A language for Modular Information-passing Agents

Rogier M. van Eijk, Frank S. de Boer, Wiebe van der Hoek
and John-Jules Ch. Meyer

Universiteit Utrecht, Department of Computer Science
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
tel. +31–30–2531454
{rogier, frankb, wiebe, jj}@cs.ruu.nl

Abstract. For multi-agent systems, as for any complex system, a thorough theoretical foundation is indispensable. Hence, agent-oriented languages used for descriptions and implementations of multi-agent systems should be logically grounded and accompanied with a clear semantics. As a hopefully fruitful starting point towards such semantically well-founded languages, we propose a language of Modular Information-passing Agents. This language is designed for systems of agents inhabiting an environment on which they have a limited view or expertise, and hence in order to increase their knowledge, communicate on each other's expertises. We consider the syntax of the language and subsequently develop a structural operational semantics via a transition system.

1 Introduction

In this paper we attempt to bridge the gap between the extensive research on concurrent programming paradigms and the research on multi-agent systems. In our development of a system of Modular Information-passing Agents, being a stripped version of a multi-agent system, we try to incorporate as many useful concepts from existing concurrent programming languages like Concurrent Constraint Programming (CCP) [14], Communicating Sequential Processes (CSP) [9] and Algebra of Communicating Processes (ACP) [1] as possible. Whenever necessary we adapt them according to our purposes. We emphasize that our method contrasts with most of the current approaches, as we aim to develop a theoretically well-founded *algebraic* description of multi-agent systems. One of the advantages of such an approach is that it allows agent-oriented programs having a clear syntactical structure. Moreover, the meaning of a complex program can be understood by combining the meanings of its constituents. In this way a methodology for the top-down design of agent-oriented programs is obtained together with a mechanism for the specification and verification of these programs.

Multi-agent systems.

We view multi-agent systems [18] as systems composed of several interacting agents inhabiting an external environment. These agents are autonomous entities that are able to observe the world they inhabit and are capable of establishing changes in it. An additional interaction mechanism is provided by their ability to communicate with each other. The agents may be assigned mentalistic notions, such as knowledge, belief, desire and intention, which together with their reasoning processes direct their initiative-taking behaviour. In this paper however, we restrict ourselves to the mental attitude *knowledge on the external environment*, leaving the explication of other attitudes for future refinements of the framework.

Concurrent programming languages.

Many concurrent programming languages agree upon the incorporation of the general programming constructions of sequential composition, parallel composition, non-deterministic choice and recursion. Most of them also allow the introduction of local variables and mechanisms to rename variables.

We next discuss the distinguishing features of some concurrent programming languages we will base our framework on. First, CCP is regarded as a generalization of most of the concurrent logical languages. In this paradigm, several concurrently operating processes interact with each other via a shared store. This store is seen as a constraint on the range of values that variables can take, rather than as an explicit assignment of values to them. The processes build the store by supplying it with new constraints. These constraints may subsequently be inspected by the other processes, yielding a synchronization mechanism; the execution of a process checking for a constraint in the store is blocked until the constraint is eventually implied by the store. The paradigm assumes an underlying constraint system covering an entailment relation on constraints, an operation $\wedge$ to combine constraints and an operator $\exists$ to hide variables in constraints.

In contrast, CSP does not cover shared variables. Processes in this framework interact and synchronize by means of the transport of values along interconnecting channels. In (the synchronous version of) CSP, a sending process trying to emit a value, which is implied by its local state, and a receiving process trying to assign this value to one of its variables, have to agree upon the moment of communication. Until this moment their execution is blocked. The paradigm can be extended with a functionality from ACP allowing the introduction of encapsulated channels; channels that are local and can only be accessed by some processes. In this paper, we will attempt to integrate the concepts described above in a system of Modular Information-passing Agents.

Modular Information-passing Agents.

The framework of MI-Agents assumes a constraint system consisting of an entailment relation and a constraint language to express facts about the external world. These facts are gathered in a global store of constraints, called the *workspace*. We stress that this store is *the* representation of the external world; it is used for observations in the world as well as for establishments of new facts in it. We initially restrict ourselves to a workspace showing monotonic-increasing behaviour, thereby avoiding the problems of theory revision [5]. In future research, we will examine the implications of dropping this assumption.

The framework incorporates agent systems that maintain the workspace. These agent systems are compound; they are composed of smaller agent systems. The smallest systems are constituted by the basic agents, which are assigned a programming statement expressing their behaviour. Such a system may be viewed of as consisting of a team of experts that share a common view, or expertise. This view constitutes the interaction mechanism among the team members, as the members may inspect and extend it. In this paper, for the sake of clarity, we model views as windows of constraints on a collection of *accessible* variables. Thereby we abstract from more elaborate formalizations of views like signatures (languages). The interaction mechanism between teams is provided by the transport of information on expertises along interconnecting channels. The basic agents are assigned a communication base to store such information, which together with their own expertises constitute their knowledge bases.

The proposed framework incorporates many design features relating to the object-oriented paradigm. The agent systems are *hierarchical* as they are specified in terms of their components. The language allows the *encapsulation* of interconnecting channels as well as the encapsulation of data (local variables and expertises). Additionally, programming statements and communication stores assigned to agents are *private* to them. Finally, agents are designed in a *modular* fashion; a renaming paradigm can be used to define the interface with the workspace and the other agents.

The framework in perspective.

Before we examine the framework of MI-Agents in greater detail, we will try to situate its position. We imagine a wide spectrum of agent-oriented languages; at one end of the spectrum languages like AGENT-0 [15] reside, which are *implemented* programming languages that however suffer from the lack of both a firm logical foundation and a clear semantics. At the other end, *specification* languages for instance based on modal logics [16] or on the situation calculus [11] are located, which however cannot straightforwardly be implemented. As the development of agent-oriented languages that bridge the gap between these two extremes constitutes one of the current challenges, several alternative languages have been proposed. The development of the language CONGOLOG [11], which objective is the design of an *executable* version of an agent-oriented specification language, represents one way of decreasing the discrepancy. Alternative approaches to bridge the gap are those that start with a general, well-understood and implemented programming language and aim to accommodate it to suit descriptions of multi-agent systems. The language Concurrent METATEM [17], the language described in [2], as well as our language of Modular

Information-passing Agents serve as examples of the latter. The former two of these treatments are based on executable temporal logic and higher order logic, respectively, whereas our framework is underpinned by existing, well-understood concurrent programming languages. We thereby shift the stress on aspects of agent-oriented languages like concurrency, communication, synchronization and modularity. Moreover, in contrast to current more or less ad hoc approaches, we deal with these aspects in a theoretically well-founded algebraic manner.

The rest of the paper is organized as follows. In section 2 we embark on a description of the syntax of the language. The dynamics of the system is developed in section 3 which deals with transitions and semantics. We subsequently elaborate an example in section 4. In section 5 we identify several topics that need a closer examination and finish the treatment by discussing related research in section 6.

2 Syntax

In our framework, as in CCP, information systems are given by systems of constraints.

