Coordination and Composition in Multi-Agent Systems

Mehdi Dastani
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands
mehdi@cs.uu.nl

Farhad Arbab
CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands
farhad@cwi.nl

Frank de Boer
CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands
F.S.de.Boer@cwi.nl

ABSTRACT

In this paper we describe a channel-based exogenous coordination language, called Reo, and discuss its application to multi-agent systems. Reo supports a specific notion of compositionality for multi-agent systems that enables the composition and coordination of both individual agents as well as multi-agent systems. Accordingly, a multi-agent system consists of a set of individual and/or multi-agent systems whose collective behavior is coordinated by a Reo expression. This coordination language can be used to specify and implement the organization of multi-agent systems and their dynamic reconfiguration during system run.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Muliagent systems, Coherence and coordination

General Terms

Theory, Languages

Keywords

Coordination, Composition, Multi-Agent Systems

1. INTRODUCTION

Multi-agent systems represent an advance in abstraction which can be used by software developers to naturally model and construct systems for complex applications such as social simulations, manufacturing applications, electronic auctions, e-institutions, and business-to-business applications. These applications are understood and analyzed as multi-agent systems consisting of autonomous agents whose interactions are coordinated through an organizational structure [10].

The organizational structure of a multi-agent system specifies the coordinated behavior of individual agents and determines the overall behavior of the multi-agent system [5,

6, 8]. These organizational structures are usually described in terms of a variety of social and organizational concepts such as norm, trust, power, delegation of tasks, responsibilities, permissions, access to resources, and communication. The organization of multi-agent systems can optimize the activities of agents, manage and secure the information flow among agents, and guarantee certain outcomes as the result of agents' interactions.

Existing multi-agent approaches focus on the organization of individual agents [5, 6, 10]. On the other hand, applications like incident management, electronic markets, and e-governments involve the organization of different multi-agent systems. More specifically, incident management involves coordination of several multi-agent systems such as police, fire, and ambulance departments. Each of these multi-agent systems in turn consists of its own organizational structure that coordinates the behavior of its constituting individual agents or multi-agent systems. For instance, a police department consists of subdepartments such as the detectives department and the administration department. At the lowest level, the organization of multi-agent systems coordinate the behavior of individual agents.

This hierarchical perspective gives rise to a new model of the organizational structure of multi-agent systems that provides a separation of concerns between the autonomous behavior of multi-agent systems and the coordination of their mutual interactions. The main challenge addressed in this paper is to exploit this separation of concerns in the construction of multi-agent systems through composition of coordinated multi-agent systems. The most significant benefit of this approach is that multi-agent systems can be composed to form a more complex multi-agent system through composition of their coordination mechanisms.

We aim at the compositional construction of coordination models for multi-agent systems that reflect their organizational structures. Moreover, we investigate some properties of the composition operators which are in this case specific coordination mechanisms. In order to achieve this aim, we adopt an exogenous channel-based compositional coordination language, called Reo, to manage the interaction of individual agents and multi-agent systems [2]. This approach proposes a notion of compositionality for multi-agent systems that is essential for any computational methodology. Moreover, we give some examples to show that Reo coordination models can be interpreted in terms of social and organizational concepts.

In section 2, we discuss coordination and composition of multi-agent systems and define a specific notion of compo-

sitionality for multi-agent systems. In section 3, we briefly present the channel-based exogenous coordination language Reo, its syntax and semantics, and discuss its application to multi-agent systems. In section 4, we discuss the Reo-based composition of multi-agent systems and present some preliminary results. Finally, in section 5 we conclude the paper with final remarks and future work.

2. ORGANIZATION AND COORDINATION IN MULTI-AGENT SYSTEMS

In this paper, multi-agent systems are considered as consisting of individual agents/multi-agent systems and an organization structure that coordinates their behaviors. An organization structure determines the dependencies and interactions among constituting agents/multi-agent systems and is responsible for the global properties of the system. As argued in [5], the coordination of agents can be looked at from either the viewpoint of agents (subjective view) or the viewpoint of the external observer who is not involved in the multi-agent system (objective view). Although we focus on the objective view on coordination in this paper, we believe that optimal coordination in multi-agent systems can be achieved through a balanced combination of subjective and objective views [5].

Furthermore, we consider an organization structure as a coordination artifact. A motivation for the application of a coordination language to multi-agent systems is to achieve compositionality and to make the class of multi-agent systems closed under composition (i.e. coordination) operators. In [5] (page 277), it is stated:

> ... a purely subjective approach to coordination in the engineering of agent systems would endorse a mere reductionistic view, coming to say that agent systems are compositional, and their behaviour is nothing more than the sum of the individual's behaviour – an easily defeasible argument, indeed.

This may suggest that a non-purely subjective coordination approach means that multi-agent systems are not compositional. We disagree with this suggestion and reformulate the compositionality of multi-agent systems as follows: *the behavior of multi-agent systems is nothing more than the sum of the behaviors of its individual agents plus its coordination mechanism.*

In addition to the compositionality principle, other advantages of the objective approach to coordination are the encapsulation of system-level properties and the separation of concerns principle. These principles improve and simplify the development of multi-agent systems considerably [5]. The encapsulation of properties means that system-level properties can be specified and verified in terms of properties of the coordination artifacts (and not as properties of individual agents). For example, one can specify and verify that a coordination artifact does not allow the interception of specific information exchanged among some agents. The separation of concerns principle means that the organizational concepts should no longer be modelled (or implemented) indirectly in terms of individual agent concepts. For example, the requirement that agents in an E-market should register before making proposals are system-level requirements which can be modelled (or implemented) directly

in terms of social and organizational concepts (such as norm, permission, responsibility, and rights) rather than indirectly in terms of individual agent concepts (such as beliefs, goals, and plans).

