, 2003.
- [3] M. A. Bednarczyk, L. Bernardinello, W. Pawłowski, and L. Pomello. Modelling mobility with Petri hypernets. In J. L. Fiadeiro, P. D. Mosses, and F. Orejas, editors, *Recent Trends in Algebraic Development Techniques. 17th International Workshop WADT 2004, Barcelona, Spain, March 27-30, 2004. Revised Selected Papers*, volume 3423 of LNCS, pages 28–44. Springer-Verlag, 2005.
- [4] M. A. Bednarczyk, L. Bernardinello, W. Pawłowski, and L. Pomello. From Petri hypernets to 1-safe nets. In D. Moldt, editor, *Proceedings of the 4th International Workshop on Modelling of Objects, Components, and Agents, MOCA'06, Turku, Finland, 2006*.
- [5] M. A. Bednarczyk, W. Jamroga, and W. Pawłowski. Expressing and verifying temporal and structural properties of mobile agents. *Fundamenta Informaticae*, 72(1-3):51–63, 2006.
- [6] L. Cardelli and A. D. Gordon. Mobile ambients. In *Foundations of Software Science and Computation Structures: First International Conference, FOSSACS '98*. Springer-Verlag, Berlin Germany, 1998.
- [7] C. A. R. Hoare. *Communicating sequential processes*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1985.
- [8] A. Mazurkiewicz. Semantics of concurrent systems: a modular fixed point trace approach. Technical Report 84-19, Institute of Applied Mathematics and Computer Science, University of Leiden, Leiden, 1984.
- [9] R. Valk. Petri nets as token objects: An introduction to elementary object nets. In W. van der Aalst and E. Best, editors, *Applications and Theory of Petri Nets 1998, Proceedings*, volume 1420 of LNCS, pages 1–25. Springer-Verlag, 1998.
- [10] G. Winskel. Category theory and models of parallel computation. In D. P. et al, editor, *Category Theory and Computer Programming*, volume 240 of LNCS, pages 266–281. Springer-Verlag, 1985.
- [11] W. Zielonka. Notes on finite asynchronous automata. *RAIRO, Informatique Théoretique et Applications*, 21:99–135, 1987.