Definition 1. *(The constraint system)*
We assume a collection of constraints, which are represented in a first-order language $\mathcal{L}$ covering constructs built from variables typically denoted as x,y,z, functions typically written as f,g and predicates typically denoted as p. The constraint system comprises an information ordering $\vdash$, which is decided by a constraint solver. Additionally, to describe the semantics of the programming language, the system explicates an operation $\exists x$ to hide information and an operation $\wedge$ describing the accumulation of information (which mathematically corresponds to the least upper bound with respect to $\vdash$).

For instance, the first-order atoms *true*, $x = 7$ and $y = x$, the composition $x = 7 \wedge y = x$ as well as $\exists x(x = 7 \wedge y = x)$ are constraints. The latter can be considered equivalent to the constraint $y = 7$; that is, $\exists x(x = 7 \wedge y = x) \vdash y = 7$ and $y = 7 \vdash \exists x(x = 7 \wedge y = x)$. We assume the familiar notions of free and bound occurrences of variables in constraints. Vectors of variables are sequences typically given by $\mathbf{x}$ or when consisting of a single variable simply by x. If $\mathbf{x}$ denotes the vector $x_1,\ldots,x_n$ and $\mathbf{y}$ denotes $y_1,\ldots,y_n$ then the operation $\exists\mathbf{x}$ abbreviates $\exists x_1 \cdots \exists x_n$ and additionally, $\mathbf{x} = \mathbf{y}$ is a shorthand for $x_1 = y_1 \wedge \cdots \wedge x_n = y_n$. Finally, the set *Chan* is a set of channel names typically represented as c.

The formulation of the syntax of agent systems in our framework will be preceded by several additional definitions.

Definition 2. *(Workspaces, communication bases and windows)*
A *workspace* σ and a *communication base* B are simply constraints in $\mathcal{L}$. Let $Var(\sigma)$ be the vector of all variables occurring in σ. A *window* on a workspace σ with respect to the variables $\mathbf{x}$, is defined as

$$window(\sigma,\mathbf{x}) =_{def} \exists\mathbf{y}\sigma \qquad \text{where } \mathbf{y} \text{ equals } Var(\sigma)\setminus\mathbf{x}.$$

That is, a window on the workspace consists of the constraints on the variables $\mathbf{x}$; the constraints on all other variables are hidden.

Definition 3. *(Syntax of atomic actions a)*
Let $\varphi \in \mathcal{L}$ and $c \in Chan$. Atomic actions a are defined as

$$a ::= \mathsf{est}(\varphi) \mid \mathsf{verify}(\varphi) \mid \mathsf{send}(c,\varphi) \mid \mathsf{receive}(c,\varphi)$$

The language covers four atomic actions. The action $\mathsf{est}(\varphi)$ establishes the constraint φ in the workspace. The execution of $\mathsf{verify}(\varphi)$ succeeds if the constraint φ can be verified. Finally, the actions $\mathsf{send}(c,\varphi)$ and $\mathsf{receive}(c,\varphi)$ denote the emittance of the constraint φ along the channel c, and the reception of some constraint along the channel c yielding the constraint φ to be known, respectively.

Definition 4. *(Syntax of programming statements S)*
Let $x,y \in Var$ and let I be a finite, non-empty set of indices.

$$S ::= S \,\&\, S \mid \textstyle\sum_{i\in I} a_i.S_i \mid \mathsf{loc}_{\mathbf{x}}S \mid \mathsf{ren}_{\mathbf{yx}}S \mid P(x) \mid \mathsf{skip}$$

The statement S_1 & S_2 denotes the parallel composition of the statements S_1 and S_2. The statement $\sum_{i\in I} a_i.S_i$ stands for the non-deterministic choice between the statements S_i, which each are prefixed by an atomic action a_i. A non-deterministic choice with a singleton index set may be denoted by its single operand. The statement $\mathsf{loc}_{\mathbf{x}}S$ identifies the variables $\mathbf{x}$ to be local in S. The statement $\mathsf{ren}_{\mathbf{yx}}S$ expresses the variables $\mathbf{y}$ to be renamed to $\mathbf{x}$ in S. The statement $P(\mathbf{x})$ denotes a call to the procedure $P(\mathbf{y})$, where the vectors $\mathbf{x}$ and $\mathbf{y}$ stand for the actual and the formal parameters of the procedure, respectively (the framework incorporates a call-by-name parameter-passing mechanism). We assume a set W of (recursive) procedure declarations of the form $P(\mathbf{y}) :: S$ where S is a statement. Finally, the statement skip always succeeds and has no effects; the construct $\sum_{i\in I} a_i.\mathsf{skip}$ may be abbreviated to $\sum_{i\in I} a_i$.

Definition 5. *(Syntax of agent systems A)* Let $B \in \mathcal{L}$ and $c \in Chan$.

$$A ::= (S,B) \mid A + A \mid A \,;\, A \mid A \,\|\, A \mid \delta_c(A)$$

In our framework, basic agent systems, or teams, are given by their activities S, together with a collection B representing the information obtained from communication with other teams. We implicitly assume each basic agent is assigned an expertise, which is modeled as a vector of accessible global variables $\mathbf{x}$ constituting its window on the workspace. We require that whenever S contains the action $\mathsf{est}(\varphi)$, the free variables occurring in φ are either local variables, or accessible global variables $\mathbf{x}$, or are renamed versions of these local and global variables. This requirement corresponds to the idea that agents are not allowed to establish facts about variables other than their local variables and the global variables from their window. Complex agent systems are built from simpler ones by means of non-deterministic choice, denoted as +, sequential composition, represented as ; and parallel composition, which is denoted as $\|$. The execution of the agent system $A_1 + A_2$ consists of the execution of either A_1 or A_2. The execution of $A_1 \,;\, A_2$ is the execution of A_1 followed by that of A_2. The execution of $A_1 \,\|\, A_2$ is modeled as an interleaving of the executions of A_1 and A_2. The encapsulation operator δ_c when applied to the agent system A, defines the channel c to be local in A.

3 Transitions and operational semantics

Computation steps of agent systems are represented by transitions [13], which take systems from one configuration to subsequent ones. A configuration of an agent system A in a workspace σ is denoted as $\langle A,\sigma\rangle$. A transition is of the form

$$\langle A,\sigma\rangle \overset{\alpha}{\Longleftrightarrow} \langle A',\sigma'\rangle.$$

It indicates that the agent system A in a workspace σ performs a computation step resulting in an agent system A' (i.e. the part of A still to be executed) in a workspace σ'. The label α in the transition expresses whether the computation step involves some communication of information among the agents in the system, and if this is the case, additionally identifies the type of communication. Labels that denote such information-passing are of the form $c\,!\,\varphi$ or $c\,?\,\varphi$, where the symbols ! and ? stand for the emittance and the reception of information, respectively, φ denotes the constraint to be communicated and c is the channel it is to be transported along. The alternative label occurring in transitions is the label τ from CCS [12] representing internal, non-communicative computation steps. As agent systems are defined inductively, their transitions are defined in terms of the transitions of their components. For example, the transitions of the agent system $A_1 \,;\, A_2$ are defined in terms of those of the agent systems A_1 and A_2. The agent system performs the computation steps A_1 performs, and upon termination of A_1, the steps A_2 performs. In general, we describe the inference of transitions of agent systems by inference rules, which are of the form

$$\frac{\langle A_1,\sigma\rangle \overset{\alpha_1}{\Longleftrightarrow} \langle A_1',\sigma'\rangle \;\ldots\; \langle A_n,\sigma\rangle \overset{\alpha_n}{\Longleftrightarrow} \langle A_n',\sigma'\rangle}{\langle A,\sigma\rangle \overset{\alpha}{\Longleftrightarrow} \langle A',\sigma'\rangle}.$$

This rule states that the transition below the line can be concluded from the transitions above it. Rules with no premises are called axioms, written as $\langle A,\sigma\rangle \overset{\alpha}{\Longleftrightarrow} \langle A',\sigma'\rangle$. A collection of transition rules and axioms constitutes a *transition system*, which is a formal system for deriving transitions.