In order to introduce an objective approach to coordination and composition of multi-agent systems, we assume a coordination language the expressions of which specify the coordination of individual agents or multi-agent systems. The expressions of this language are assumed to be compositional under coordination operators such that multi-agent systems can be composed to form more complex multi-agent systems through composition of their coordination mechanisms. This idea of composition of multi-agent systems is illustrated in Figure 1. In this figure, the sets $a_1, \ldots, a_n$ and $b_1, \ldots, b_m$ of individual agents are composed by the coordination models $C1$ and $C2$, respectively. These compositions result multi-agent systems $mas1$ and $mas2$, respectively. Moreover, multi-agent systems $mas1$ and $mas2$ are in turn composed by the coordination model $C3$ forming the third multi-agent system denoted by $mas3$. As suggested in this figure, the coordination of $mas1$ and $mas2$ is through the composition of their coordination mechanisms, i.e., $c3$ composes $C1$ and $C2$.
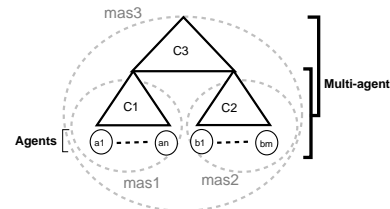

Figure 1: *The compositionality of multi-agent systems.*

In order to formally describe this idea of compositionality of multi-agent systems, we assume a coordination language C, which will be explained in details in the next section. The following simple inductive definition forms then the core of our notion of compositionality and provides the class of multi-agent systems closed under the coordination expressions.

DEFINITION 1 (MULTI-AGENT SYSTEMS). *Let Agents be a set of individual agents and C be a coordination language. The class of multi-agents systems MAS is defined as follows:*

- *If $A = \{a_1, \ldots, a_n\} \subseteq Agents$, $n \geq 1$ and $c^A \in C$ is a coordination expression that specifies the interactions of agents in A, then $c^A(a_1, \ldots, a_n) \in MAS$*

- *If $M = \{m_1, \ldots, m_k\} \subseteq MAS$, $k \geq 1$ and $c^M \in C$ is a coordination expression that specifies the interactions of multi-agent systems in M, then $c^M(m_1, \ldots, m_k) \in MAS$*

The expression $c^M(m_1, \ldots, m_k)$ should be read as "the multi-agent system consisting of multi-agent systems $m_1, \ldots, m_k$ coordinated by the coordination expression c^M". Note that according to this definition a multi-agent system may consist of one single agent and the coordination expression can be such that it does not constrain the behavior of the agent. It is important to observe that that this inductive definition requires some kind of compositionality of the underlying coordination language.

440

3. REO: A COORDINATION LANGUAGE

Reo is a channel-based exogenous coordination model that allows objective and dynamic coordination of agents and multi-agent systems [1, 2, 3]. In Reo, complex coordinators, called *connectors* or *networks*, are compositionally built out of simpler ones. The simplest connectors in Reo are a set of *channels* with well-defined behavior supplied by users [2]. The emphasis in Reo is on connectors, their behavior, and their composition, not on the agents that connect, communicate, and cooperate through them. The behavior of every connector in Reo imposes a specific coordination pattern on the agents that perform normal I/O operations through I/O interfaces of that connector, without the knowledge of those agents.
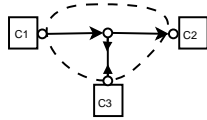

Figure 2: Connectors and component composition

Reo's notion of agents and connectors is depicted in Figure 2, where agents are represented as boxes, channels as arrows, and connectors are delineated by dashed lines. Each connector in Reo is, in turn, constructed compositionally out of simpler connectors, which are ultimately composed out of primitive channels.

For instance, the connector in Figure 2 may in fact be a flow-regulator which represents a system composed out of two *writer* agents (C1 and C3), plus a *reader* agent (C2). Every agent performs its I/O operations following its own timing and logic, independently of the others. None of these agents is aware of the existence of the others, the specific connector used to glue it with the rest, or even of its own role in the composite system. Nevertheless, the protocol imposed by our flow-regulator glue code (see [2] and [3]) ensures that a data item passes from C1 to C2 only whenever C3 writes a data item (whose actual value is ignored): the "tokens" written by C3, thus, serve as cues to regulate the flow of data items from C1 to C2. The behavior of the connector, in turn, is independent of the agents it connects: without their knowledge, it imposes a coordination pattern among C1, C2, and C3 that regulates the precise timing and/or the volume of the data items that pass from C1 to C2, according to the timing and/or the volume of tokens produced by C3.

Reo defines a number of operations for agents to (dynamically) compose, connect to, and perform I/O through connectors. Atomic connectors are *channels*. A channel is a primitive communication medium with exactly two ends, each with its own unique identity. There are two types of channel ends: *source* end through which data enters and *sink* end through which data leaves a channel. A channel must support a certain set of primitive operations, such as I/O, on its ends; beyond that, Reo places no restriction on the behavior of a channel. This allows an open-ended set of different channel types to be used simultaneously together in Reo, each with its own policy for synchronization, buffering, ordering, computation, data retention/loss, etc.