Definition 6. *(Transitions for atomic actions)*
As mentioned before, we assume each basic agent is assigned a vector $\mathbf{x}$ of accessible global variables. Let $K(\sigma,\mathbf{x},B)$ be the knowledge base defined by $\mathit{window}(\sigma,\mathbf{x}) \wedge B$ and let E be the standard symbol denoting successful termination. The transition system contains the following transitions concerning the atomic actions.

– The establishment of a constraint is defined to be its addition to the workspace.

$$\langle(\mathsf{est}(\varphi),B),\sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\!\rightarrow} \langle(E,B),\sigma\wedge\varphi\rangle$$

– The verification of a constraint succeeds if it is implied by the knowledge base.

$$\langle(\mathsf{verify}(\varphi),B),\sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\!\rightarrow} \langle(E,B),\sigma\rangle \qquad \text{if } K(\sigma,\mathbf{x},B)\vdash\varphi$$

– The emittance of a constraint along a channel is defined for those constraints that are implied by the knowledge base.

$$\langle(\mathsf{send}(c,\varphi),B),\sigma\rangle \overset{c\,!\,\varphi}{\Leftrightarrow\!\!\!\rightarrow} \langle(E,B),\sigma\rangle \qquad \text{if } K(\sigma,\mathbf{x},B)\vdash\varphi$$

– The reception of a constraint φ relative to the constraint ψ along a channel, succeeds for those constraints φ of which addition to the knowledge base yields a base from which ψ is derivable.

$$\langle(\mathsf{receive}(c,\psi),B),\sigma\rangle \overset{c\,?\,\varphi}{\Leftrightarrow\!\!\!\rightarrow} \langle(E,B\wedge\varphi),\sigma\rangle \qquad \text{if } (K(\sigma,\mathbf{x},B)\wedge\varphi)\vdash\psi$$

Informally, an establishment of a constraint corresponds to the idea that this constraint is *brought about*. The actions for emittance and reception denote intentions to communicate. They are not executed until in a parallel context, there is an agent with a matching intention. The statement $\mathsf{receive}(c,\psi)$ indicates that the agent is willing to accept any information along the communication channel c from which it is able to conclude ψ. We remark that this action integrates two alternative communication primitives. One is the uncontrolled reception of simply any constraint provided along the channel c, which can be mimicked by $\mathsf{receive}(c,\mathit{true})$. In the other, the receiver stores the conclusion ψ it wants to draw from the information provided along the channel, thereby ignoring any stronger information possibly included. The transition for the latter action is given by

$$\langle(\mathsf{receive\_only}(c,\psi),B),\sigma\rangle \overset{c\,?\,\varphi}{\Leftrightarrow\!\!\!\rightarrow} \langle(E,B\wedge\psi),\sigma\rangle \qquad \text{if } (K(\sigma,\mathbf{x},B)\wedge\varphi)\vdash\psi.$$

Hence, the action $\mathsf{receive}(c,\psi)$ represents the trade-off between the storage of just anything provided by the sender, and the storage of only a specific conclusion drawn from the information received; it accepts any information provided that the conclusion ψ can be drawn from it.

Example 7. (Information Retrieval)
Information Retrieval techniques [10] aim to support, in very large collections of data, the search for documents that satisfy some relevance criteria. One of the criteria is called *aboutness*, which is used to evaluate documents on their bearing on some particular piece of information. This example illustrates how the primitive actions from our framework can be used by agents assisting in an information retrieval process. We consider an information retrieval system containing a collection of documents distributed over several sites. Each site is assigned a group of agents, whose windows consist of information on the documents they can access, which are a sub-collection of all the documents located at the site. The agents use the primitives $\mathsf{est}(\varphi)$, $\mathsf{verify}(\varphi)$, $\mathsf{send}(c,\varphi)$ and $\mathsf{receive}(c,\varphi)$ in gathering information concerning the aboutness of documents. First of all, if at a particular site an agent has ascertained by some decision procedure that the document x is about p, where x is one of the agent's accessible documents, it performs $\mathsf{est}(\mathit{about}(x,p))$ to exhibit its establishment in the workspace. Subsequently, this constraint may be inspected by all agents operating at this site that have the document x in their view; an agent performs $\mathsf{verify}(\mathit{about}(x,p))$ to check whether the constraint $\mathit{about}(x,p)$ has already been established (by the agent itself or some other agent having access to x). This action is also used in case the document x is outside the agent's expertise, as the agent may already have gathered relevant information in its communication base. Alternatively, the agent may

choose to communicate with other agents. By performing $\mathsf{receive}(c, \mathit{about}(x,p))$ it indicates that it accepts any constraint along the channel c from which it is able to derive $\mathit{about}(x,p)$. Such agent may for instance communicate with another agent, possibly located at another site that performs $\mathsf{send}(c, \mathit{about}(x,p))$. Successful communication requires this sending agent to have $\mathit{about}(x,p)$ in its knowledge base.

Definition 8. *(Inference rule for non-deterministic choice between prefixed statements)*

$$\frac{\langle (a_j, B), \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\rightarrow} \langle (E, B'), \sigma'\rangle}{\langle ((\sum_{i\in I} a_i.S_i), B), \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\rightarrow} \langle (S_j, B'), \sigma'\rangle} \quad \text{where } j \in I$$

The transition of a non-deterministic choice between prefixed statements is given by a transition of one of these statements. The first computation step of a prefixed statement equals the transition of its prefix. Hence, from the transition of one of the prefixes a_j, which is labeled by α and which changes the communication store B to B' and the workspace σ to σ', we infer a transition of the non-deterministic choice $\sum_{i\in I} a_i.S_i$, which propagates the label α together with the communication store and workspace changes. It additionally identifies the statement S_j as the part that remains to be executed.

Definition 9. *(Inference rule for internal parallelism)*

$$\frac{\langle (S_1, B), \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\rightarrow} \langle (S_1', B'), \sigma'\rangle}{\begin{array}{c}\langle (S_1 \,\&\, S_2, B), \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\rightarrow} \langle (S_1' \,\&\, S_2, B'), \sigma'\rangle \\ \langle (S_2 \,\&\, S_1, B), \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\rightarrow} \langle (S_2 \,\&\, S_1', B'), \sigma'\rangle\end{array}}$$

An inference rule with several conclusions above each other, is used to abbreviate a collection of rules, each having one of them as its conclusion. The execution of a parallel statement $S_1 \,\&\, S_2$ is modeled as an interleaving of the computation steps of S_1 and S_2. The statement $S_1 \,\&\, S_2$ performs a computation step if one of the statements S_1 and S_2 executes one. Therefore, from a transition of S_1 we can infer a transition of $S_1 \,\&\, S_2$ in which S_1 acts as the left operand, or a transition of $S_2 \,\&\, S_1$ in which it is the right operand. We define $S \,\&\, E$ and $E \,\&\, S$ to be equal to S.