A connector is a set of channel ends organized in a graph of nodes and edges such that 1) zero or more channel ends coincide on every node, 2) every channel end coincides on exactly one node, 3) there is an edge between two (not nec-

essarily distinct) nodes iff there is a channel one end of which coincides on each of those nodes. A node is an important concept in Reo. Not to be confused with a location or an agent, a node is a logical construct representing the fundamental topological property of coincidence of a set of channel ends, which has specific implications on the flow of data among and through those channel ends.
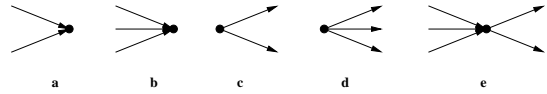

Figure 3: Sink, Source, and Mixed nodes

The set of channel ends coincident on a node A is disjointly partitioned into the sets $Src(A)$ and $Snk(A)$, denoting the sets of source and sink channel ends that coincide on A, respectively. A node A is called a *source node* if $Src(A) \neq \emptyset \wedge Snk(A) = \emptyset$. Analogously, A is called a *sink node* if $Src(A) = \emptyset \wedge Snk(A) \neq \emptyset$. A node A is called a mixed node if $Src(A) \neq \emptyset \wedge Snk(A) \neq \emptyset$. Figures 3.a and 3.b show sink nodes with, respectively, two and three coincident channel ends. Figures 3.c and 3.d show source nodes and Figure 3.e shows a mixed node where three sink and two source channel ends coincide. In the following, we use $\hat{x}$ to denote the unique node on which the channel end x coincides and use $[N]$ to denote the set of all channel ends coincident on node N (note that $x \in [\hat{x}]$).

Reo provides operations that enable agents to connect to and perform I/O on source and sink nodes only; agents cannot connect to, read from, or write to mixed nodes. This implies that the sink and source nodes define the I/O *interfaces* of Reo network. At most one agent can be connected to a (source or sink) node at a time. A component can write data items to a source node e that it is connected to through `write(t,e,v)` operation, where t is an optional time-out parameter, e is a channel end on which the agent performs the write operation, and v is the value written to every channel end $x \in [\hat{e}]$. The write operation succeeds only if all (source) channel ends coincident on the node accept the data item, in which case the data item is transparently written to every source end coincident on the node. A source node, thus, acts as a *replicator*.

An agent can obtain data items from a sink node that it is connected to through destructive take (`take(t,e,v,pat)`) and non-destructive read (`read(t,e,v,pat)`) operations. In these operations, t, e, and v are similar parameters as in the `write` operation except that v is also optional and *pat* is the optional parameter that indicates the pattern with which the data item to be assigned to v should match. A take operation succeeds only if at least one of the (sink) channel-ends coincident on the node offers a suitable data item; if more than one coincident channel end offers suitable data items, one is selected nondeterministically. A sink node, thus, acts as a nondeterministic *merger*.

A mixed node is a self-contained "pumping station" that combines the behavior of a sink node (merger) and a source node (replicator) in an atomic iteration of an endless loop: in every iteration a mixed node nondeterministically selects and takes a suitable data item offered by one of its coincident sink channel ends and replicates it into all of its coincident source channel ends. A data item is suitable for selection in an iteration, only if it can be accepted by all source channel

ends that coincide on the mixed node. A mixed node, thus, acts as a *merger* followed by a *replicator*.

It follows that every channel represents a (simple) connector with two nodes. More complex connectors are constructed in Reo out of simpler ones using its `join` operation. Joining two nodes destroys both nodes and produces a new node on which all of their coincident channel ends coincide. This single operation allows construction of arbitrarily complex connectors involving any combination of channels picked from an open-ended assortment of user-defined channel types. The semantics of a connector is defined as a composition of the semantics of its (1) constituent channels, and (2) nodes. The semantics of a channel is defined by the user who provides it. Reo defines the semantics of its three types of nodes, as mentioned above. In addition to the `join` operation, Reo proposes the `split(e,quoin)` operation which produces a new node N and splits the set of channel ends in $[\widehat{e}]$ between the old $\widehat{e}$ and N, according to the set of edges specified in *quoin*. Using this operation together with the `join` operation, the topology and structure of Reo networks can be changed dynamically. This is an essential characteristic of Reo which enables the dynamic reconfiguration of organization is multi-agent systems.

3.1   Coordination by Connectors

The simplest channels used in connectors are synchronous (`Sync`) channels, represented as simple solid arrows. A `Sync` channel has a source and a sink end, and no buffer. It accepts a data item through its source end iff it can simultaneously dispense it through its sink. A lossy synchronous (`LossySync`) channel is similar to a `Sync` channel, except that it always accepts all data items through its source end. If it is possible for it to simultaneously dispense the data item through its sink (e.g., there is a take operation pending on its sink) the channel transfers the data item; otherwise the data item is lost. `LossySync` channels are depicted as dashed arrows. An asynchronous channel can have a buffer with the bounded capacity x (`FIFO`$_x$). These channels are denoted by an arrow with the small box in the middle of the arrow representing their buffers. Such a channel accepts a data item through its source end and offers them through its sink end in the same order.

An example of the more exotic channels permitted in Reo is the synchronous drain channel (`SyncDrain`), whose visual symbol is like a `Sync` channel, but with two source ends. Because it has no sink end, no data value can ever be obtained from this channel. It accepts a data item through one of its ends iff a data item is also available for it to simultaneously accept through its other end as well. All data accepted by this channel are lost. A close kin of `SyncDrain` is the asynchronous drain (`AsyncDrain`) channel: it has two source ends through which it accepts and loses data items, but never simultaneously. `SyncSpout` and `AsyncSpout` are duals to the drain channel types as they have two sink ends [2].

The basic channels, also called simple Reo connectors, introduced above can be combined to make more complex Reo connectors. These Reo connectors can be used in many applications with organizational structures. In such applications, Reo connectors can be interpreted in terms of social and organizational concepts such as norms, permissions, delegation of tasks and information flow. For example, in auctions a permission mechanism is required that allows the administrator to control the bids of agents and permit only

their valid bids. In Figure 4, a REO connector is illustrated that models such a permission mechanism. According to this Reo connector, the administrator should give the permission to pass the valid bids from the buyers to the auctioneer. Note that this REO connector models only one simple aspect of auctions. A more comprehensive model of auction by Reo connectors is presented in [11].
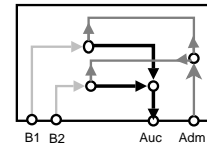
Figure 4: *A REO network for the bid permission part of the auction.*

3.2   Abstract Behavior Types (ABT)

An ABT defines an abstract behavior as a relation among the observable input/output that occur through a set of "contact points" without specifying any detail about: (1) the operations that may be used to implement such behavior; or (2) the data types those operations may manipulate for the realization of that behavior.

There are several different ways to formalize the concept of ABT. The formalization presented in [1] defines an ABT as a relation (on a set of timed-data-streams which will be defined below). This formalization emphasizes the relational aspect of the ABT model explicitly and abstracts away any hint of an underlying operational semantics of its implementation. This helps to focus on behavior specifications and their composition, rather than on operations that may be used to implement entities that exhibit such behavior and their interactions.

Timed-data-streams were introduced to define a coalgebraic semantics for Reo [3, 9]. A *stream* (over A) is an infinite sequence of elements of some set A. The set of all streams over A is denoted as A^ω. Streams in $DS = D^\omega$ over a set of (uninterpreted) data items D are called *data streams* and are typically denoted as α, β, γ, etc. Zero-based indices are used to denote the individual elements of a stream, e.g., $\alpha(0), \alpha(1), \alpha(2), \dots$ denote the first, second, third, etc., elements of the stream α. We use the infix "dot" as the stream constructor: $x.\alpha$ denotes a stream whose first element is x and whose successive elements are the elements of the stream α.

The well-known operations of head and tail on streams are called *initial value* and *derivative*: the initial value of a stream α (i.e., its head) is $\alpha(0)$, and its (first) derivative (i.e., its tail) is denoted as α'. The kth derivative of α is denoted as $\alpha^{(k)}$ and is the stream that results from taking the first derivative of α and repeating this operation on the resulting stream for a total of k times. Relational operators on streams apply pairwise to their respective elements, e.g., $\alpha \geq \beta$ means $\alpha(0) \geq \beta(0), \alpha(1) \geq \beta(1), \alpha(2) \geq \beta(2), \dots$.

Constrained streams in $TS = \mathbb{R}_+^\omega$ over positive real numbers representing moments in time are called *time streams* and are typically denoted as a, b, c, etc. To qualify as a time stream, a stream of real numbers a must be (1) strictly increasing, i.e., the constraint $a < a'$ must hold; and (2) progressive, i.e., for every $N \geq 0$ there must exist an index $n \geq 0$ such that $a(n) > N$.

A *Timed Data Stream* is a twin pair of streams $\langle \alpha, a \rangle$ in $TDS = DS \times TS$ consisting of a data stream $\alpha \in DS$ and a time stream $a \in TS$, with the interpretation that for all $i \geq 0$, the input/output of data item $\alpha(i)$ occurs at "time moment" $a(i)$. Two timed data streams $\langle \alpha, a \rangle$ and $\langle \beta, b \rangle$ are equal if their respective elements are equal, i.e. $\langle \alpha, a \rangle = \langle \beta, b \rangle \equiv \alpha = \beta \wedge a = b$.

Formalization of ABT in terms of timed data streams provides a simple yet powerful framework for the formal semantics of Reo. Timed data streams are used to model the flows of data through channel ends. A channel itself is just a (binary) relation between the two timed data streams associated with its two ends. A more complex connector is simply an n-ary relation among n timed data streams, each representing the flow of data through one of the n nodes of the connector.

The simplest channel, Sync, is defined as the relation:
$$\langle \alpha, a \rangle \text{ Sync } \langle \beta, b \rangle \equiv \alpha = \beta \wedge a = b$$
This equation states that every data item that goes into a Sync channel comes out in the exact same order. The equation states that the arrival and the departure times of each data item are the same: there is no buffer in the channel for a data item to linger on for any length of time.

A FIFO channel is defined as the relation:
$$\langle \alpha, a \rangle \text{ FIFO } \langle \beta, b \rangle \equiv \alpha = \beta \wedge a < b$$
As in a synchronous channel, every data item that goes in, comes out of a FIFO channel in exactly the same order ($\alpha = \beta$). However, the departure time of each data item is necessarily after its arrival time ($a < b$): every data item must necessarily spend some non-zero length of time in the buffer of a FIFO channel.