To constitute a computational model the framework allows the introduction of local variables. As in CCP, to describe its transition, we extend the syntax with a construct $\mathsf{loc}^{\rho}_{\mathbf{y}} S$ that expresses the variables $\mathbf{y}$ to be local in S and ρ the store of constraints on it. In this representation, the statement $\mathsf{loc}_{\mathbf{y}} S$ is defined as $\mathsf{loc}^{\mathit{true}}_{\mathbf{y}} S$. The corresponding transition rule is analogous to the one in CCP except that it also covers the communication of constraints along channels.

Definition 10. *(Inference rule for local variables)*

$$\frac{\langle (S, B_1), \rho_1\rangle \overset{\alpha}{\Leftrightarrow\!\!\rightarrow} \langle (S', B_2), \rho'\rangle}{\langle (\mathsf{loc}^{\rho}_{\mathbf{y}} S, B), \sigma\rangle \overset{\alpha'}{\Leftrightarrow\!\!\rightarrow} \langle (\mathsf{loc}^{\rho'}_{\mathbf{y}} S', B'), \sigma'\rangle}$$

where $B_1 = \exists \mathbf{y} B,\ \rho_1 = \rho \wedge \exists \mathbf{y}\sigma,\ B' = B \wedge \exists \mathbf{y} B_2,\ \sigma' = \sigma \wedge \exists \mathbf{y}\rho'$ and α' equals α except that in case of communication its constraint φ is replaced by $\exists \mathbf{y}\varphi$.

The transition of $\langle (\mathsf{loc}^{\rho}_{\mathbf{y}} S, B), \sigma\rangle$ is derived from a transition involving the statement S. As in $\mathsf{loc}^{\rho}_{\mathbf{y}} S$ the variables $\mathbf{y}$ are regarded local, and both the communication base B and the workspace σ contain constraints on the global variables $\mathbf{y}$, these global constraints should be overwritten by the local ones. This yields a workspace ρ_1. The communication base does not contain local constraints and is simply given by B_1. Additionally, it is implicitly assumed that the window is expanded with $\mathbf{y}$, as agents are allowed to verify and establish constraints on their local variables. After one computation step S' denotes the part of S that remains to be executed and B_2 and ρ' denote the new communication base and workspace, respectively. The workspace ρ' contains two types of constraints: constraints on the local variables $\mathbf{y}$ and constraints on all the other variables. The latter constraints are reflected in the workspace visible outside the scope of the statement, by the addition of $\exists \mathbf{y}\rho'$ to the information σ preserved under the computation step. The former ones are (although together with the other constraints) stored as ρ' in the construct $\mathsf{loc}^{\rho'}_{\mathbf{y}} S'$. The visible change of the communication base is reflected by the addition of $\exists \mathbf{y} B_2$ to the information B preserved under the computation. Additionally, the label α is adapted by the hiding of local constraints, yielding α'. We define $\mathsf{loc}^{\rho}_{\mathbf{y}} E$ to be equal to E. To illustrate the rule we consider an example.

Example 11. (Transition involving local variables)

$$\frac{\langle(\mathsf{send}(c, y=0), \mathit{true}), y = z \wedge z = 0\rangle \overset{c\,!\,y=0}{\Leftrightarrow\!\!\!\rightarrow} \langle(E, \mathit{true}), y = z \wedge z = 0\rangle}{\langle(\mathsf{loc}^{y=z}_{y}\mathsf{send}(c, y=0), \mathit{true}), z = 0\rangle \overset{c\,!\,\mathit{true}}{\Leftrightarrow\!\!\!\rightarrow} \langle(E, \mathit{true}), z = 0\rangle}$$

We remark that in this transition rule, whenever possible, we replaced constraints by simpler, logically equivalent ones. The example shows that the intention to send the local constraint $y = 0$ actually corresponds to the intention to send *true*. The example additionally indicates that in contrast to CCP, there is need for a special renaming paradigm. In CCP, the renaming of a variable z to y in S, can be simulated by the construction of $\mathsf{loc}^{y=z}_{y} S$. In our system however, such simulation does not take communication into account. The intention to send the local constraint $y = 0$ when y is a renamed version of z should actually correspond to the intention to send $z = 0$ and not like in the example, to the intention to send *true*. Hence, communication along channels in our framework gives rise, in addition to a construct for local variables, to the introduction of a separate renaming operator, as shown by example 11 and example 14 given below.

Renaming of variables allows the design of modular statements; in $\mathsf{ren}_{\mathbf{zy}} S$ the variables which are outside known as $\mathbf{z}$, are referred to as $\mathbf{y}$. Before we give the transition rule concerning renaming, we define what we mean by a simultaneous substitution.

Definition 12. We define the *simultaneous substitution* of $\mathbf{x}$ to $\mathbf{y}$ in φ by

$$\varphi[\mathbf{y}/\mathbf{x}] \; =_{def} \; \exists \mathbf{d}(\exists \mathbf{x}(\varphi \wedge \mathbf{x} = \mathbf{d}) \wedge \mathbf{d} = \mathbf{y})$$

where $\mathbf{d}$ are fresh variables not occurring in φ and distinct from $\mathbf{x}$ and $\mathbf{y}$.

The variables $\mathbf{d}$ are introduced in order to avoid problems related to name clashes between $\mathbf{x}$ and $\mathbf{y}$. Employing the definition above we for instance derive that $(x = 0)[x/x]$ equals $\exists d(\exists x(x = 0 \wedge x = d) \wedge d = x)$, which is logically equivalent to $\exists d(d = 0 \wedge d = x)$, and which subsequently equals $x = 0$.

Definition 13. *(Inference rule for the renaming of variables)*

$$\frac{\langle(S, B_1), \sigma_1\rangle \overset{\alpha}{\Leftrightarrow\!\!\!\rightarrow} \langle(S', B_2), \sigma_2\rangle}{\langle(\mathsf{ren}_{\mathbf{zy}} S, B), \sigma\rangle \overset{\alpha'}{\Leftrightarrow\!\!\!\rightarrow} \langle(\mathsf{ren}_{\mathbf{zy}} S', B'), \sigma'\rangle}$$

where $B_1 = (\exists \mathbf{y} B)[\mathbf{y}/\mathbf{z}]$, $\sigma_1 = (\exists \mathbf{y}\sigma)[\mathbf{y}/\mathbf{z}]$, $B' = B \wedge B_2[\mathbf{z}/\mathbf{y}]$, $\sigma' = \sigma \wedge \sigma_2[\mathbf{z}/\mathbf{y}]$ and α' equals α except that in case of communication its constraint φ is replaced by $\varphi[\mathbf{z}/\mathbf{y}]$.