A FIFO_1 channel is very similar to a FIFO:
$$\langle \alpha, a \rangle \text{ FIFO}_1 \langle \beta, b \rangle \equiv \alpha = \beta \wedge a < b < a'$$
Not only the departure time of every data item is necessarily after its arrival time ($\alpha < \beta$), but since the channel can contain no more than 1 element, the arrival time of the next data item, , must be after the departure time of its preceding element ($a < b < a' \equiv a(i) < b(i) < a(i+1)$).

A $\text{FIFO}_1(D)$ represents an asynchronous channel with the bounded capacity of 1 filled to contain the data item D as its initial value. The behavior of a $\text{FIFO}_1(D)$ channel is very similar to that of a FIFO_1:
$$\langle \alpha, a \rangle \text{ FIFO}_1(D) \langle \beta, b \rangle \equiv \beta = D.\alpha \wedge b < a < b'$$
This channel produces an output data stream $\beta = D.\alpha$ consisting of the initial data item D followed by the input data stream α of the ABT, and (2) for $i \geq 0$ performs its ith input operation some time between its ith and $i+1^{st}$ output operations ($b < a < b'$).

A SyncDrain channel merely relates the timing of the operations on its two ends:
$$\langle \alpha, a \rangle \text{ SyncDrain } \langle \beta, b \rangle \equiv a = b$$
The replication that takes place at Reo nodes can be defined in terms of the ternary relation Rpl:
$$Rpl(\langle \alpha, a \rangle; \langle \beta, b \rangle, \langle \gamma, c \rangle) \equiv \beta = \alpha \wedge \gamma = \alpha \wedge b = a \wedge c = a$$
The semicolon delimiter separates "input" and "output" arguments of the relation. The relation Rpl represents the replication of the single "input" timed data stream $\langle \alpha, a \rangle$ into two "output" timed data streams $\langle \beta, b \rangle$ and $\langle \gamma, c \rangle$.

The nondeterministic merge that happens at Reo nodes is defined in terms of the ternary relation Mrg:

$$Mrg(\langle \alpha, a \rangle, \langle \beta, b \rangle; \langle \gamma, c \rangle) \equiv$$
$- \alpha(0) = \gamma(0) \wedge a(0) = c(0) \wedge Mrg(\langle \alpha', a' \rangle, \langle \beta, b \rangle; \langle \gamma', c' \rangle)$
   if $a(0) < b(0)$
$- \exists t : a(0) < t < min(a(1), b(1)) \wedge \exists r, s \in \{a(0), t\} \wedge r \neq s \wedge$
   $Mrg(\langle \alpha, r.a' \rangle, \langle \beta, s.b' \rangle; \langle \gamma, c \rangle)$ if $a(0) = b(0)$
$- \beta(0) = \gamma(0) \wedge b(0) = c(0) \wedge Mrg(\langle \alpha, a \rangle, \langle \beta', b' \rangle; \langle \gamma', c' \rangle)$
   if $a(0) > b(0)$

An ABT is thus a relation over timed data streams. Every timed data stream involved in an ABT is tagged either as its input or its output. The notation $R(I_1, \ldots, I_m; O_1, \ldots, O_n)$ is used to denote the ABT relation R defined on input timed data streams $I_1, \ldots, I_m$, also called input portals, and output timed data streams $O_1, \ldots, O_n$, also called output portals. Note that such an ABT is used to specify the semantics of a Reo connector with m source nodes corresponding to the input portals $I_1, \ldots, I_m$ and n sink nodes corresponding to the output portals $O_1, \ldots, O_n$.

Because an ABT is a relation, two ABTs can be composed to yield another ABT through a relational composition similar to the join operation in relational databases. This yields a simple, yet powerful formalism for specification of complex behavior as a composition of simpler ones. Composition of simple interaction primitives into non-trivial behavior, such as the Reo networks in the above examples, can be expressed as ABT composition [1]. Two ABTs A and B can be composed over a common timed data stream s if one is the producer of s (tag s as output) and the other is the consumer of s (tag s as input). This can be generalized to the composition of two ABTs over one or more timed data streams $s_1, \ldots, s_k$ if each ABT plays the role of the producer or the consumer of one of timed data streams s_i for $1 \leq i \leq k$. For example, consider the ABTs $A = \langle \langle \alpha, a \rangle, \langle \beta, b \rangle; \langle \gamma, c \rangle \rangle$ and $B = \langle \langle \delta, d \rangle; \langle \beta, b \rangle \rangle$. These two ABTs can be composed over the common timed data streams $\langle \beta, b \rangle$ resulting the composed ABT $C = \langle \langle \alpha, a \rangle, \langle \delta, d \rangle; \langle \gamma, c \rangle \rangle$. Note that $\langle \beta, b \rangle$ tagged as an input stream in the ABT A is connected to $\langle \beta, b \rangle$ tagged as an output stream in the ABT B.

4. REO-BASED COMPOSITION OF MULTI-AGENT SYSTEMS

In definition 1, the compositionality of multi-agent systems is defined in terms of a coordination language the expressions of which are considered as composition operators. In this section, we make this definition more concrete by taking Reo as the actual coordination language. Moreover, we assume that individual agents interact with a Reo network by performing Reo operations such as read, write, and take. Note that in the case of cognitive agents the values that are read/taken from or written to the Reo channels can be logical formulae including belief or goal formulae, and plan expressions. In this paper, we abstract from the exact nature of these values and focus on the coordination and composition of multi-agent systems.

A multi-agent system can thus consist of a Reo network that coordinates a non-empty set of individual agents or multi-agent systems. The basic case of multi-agent systems consists of a non-empty set of individual agents coordinated by a Reo network which can be as simple as one channel. However, the challenge is the composition of a set of multi-agent systems, which requires the composition of their constituting Reo networks in such a way that the composed Reo network coordinates the sets of agents that are involved in the constituent multi-agent systems. In principle, Reo net-

works can be composed through source, sink, and mixed nodes. The composition of Reo networks through mixed nodes is, however, semantically undefined. The reason is that the composition of ABT's is defined only in terms of input and output timed data streams which correspond to the source and sink nodes of Reo networks, respectively.

The composition of a set of multi-agent systems is thus established only through source and sink nodes (I/O interfaces) of their Reo networks. However, in closed multi-agent systems[1], where agents cannot be added to or removed from the system, the source and sink nodes of the corresponding Reo networks are supposed to be used by the existing agents to perform I/O operations. For closed multi-agent systems there is no sense in having a Reo network with source or sink nodes that are not used by agents. After all a Reo network is supposed to coordinate the behavior of agents through I/O interfaces used by the agents.

In order to compose two closed multi-agent systems, we propose a certain composition procedure which indicates that some agents need to be disconnected from the (source or sink) nodes of each corresponding Reo network in order to allow their composition through the source or sink nodes. The disconnected agents should then be re-connected to the composed Reo networks in a certain way as described in the rest of this section. Note that the disconnected agents cannot be re-connected to the nodes through which the composition is established since Reo does not allow to connect more than one agent to a (source or sink) node at a time. Moreover, agents cannot be connected to the nodes that are resulted by joining one source and one sink node since the resulting node is a mixed node to which no agent can be connected (see previous section).

The composition of two multi-agent systems can be established either directly by joining source or sink nodes of their Reo networks from which agents are disconnected, as illustrated in Figure 5-A, or indirectly through a Reo network with source and sink nodes that are joined with the sink and source nodes of the Reo networks of the multi-agent systems from which agents are disconnected, as illustrated in Figure 5-B. The reconnection of the disconnected agents to the composed Reo network (resulting from either a direct or an indirect join of the nodes) is accomplished by the following procedure which is illustrated in Figure 6.

Figure 5: *Direct and indirect composition through Reo connectors.*

[1]We do not study open multi-agent systems in this paper.

PROCEDURE 1. *Consider the nodes N_1 and N_2 from multi-agent systems mas_1 and mas_2 to which agents A_1 and A_2 are connected, respectively. If the composition requires direct join of the nodes N_1 and N_2 into node N (left column of Figure 6), then agents A_1 and A_2 are disconnected from N_1 and N_2 to allow their join according to Figure 5-A. Subsequently, agents A_1 and A_2 are connected to the node N of the composed Reo network as follows. If N_1 (or N_2) is a source node, then a synchronous channel and a source node*

N_1' (or N_2') are created such that the source end of the channel coincides on N_1' (or N_2') and its sink coincides on N. The agent A_1 (or A_2) is then connected to N_1' (or N_2'). If the node N_1 (or N_2) is a sink node, then a similar procedure is followed except that a sink node N_1' (or N_2') is created and the direction of the created channels is the other way around, i.e. it source is at N and its sink is at N_1' (or N_2'). If the composition requires an indirect join of the nodes through a Reo network (right column of Figure 6), then similar operations should take place twice.
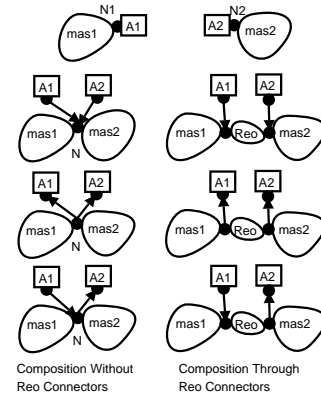

Figure 6: *Possible compositions of multi-agent systems.*

In the following, we show some properties of the multi-agent systems that are composed according to procedure 1. We first show that a Reo network R with a source (or sink) node N has the same behavior as another Reo network R' which is identical to R except that it has an additional source (or sink) node N' and a synchronous channel ch such that the sink (or source) end of ch coincides on N and its source (or sink) coincides on N'. In fact, in R' the (source or sink) node N' replaces the (source or sink) node N. The Reo networks R and R' are illustrated in Figure 7.

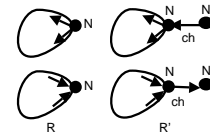**Figure 7:** *Equivalent multi-agent systems.*

THEOREM 1. *The Reo networks R and R', illustrated in Figure 7, have the same behavior, i.e. they have the same semantics.*

PROOF. Let node N from R be a source node and let the source of the channels $ch_1, \ldots, ch_n$ be connected to N. This implies that the behavior of N can be described as the following replicator: $Rpl(\langle \alpha, a \rangle; \langle \beta_1, b_1 \rangle, \ldots, \langle \beta_n, b_n \rangle) \equiv \alpha = \beta_i \wedge a = b_i \; \forall i \; 1 \leq i \leq n$ where $\langle \alpha, a \rangle$ is the input stream of node N and $\langle \beta_i, b_i \rangle$ are input stream of $ch_1, \ldots, ch_n$. Now suppose that $\langle \alpha, a \rangle$ is the input of the node N' in R'. Since this node is connected to N by a **Sync** channel, the input of the node N is determined by this

channel as follows: $\langle \alpha, a \rangle$ `Sync` $\langle \alpha', a' \rangle \equiv \alpha = \alpha' \wedge a = a'$. The behavior of N is determined by a replicator as follows: $Rpl(\langle \alpha', a' \rangle ; \langle \beta_1, b_1 \rangle, \ldots, \langle \beta_n, b_n \rangle) \equiv \alpha' = \beta_i \wedge a' = b_i \; \forall i \; 1 \leq i \leq n$. Since $\alpha = \alpha' \wedge a = a'$, the behavior of N in R' is defined as $\alpha = \beta_i \wedge a = b_i \; \forall i \; 1 \leq i \leq n$ which is the same as the behavior of N in R. Finally, since, R and R' are identical for all other nodes, we conclude that R and R' have the same behavior. We have assumed that N is a source node. If the node N is a sink node, the proof is similar except that the behavior of N is determined by a merger. $\square$