Like for local variables, the transition of $\langle(\mathsf{ren}_{\mathbf{zy}} S, B), \sigma\rangle$ is derived from a transition involving S. In the statement S the variables $\mathbf{z}$ are known as $\mathbf{y}$, which implies that the global variables $\mathbf{y}$ are inaccessible. Hence, the constraints on the global variables $\mathbf{y}$ can be removed from the communication base and workspace. Additionally, the constraints on the variables $\mathbf{z}$ can be represented as constraints on $\mathbf{y}$, yielding B_1 and σ_1. After one computation step, S' represents the part of S that remains to be executed, B_2 and σ_2 denote the new communication base and workspace, respectively. The effects visible outside, are reflected by the addition of $B_2[\mathbf{z}/\mathbf{y}]$ and $\sigma_2[\mathbf{z}/\mathbf{y}]$, in which the constraints on $\mathbf{y}$ are represented as constraints on $\mathbf{z}$, to B and σ, respectively. The constraints in the label α are adapted accordingly, yielding α'. We put $\mathsf{ren}_{\mathbf{zy}} E$ to be equal to E.

Example 14. (Transition involving the renaming of variables)

$$\frac{\langle(\mathsf{send}(c, y=0), \mathit{true}), y = 0\rangle \overset{c\,!\,y=0}{\Leftrightarrow\!\!\!\rightarrow} \langle(E, \mathit{true}), y = 0\rangle}{\langle(\mathsf{ren}_{zy}\mathsf{send}(c, y=0), \mathit{true}), z = 0\rangle \overset{c\,!\,z=0}{\Leftrightarrow\!\!\!\rightarrow} \langle(E, \mathit{true}), z = 0\rangle}$$

The example shows that the intention to send $y = 0$, where y is a renamed version of z, actually corresponds to the intention to send $z = 0$.

Definition 15. *(Axiom for procedure calls)*

$$\langle (P(\mathbf{z}), B), \sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\to} \langle (\mathsf{ren}_{\mathbf{zy}} S, B), \sigma\rangle \qquad \text{where } P(\mathbf{y}) :: S \in W$$

The transition of a procedure call is an axiom of the transition system. The computation step of the call $P(\mathbf{z})$ is defined to be the replacement of the name of the procedure by its body S, in which the actual parameters are renamed to the formal ones. The workspace and the communication store are left intact.

Definition 16. *(Axiom for skip)*

$$\langle (\mathsf{skip}, B), \sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\to} \langle (E, B), \sigma\rangle$$

The execution of the statement skip always succeeds and leaves the communication base and workspace intact.

Next, we define transitions of compound agent systems.

Definition 17. *(Inference rules for the parallel, non-deterministic and sequential composition of agent systems)*

$$\frac{\langle A_1, \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\to} \langle A_1', \sigma'\rangle}{\begin{array}{c}\langle A_1 \parallel A_2, \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\to} \langle A_1' \parallel A_2, \sigma'\rangle \\ \langle A_2 \parallel A_1, \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\to} \langle A_2 \parallel A_1', \sigma'\rangle\end{array}} \qquad \frac{\langle A_1, \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\to} \langle A_1', \sigma'\rangle}{\begin{array}{c}\langle A_1 + A_2, \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\to} \langle A_1', \sigma'\rangle \\ \langle A_2 + A_1, \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\to} \langle A_1', \sigma'\rangle\end{array}} \qquad \frac{\langle A_1, \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\to} \langle A_1', \sigma'\rangle}{\langle A_1 \,;\, A_2, \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\to} \langle A_1' \,;\, A_2, \sigma'\rangle}$$

We additionally define successfully terminated basic agents (E, B) to be equal to E. The agent systems $E \,;\, A$, $A \parallel E$ and $E \parallel A$ are all defined to be equal to A. The execution of the parallel composition $A_1 \parallel A_2$ of two agent systems coincides with that of the parallel composition S_1 & S_2 of two statements; it is modeled as the interleaving of the execution steps of its components. The computation steps the non-deterministic composition of two agent systems takes, are exactly the computation steps of one of the agent systems involved. The transitions of the sequential composition of two agent systems A_1 and A_2 are the computation steps of A_1 followed by those of A_2.

Definition 18. *(Inference rule for communication)*

$$\frac{\langle A_1, \sigma\rangle \overset{c\,?\,\varphi}{\Leftrightarrow\!\!\to} \langle A_1', \sigma\rangle \quad \langle A_2, \sigma\rangle \overset{c\,!\,\varphi}{\Leftrightarrow\!\!\to} \langle A_2', \sigma\rangle}{\begin{array}{c}\langle A_1 \parallel A_2, \sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\to} \langle A_1' \parallel A_2', \sigma\rangle \\ \langle A_2 \parallel A_1, \sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\to} \langle A_2' \parallel A_1', \sigma\rangle\end{array}}$$

In case an agent in the agent system A_1 and an agent in the agent system A_2 want to communicate by means of the transport of the constraint φ along the channel c, the transition of $A_1 \parallel A_2$ can be inferred from both the transitions of A_1 and A_2. We note that the workspace thereby is left intact. Our choice in favour of *synchronous* communication is not essential. With respect to the asynchronous variant, it however frees us from the necessity of buffering emitted constraints.

Definition 19. *(Inference rule for encapsulation)*

$$\frac{\langle A, \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\to} \langle A', \sigma'\rangle}{\langle \delta_c(A), \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\to} \langle \delta_c(A'), \sigma'\rangle} \qquad \text{if } \alpha \text{ does not involve the channel } c$$

The encapsulation of channels restricts the propagation of communicative labels in inference rules. The operator δ_c when applied to the agent system A prohibits an agent residing inside A to communicate along the channel c with an agent that is located outside A. Hence, it defines c to be a local channel in the agent system A. We define $\delta_c(E)$ to be equal to E.

Example 20. (Local channels)
Consider the agent $A_1 = (\mathsf{send}(c,\varphi).S_1, \mathit{true})$ which intends to send information and the agents $A_2 = (\mathsf{receive}(c,\varphi).S_2, \mathit{true})$ and $A_3 = (\mathsf{receive}(c,\varphi).S_3, \mathit{true})$ which intend to receive information. In the agent system $\langle \delta_c(A_1 \parallel A_2) \parallel A_3, \sigma\rangle$, the agent A_1 can communicate with the agent A_2 along the channel c, but is however not able to use this channel to communicate with the agent A_3. In other words, the transition

$$\langle \delta_c(A_1 \parallel A_2) \parallel A_3, \sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\rightarrow} \langle \delta_c(A_1' \parallel A_2') \parallel A_3, \sigma\rangle$$

where $A_1' = (S_1, \mathit{true})$ and $A_2' = (S_2, \varphi)$ is derivable, whereas the transition

$$\langle \delta_c(A_1 \parallel A_2) \parallel A_3, \sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\rightarrow} \langle \delta_c(A_1' \parallel A_2) \parallel A_3', \sigma\rangle$$

with $A_1' = (S_1, \mathit{true})$ and $A_3' = (S_3, \varphi)$ is not derivable. To elucidate the latter we remark that one of the premises, viz.

$$\langle \delta_c(A_1 \parallel A_2), \sigma\rangle \overset{c\,!\,\varphi}{\Leftrightarrow\!\!\rightarrow} \langle \delta_c(A_1' \parallel A_2), \sigma\rangle$$

for application of the rule for communication is not derivable as its label $c\,!\,\varphi$ involves the channel c.