In order to study the suggested notion of compositionality for multi-agent systems, we may investigate the properties of the composition operators. According to definition 1, the coordination expressions (in this case Reo networks) are the composition operators such that the properties of the composition operators are the properties of the coordination expressions. Since the coordination expressions affect the interaction between agents or multi-agent systems, we may investigate the preservation of certain interactions between agents or multi-agent systems under specific compositions (or coordination mechanisms).
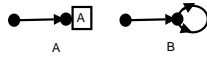

Figure 8: *The neutral multi-agent system.*

An interesting case to investigate is the preservation of interactions between agents or multi-agent systems when they are composed with a neutral multi-agent system mas_N according to the composition operation as described in procedure 1. In the sequel, we call this composition operation $proc_1$. A neutral multi-agent system is defined as an agent that is connected to the sink of one synchronous channel and performs continuously a take operation. This is illustrated in Figure 8-A (as illustrated in 8-B, the neutral multi-agent system can also be modelled by a Reo network without any agent). The idea is that the suggested composition of a multi-agent system mas with a neutral multi-agent system does not affect the interaction between agents from mas. In order to define this property in a formal way, we consider the interaction between agents in game theoretic sense in terms of game outcomes and game equilibria [7].

DEFINITION 2 (INTERACTION PRESERVING PROPERTY). *Let mas be a multi-agent system, mas_N be the neutral multi-agent system, and $proc_1$ be the composition operation as described in procedure 1. Let Agents be the set of agents, $A = \{a_1, \ldots, a_n\} \subseteq$ Agents be the subset of agents that are involved in multi-agent system mas, and Str_i be the set of strategies (actions) of agent a_i. If $(str_1^l, \ldots, str_n^k)$ is an outcome in mas, then it is also an outcome in $proc_1(mas, mas_N)$ and $proc_1(mas_N, mas)$, where $str_i^j \in Str_i$ for all agents a_i. Moreover, if $(str_1^l, \ldots, str_n^k)$ is an Nash equilibrium, then it remains a Nash equilibrium in $proc_1(mas, mas_N)$ and $proc_1(mas_N, mas)$.*

In the following, we assume that the preferences of agents on the outcomes of their interactions remains the same under the composition operations. Moreover, we assume that the action selection functions of agents (that determine possible interactions) are defined only in terms of agents' preferences and agents' information. Based on theorem 1, we show that the composition of multi-agent systems as described in procedure 1 satisfies the properties specified in definition 2.

THEOREM 2. *The $proc_1$ composition of multi-agent systems with a neutral multi-agent system and based on the direct join of source and sink nodes (as described in procedure 1 and illustrated by the left column of Figure 6) satisfies the property specified in definition 2.*

PROOF. Because of the space limitation, we present only a sketch of the proof. The left column of Figure 6 enumerates all possible compositions of two multi-agent systems. Suppose that the multi-agent system mas_2 is the neutral multi-agent system. Then, the behavior of the node N in the composed multi-agent system is determined only by the data streams from agents involved in the multi-agent system mas_1 which includes agent A_1. This implies that the interactions between the agents in the composed multi-agent system is determined only by the multi-agent system mas_1 in which agent A_1 is disconnected from N and connected to N' which in turn is connected to N by the synchronous channel. Theorem 1 states that this multi-agent system has the same behavior as mas_1 which means that the agent interactions in the composed multi-agent system is identical to the interactions in mas_1. $\square$

Note that there is no guarantee to preserve the interaction in a multi-agent system when the multi-agent system is composed with a non-neutral multi-agent system and through a *direct* join of the nodes as explained in procedure 1. This is due to the possibility of the uncontrolled flow of information from one multi-agent system to another. This may influence the behavior of the constituting agents and therefore affecting their interactions. The situation is different for the compositions of non-neutral multi-agent systems through indirect join of node and by means of specific Reo networks.

THEOREM 3. *Let mas_1 and mas_2 be two non-neutral multi-agent systems. The $proc_1$ composition of mas_1 and mas_2 by means of a `SyncDrain` channel (as described in procedure 1) satisfies the property specified in definition 2.*

PROOF. Following the $proc_1$ composition operation as described by procedure 1 and illustrated in the right column of Figure 6, the result consists of two modified multi-agent systems, which have the same behavior as mas_1 and mas_2 according to theorem 1. Moreover, the modified multi-agent systems are connected to each other at nodes N_1 and N_2 through a `SyncDrain` channel. According to the semantics of the channels, a `SyncDrain` relates only the timing of the operations on its two ends. Thus, no information flows between the modified multi-agent systems which implies that the interactions between agents in the modified multi-agent systems are preserved. $\square$

It should be noted that the composition and coordination of multi-agent systems is usually expected to change the behavior of the constituting multi-agent systems. For example, a coordination expression that composes multi-agent systems may be used to model a normative systems that will constrain the interaction of the constituting agents. Reo provides expressive and effective means to specify coordination mechanisms that model a variety of social and

organizational structures such as power structures, information flow, and delegation of goals or tasks. For example, the Reo network illustrated in Figure 4 models a permission structure, i.e. agent *admin* permits the communication between agents B_1 and B_2 on the one hand and agent *Auc* on the other hand.

5. CONCLUSION AND FUTURE WORK

We believe that multi-agent systems will be accepted as a computational model and software development methodology only if they can satisfy the principle of compositionality in an effective and satisfactory manner. A key problem related to the compositionality principle in multi-agent systems is the existence of two different computational entities. The first entity is an individual agent and the second is a multi-agent system. So, the question is how the notion of compositionality should be defined with respect to these two different computational entities. In other words, should the concepts of individual agents, multi-agent systems, or both be defined compositionally? The general agreement is that multi-agent systems are composed out of individual agents. However, this definition does not indicate whether individual agents and multi-agent systems are compositional entities. One of the existing approaches which has focused on the notion of compositionality in multi-agent systems is DESIRE [4]. The problem with this approach is that the computational models of individual agents and multi-agent systems are identical. This implies that DESIRE does not respect the principle of separation of concerns to distinguish individual agents from multi-agent systems. According to this view, a multi-agent system is modeled as an individual agent to which mental attitude and decision procedure can be assigned.

In this paper, we have provided a compositional definition of multi-agent system which is closed under coordination operations. In fact, multi-agent systems are defined in terms of individual agents and their organizational structure. Moreover, given a set of multi-agent systems, they can be composed to form a more complex multi-agent system by composing their organizational structures. In order to model the organizational structure of multi-agent systems, we have proposed to apply Reo. A significant advantage of Reo is the possibility of dynamic reconfiguration of organizational structures of multi-agent systems. Space limitation does not allow us to explore this aspect of Reo in this paper, which we leave for future research. We also have shown that the use of Reo and exogenous coordination is compatible with the *intentionality* that sets agents apart from normal processes, objects, or components.

It should be clear that Reo allows to specify interaction protocols directly. This is in contrast with other process algebras, like pi-calculus, that allows the specification of processes, not protocols. Reo allows the specification of protocols without specifying the processes or agents that participate in these protocols. This is the significance of exogenous coordination: a protocol can be imposed on a set of actors from outside the actors. Using Reo it is possible to formally specify, verify, and reason about the collaboration that happens in multi-agent systems since protocols are considered as Reo networks, independently of the agents. Finally, further research is needed to model specific social and organizational concepts as Reo networks and investigate their compositions to construct complex social organizations.

6. REFERENCES

[1] F. Arbab. Abstract behavior types: A foundation model for components and their composition. In F.S. de Boer, M.M. Bonsangue, S. Graf, and W.-P. de Roever, editors, *Formal Methods for Components and Objects*, volume LNCS 2852, Springer-Verlag, pages 33–70. 2003.

[2] F. Arbab. Reo: A channel-based coordination model for component composition. *Mathematical Structures in Computer Science*, 14(3):329–366, 2004.

[3] F. Arbab and J.J.M.M. Rutten. A coinductive calculus of component connectors. In D. Pattinson, M. Wirsing, and R. Hennicker, editors, *Recent Trends in Algebraic Development Techniques, Proceedings of 16th International Workshop on Algebraic Development Techniques (WADT 2002)*, pages 35–56. LNCS 2755, Springer-Verlag, 2003.

[4] F.M.T. Brazier, C.M. Jonker, and J. Treur. Compositional design and reuse of a generic agent model. *Applied Artificial Intelligence Journal*, 14:491–538, 2000.

[5] A. Omicini, S. Ossowski, and A. Ricci. Coordination infrastructures in the engineering of multiagent systems. In Federico Bergenti, Marie-Pierre Gleizes, and Franco Zambonelli, editors, *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*, chapter 14, pages 273–296. Kluwer Academic Publishers, 2004.

[6] A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *3rd international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, volume 1, pages 286–293, New York, USA, 2004. ACM.

[7] Martin J. Osborne and Ariel Rubenstein. *A Course in Game Theory*. The MIT Press, Cambridge, Massachusetts, 1994.

[8] A. Ricci, M. Viroli, and A. Omicini. Role-Based Access Control in MAS using Agent Coordination Contexts. In V. Dignum, D. Corkill, C. Jonker, and F. Dignum, editors, *1st International Workshop "Agent Organizations: Theory and Practice" (AOTP'04)*, pages 15–22. AAAI Press, 2004.

[9] J.J.M.M. Rutten. Elements of stream calculus (an extensive exercise in coinduction). In S. Brookes and M. Mislove, editors, *Proc. of 17th Conf. on Mathematical Foundations of Programming Semantics, Electronic Notes in Theoretical Computer Science*, volume 45, pages 23–26. 2001.

[10] F. Zambonelli, N. Jennings, and M. Wooldridge. Organizational abstractions in the analysis and design of multi-agent systems. In *First International Workshop on Agent-Oriented Software Engineering at ICSE*. 2000.

[11] Z. Zlatev, N. Diakov, and S. Pokraev. Construction of negotiation protocols for e-commerce applications. *ACM SIGecom Exchanges*, 5(2):12–22, 2004.