The transitions of agent systems give rise to computations.

Definition 21. A *computation* is a sequence

$$\langle A_0, \sigma_0\rangle \overset{\alpha_0}{\Leftrightarrow\!\!\rightarrow} \langle A_1, \sigma_1\rangle,\ \langle A_1, \sigma_1\rangle \overset{\alpha_1}{\Leftrightarrow\!\!\rightarrow} \langle A_2, \sigma_2\rangle,\ \cdots$$

of transitions between subsequent configurations of an agent system. Such a sequence is finite if it has a final configuration from which no transition is derivable. In case this configuration is of the form $\langle E, \sigma\rangle$ we identify the corresponding computation to have successfully *terminated*; in all other cases we say that the computation has *deadlocked*. A computation is *non-terminating* if the sequence of transitions is infinite.

Before we examine semantics, we state two key properties concerning workspaces and communication bases in computations. One is the property that in a computation the workspace increases in a monotonic fashion. That is, after each computation step the resulting configuration of the workspace implies the former configuration and hence, no information is ever removed from the workspace.

Theorem 22. *For every transition $\langle A, \sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\rightarrow} \langle A', \sigma'\rangle$ we have $\sigma' \vdash \sigma$.*

The proof of the theorem as the proof of the subsequent theorem is postponed to the appendix.

The second property amounts to the fact that in computations of the form

$$\langle A_0, \sigma_0\rangle \overset{\tau}{\Leftrightarrow\!\!\rightarrow} \langle A_1, \sigma_1\rangle,\ \langle A_1, \sigma_1\rangle \overset{\tau}{\Leftrightarrow\!\!\rightarrow} \langle A_2, \sigma_2\rangle,\ \cdots$$

all communication bases remain reflections of the workspace. That is, at each point in a computation all communication bases are implied by the workspace and hence, for instance, no communication base can become inconsistent with the workspace. The property is stated in theorem 23.

Theorem 23. *If all communication bases B of the basic agents (S,B) in the initial agent system A_0 are empty then it holds for each configuration $\langle A_i, \sigma_i\rangle$ $(i = 0,1,2,\ldots)$ in the computation*

$$\langle A_0, \sigma_0\rangle \overset{\tau}{\Leftrightarrow\!\!\rightarrow} \langle A_1, \sigma_1\rangle,\ \langle A_1, \sigma_1\rangle \overset{\tau}{\Leftrightarrow\!\!\rightarrow} \langle A_2, \sigma_2\rangle,\ \cdots$$

that for all (S,B) in A_i : $\sigma_i \vdash B$.

The operational semantics of a programming language is usually given by a notion of *observables*. These observables, which express what we want to observe of an agent system, are various: entire computations, subsequent workspace changes, the output workspaces, sequences of transition labels, and so on. In our system, we choose the observables to be resulting workspaces (and thereby abstract from mental attitudes).

Definition 24. *(Observables of successfully terminating agent systems)*

$$O(A)(\sigma) \;=\; \{\sigma_n \mid \langle A,\sigma\rangle \overset{\alpha_1}{\Leftrightarrow\!\!\!\rightarrow} \langle A_1,\sigma_1\rangle, \cdots, \langle A_{n-1},\sigma_{n-1}\rangle \overset{\alpha_n}{\Leftrightarrow\!\!\!\rightarrow} \langle E,\sigma_n\rangle\}$$

Because of the monotonic-increasing behaviour of workspaces, we are in the position to assign results to non-terminating computations. The observables of such computations are infinite conjunctions of subsequent workspaces.

Definition 25. *(Observables of agent systems)*

$$O^{\infty}(A)(\sigma) \;=\; O(A)(\sigma) \,\cup\, \{\sigma \wedge \sigma_1 \wedge \cdots \mid \langle A,\sigma\rangle \overset{\alpha_1}{\Leftrightarrow\!\!\!\rightarrow} \langle A_1,\sigma_1\rangle,\ \langle A_1,\sigma_1\rangle \overset{\alpha_2}{\Leftrightarrow\!\!\!\rightarrow}\ \cdots\ \}$$

where $\sigma \wedge \sigma_1 \wedge \cdots$ denotes the least upper bound with respect to the information order induced by $\vdash$.

A denotational semantics based on a notion of observables, constitutes a formal basis for the design and specification/verification of programs. In future research, we will hence aim to develop a compositional description constituting such a denotational model.

We end our discussion by giving an example, which is adapted from an example implemented in the DESIRE [4] framework.

4 Example of cooperating agents

Three basic agents A_1, A_2 and A_3 explore a 3-dimensional grid, which is constituted by the orthogonal base vectors **x**, **y** and **z**. The agent A_1 observes it through the 2-dimensional grid constituted by the vectors **x** and **z**, i.e. it inspects the *projection* on this grid. We model this by assigning the agent **x** and **z** as its accessible variables. For instance, the formula $ap((x,z),square)$ denotes that a square of unity length is centred at (x,z) in the projection, where x and z in **x** and **z**, respectively, and the predicate *ap* stands for 'appears'. In this example, the agent A_1 intends to provide along the channel c the centre of any object in its view:

$$(\textstyle\sum_{i,j,o}\mathsf{send}(c,ap((i,j),o)),true),$$

in which the index variables i,j,o range over the elements of **x**, **z** and the set of observable 2-dimensional objects, respectively.

In contrast, the agent A_2 senses the 3-dimensional grid through the grid constituted by the vectors **y** and **z**. Hence, it is assigned **y** and **z** as its accessible variables. In this case, A_2 intends to send along the channel d the centre of any object in its view:

$$(\textstyle\sum_{i,j,o}\mathsf{send}(d,ap((i,j),o)),true),$$

where the index variables i,j,o range over the elements of **y**, **z** and the set of observable 2-dimensional objects, respectively.

The agent A_3, which has no view on the grid, invokes the assistance of the agents A_1 and A_2 in establishing a picture of it. The agent considers the space as a collection **v** of positions. It for instance establishes $ap(v,o)$ whenever it has concluded that the 3-dimensional object o is centred at the position v. In this example, the agent A_3 asks the agent A_1 to examine the object centred at (x,z) in the $(\mathbf{x},\mathbf{z})$ grid. Subsequently, it requests the agent A_2 to take a look at the object centred at (y,z) in the $(\mathbf{y},\mathbf{z})$ grid. The reception of the constraints provided by A_1 and A_2 gives the agent A_3 sufficient information to conclude the 3-dimensional object that is centred at the position v, where v corresponds to the point (x,y,z) in the 3-dimensional grid. The agent is given by $(P(\mathbf{s}),true)$. Its statement consists of a call to the procedure $P(\mathbf{r}) :: S$, in which S, S' and S'' are the following abbreviations:

$$\begin{aligned}
&S = \textstyle\sum_{o_1}\mathsf{receive}(c,ap((r_1,r_3),o_1)).S'\\
&S' = \textstyle\sum_{o_2}\mathsf{receive}(d,ap((r_2,r_3),o_2)).S''\\
&S'' = \mathsf{est}(ap(r_4,f(o_1,o_2)))
\end{aligned}$$

The index variables o_1 and o_2 range over the set of observable 2-dimensional objects and the vectors **s** and **r** are abbreviations for (x,y,z,v) and (r_1,r_2,r_3,r_4), respectively. From this example we extract that agents have in addition to explicit knowledge collected in their knowledge base, implicit knowledge present in their statements. For instance, the call $P(\mathbf{s})$ implicitly assumes that the position v is associated with the point (x,y,z) in the 3-dimensional grid. By calling this procedure the agent A_3 somehow knows of this relation (this strongly relates to the traditional distinction between the notions *know what* and *know how*).

We now determine the operational semantics of the agent system constituted by the parallel composition of the basic agents A_1, A_2 and A_3 in the workspace $\sigma = (ap((x,z),square) \wedge ap((y,z),circle) \wedge f(square,circle) = cylinder)$. That is, we want to compute

$$O((A_1 \parallel A_2) \parallel A_3)(\sigma).$$

To do this, we first show the formal derivation of one of the successfully terminating computations, which will consist of the derivations of seven consecutive computation steps by means of the transition system.

Computation step 1.

The first computation step concerns the execution of the procedure call $P(\mathbf{s})$ by the agent A_3. It consists of the replacement of the call by the body of the procedure in which the actual parameters $\mathbf{s}$ are renamed to the formal parameters $\mathbf{r}$. Using the transition system, we derive from the instantiation

$$\langle (P(\mathbf{s}), true), \sigma \rangle \overset{\tau}{\Leftrightarrow\!\!\!\rightarrow} \langle (\mathsf{ren}_{\mathbf{sr}} S, true), \sigma \rangle$$

of the axiom for procedure calls, by means of the rule for the parallel composition of agent systems, the following transition, which constitutes the first computation step of our agent system:

$$\langle (A_1 \parallel A_2) \parallel A_3, \sigma \rangle \overset{\tau}{\Leftrightarrow\!\!\!\rightarrow} \langle (A_1 \parallel A_2) \parallel A_3', \sigma \rangle \tag{1}$$

where $A_3' = (\mathsf{ren}_{\mathbf{sr}} S, true)$.

Computation step 2.

The second computation step of the agent system consists of the communication of the fact $ap((x,z),square)$ along the communication channel c between the agent A_1 and the agent A_3. First, the following instantiation

$$\langle (\mathsf{send}(c, ap((x,z),square)), true), \sigma \rangle \overset{c\,!\,ap((x,z),square)}{\Leftrightarrow\!\!\!\rightarrow} \langle (E, true), \sigma \rangle$$

of the emittance axiom is valid, as the knowledge base $K(\sigma, (\mathbf{x}, \mathbf{z}), true)$ equals $\exists \mathbf{y} \sigma$, which implies the formula $ap((x,z),square)$. Applying the rule for non-deterministic choice, we obtain the transition (recall that the construct $\sum_{i \in I} a_i$ is an abbreviation for the statement $\sum_{i \in I} a_i.\mathsf{skip}$):

$$\langle (\textstyle\sum_{i,j,o} \mathsf{send}(c, ap((i,j),o)), true), \sigma \rangle \overset{c\,!\,ap((x,z),square)}{\Leftrightarrow\!\!\!\rightarrow} \langle (\mathsf{skip}, true), \sigma \rangle.$$

From this transition we derive by means of the rule for parallel composition, the computation step of the sub-agent system $A_1 \parallel A_2$, which is given by:

$$\langle A_1 \parallel A_2, \sigma \rangle \overset{c\,!\,ap((x,z),square)}{\Leftrightarrow\!\!\!\rightarrow} \langle A_1' \parallel A_2, \sigma \rangle \tag{2}$$

where $A_1' = (\mathsf{skip}, true)$. Secondly, from the instantiation

$$\langle (\mathsf{receive}(c, ap((r_1,r_3),square)), true), \sigma[\mathbf{r}/\mathbf{s}] \rangle \overset{c\,?\,ap((r_1,r_3),square)}{\Leftrightarrow\!\!\!\rightarrow} \langle (E, ap((r_1,r_3),square)), \sigma[\mathbf{r}/\mathbf{s}] \rangle$$

of the reception axiom, which is valid as its condition is trivially satisfied, we deduce employing the rule for non-deterministic choice, the transition:

$$\langle (\textstyle\sum_{o_1} \mathsf{receive}(c, ap((r_1,r_3),o_1)).S', true), \sigma[\mathbf{r}/\mathbf{s}] \rangle \overset{c\,?\,ap((r_1,r_3),square)}{\Leftrightarrow\!\!\!\rightarrow} \langle (S', ap((r_1,r_3),square)), \sigma[\mathbf{r}/\mathbf{s}] \rangle.$$

From this transition we subsequently derive by means of the inference rule for the renaming of variables, the transition:

$$\langle(\mathsf{ren}_{\mathbf{sr}}\Sigma_{o_1}\mathsf{receive}(c,ap((r_1,r_3),o_1)).S',true),\sigma\rangle \overset{c\,?\,ap((x,z),square)}{\Leftrightarrow\!\!\rightarrow} \langle(\mathsf{ren}_{\mathbf{sr}}S',ap((x,z),square)),\sigma\rangle. \qquad (3)$$

The rule for communication enables us to conclude from the transitions (2) and (3), the following, second computation step of the agent system:

$$\langle(A_1 \parallel A_2) \parallel A_3',\sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\rightarrow} \langle(A_1' \parallel A_2) \parallel A_3'',\sigma\rangle \qquad (4)$$

where $A_3'' = (\mathsf{ren}_{\mathbf{sr}}S',B)$ and $B = ap((x,z),square)$.

Computation step 3.

In order to deduce the next computation step of the agent system, which comprises of the communication of the fact $ap((y,z),circle)$ along the communication channel d between the agents A_2 and A_3, we ascertain that the instantiation

$$\langle(\mathsf{send}(d,ap((y,z),circle)),true),\sigma\rangle \overset{d\,!\,ap((y,z),circle)}{\Leftrightarrow\!\!\rightarrow} \langle(E,true),\sigma\rangle$$

of the emittance axiom is valid, as the knowledge base $K(\sigma,(\mathbf{y},\mathbf{z}),true)$, which is equal to $\exists\mathbf{x}\sigma$ implies $ap((y,z),circle)$. Using the inference rule for non-deterministic choice, we infer the following transition:

$$\langle(\Sigma_{i,j,o}\mathsf{send}(d,ap((i,j),o)),true),\sigma\rangle \overset{d\,!\,ap((y,z),circle)}{\Leftrightarrow\!\!\rightarrow} \langle(\mathsf{skip},true),\sigma\rangle.$$

From this transition we conclude using the rule for parallel composition, the computation step of the sub-agent system $A_1' \parallel A_2$, which looks like:

$$\langle A_1' \parallel A_2,\sigma\rangle \overset{d\,!\,ap((y,z),circle)}{\Leftrightarrow\!\!\rightarrow} \langle A_1' \parallel A_2',\sigma\rangle \qquad (5)$$

where $A_2' = (\mathsf{skip},true)$. Additionally, an application of the transition rule for non-deterministic choice to the instantiation

$$\langle(\mathsf{receive}(d,ap((r_2,r_3),circle)),B[\mathbf{r}/\mathbf{s}]),\sigma[\mathbf{r}/\mathbf{s}]\rangle \overset{d\,?\,ap((r_2,r_3),circle)}{\Leftrightarrow\!\!\rightarrow} \langle(E,B[\mathbf{r}/\mathbf{s}]\wedge ap((r_2,r_3),circle)),\sigma[\mathbf{r}/\mathbf{s}]\rangle$$

of the reception axiom (its condition is trivially satisfied), yields the transition

$$\langle(\Sigma_{o_2}\mathsf{receive}(d,ap((r_2,r_3),o_2)).S'',B[\mathbf{r}/\mathbf{s}]),\sigma[\mathbf{r}/\mathbf{s}]\rangle \overset{d\,?\,ap((r_2,r_3),circle)}{\Leftrightarrow\!\!\rightarrow} \langle(S'',B[\mathbf{r}/\mathbf{s}]\wedge ap((r_2,r_3),circle)),\sigma[\mathbf{r}/\mathbf{s}]\rangle.$$

The inference rule for the renaming of variables subsequently enables us to derive from this transition:

$$\langle(\mathsf{ren}_{\mathbf{sr}}\Sigma_{o_2}\mathsf{receive}(d,ap((r_2,r_3),o_2)).S'',B),\sigma\rangle \overset{d\,?\,ap((y,z),circle)}{\Leftrightarrow\!\!\rightarrow} \langle(\mathsf{ren}_{\mathbf{sr}}S'',B\wedge ap((y,z),circle)),\sigma\rangle. \qquad (6)$$

We finally conclude the third computation step of the system by applying the inference rule for communication to the transitions (5) and (6), giving:

$$\langle(A_1' \parallel A_2) \parallel A_3'',\sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\rightarrow} \langle(A_1' \parallel A_2') \parallel A_3''',\sigma\rangle \qquad (7)$$

where $A_3''' = (\mathsf{ren}_{\mathbf{sr}}S'',B')$ and $B' = B\wedge ap((y,z),circle)$.

Computation step 4.

Concerning the next computation step, which consists of the establishment of the fact $ap(v, f(square, circle))$ in the workspace σ by the agent A_3, we infer from the instantiation

$$\langle(\mathsf{est}(ap(r_4, f(square, circle))), B'[\mathbf{r}/\mathbf{s}]), \sigma[\mathbf{r}/\mathbf{s}]\rangle \overset{\tau}{\Leftrightarrow\!\!\!\rightarrow} \langle(E, B'[\mathbf{r}/\mathbf{s}]), \sigma[\mathbf{r}/\mathbf{s}] \wedge ap(r_4, f(square, circle))\rangle$$

of the establishment axiom, employing the transition rule for the renaming of variables, the transition :

$$\langle(\mathsf{ren}_{\mathbf{sr}}\mathsf{est}(ap(r_4, f(square, circle))), B'), \sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\!\rightarrow} \langle(\mathsf{ren}_{\mathbf{sr}}\mathsf{skip}, B'), \sigma \wedge ap(v, f(square, circle))\rangle.$$

If we use the rule for the parallel composition, we obtain from this transition the next computation step of the agent system:

$$\langle(A_1' \parallel A_2') \parallel A_3''', \sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\!\rightarrow} \langle(A_1' \parallel A_2') \parallel A_3'''', \sigma'\rangle \tag{8}$$

where $A_3'''' = (\mathsf{ren}_{\mathbf{sr}}\mathsf{skip}, B')$ and $\sigma' = \sigma \wedge ap(v, f(square, circle))$.

Computation steps 5, 6 and 7.

The next three computation steps are given by the executions of the skip statement by A_2, A_1 and A_3 successively. From the instantiation

$$\langle(\mathsf{skip}, true), \sigma'\rangle \overset{\tau}{\Leftrightarrow\!\!\!\rightarrow} \langle(E, true), \sigma'\rangle$$

of the axiom for skip we infer using the inference rule for parallel composition and the simplifications $(E, true) = E$ and $A_1' \parallel E = A_1'$, the transition:

$$\langle(A_1' \parallel A_2'), \sigma'\rangle \overset{\tau}{\Leftrightarrow\!\!\!\rightarrow} \langle A_1', \sigma'\rangle.$$

Another application of the rule for parallel composition enables us to derive from this transition, the computation step:

$$\langle(A_1' \parallel A_2') \parallel A_3'''', \sigma'\rangle \overset{\tau}{\Leftrightarrow\!\!\!\rightarrow} \langle A_1' \parallel A_3'''', \sigma'\rangle. \tag{9}$$

We are also able to infer from the above instantiation of the axiom for skip, using the rule for parallel composition and the simplifications $(E, true) = E$ and $E \parallel A_3' = A_3'$, the computation step:

$$\langle A_1' \parallel A_3'''', \sigma'\rangle \overset{\tau}{\Leftrightarrow\!\!\!\rightarrow} \langle A_3'''', \sigma'\rangle. \tag{10}$$

Finally, from the instantiation

$$\langle(\mathsf{skip}, B'), \sigma'\rangle \overset{\tau}{\Leftrightarrow\!\!\!\rightarrow} \langle(E, B'), \sigma'\rangle$$

of the axiom for skip, we conclude using the inference rule for renaming and the simplification $\mathsf{ren}_{\mathbf{sr}}E = E$, the following transition:

$$\langle(\mathsf{ren}_{\mathbf{sr}}\mathsf{skip}, B'), \sigma'\rangle \overset{\tau}{\Leftrightarrow\!\!\!\rightarrow} \langle(E, B'), \sigma'\rangle.$$

The inference rule for parallel composition enables us to derive from this transition, using the simplification $(E, B') = E$, the final computation step of the agent system:

$$\langle A_3'''', \sigma'\rangle \overset{\tau}{\Leftrightarrow\!\!\!\rightarrow} \langle E, \sigma'\rangle. \tag{11}$$

The transitions (1), (4), (7), (8), (9), (10) and (11) constitute a successfully terminating computation of the agent system. Its resulting workspace σ' implies $ap(v, cylinder)$. We state that all other successfully terminating computations also lead to this workspace σ'. Hence, the operational semantics of the agent system in the workspace σ is given by

$$\mathcal{O}((A_1 \parallel A_2) \parallel A_3)(\sigma) = \{\sigma'\}.$$

5 Extensions of the framework

We distinguish two principal directions for supplementary research. One comprises the deepening of the formalization of the current framework at hand. For instance, we aim, given the operational semantics, to develop an equivalent denotational semantics. As such a denotational model provides a basis for the specification and verification of programs, we might subsequently investigate the link with logical specification languages (for instance, the one described in [16]).

Alternatively, seen in the light of agent-oriented languages, the development of the framework of Modular Information-passing Agents is yet still in a preliminary phase. Many aspects of multi-agent systems need to be passed in review. One of the primary goals of future research is the incorporation of workspaces that are updated in a non-monotonic fashion, introducing the problems related to theory revision. For instance, if the external world is continuously subject to changes, information obtained from communication may be outdated. Secondly, in multi-agent systems there is need for elaborate communication primitives as requesting, informing, refusing, promising and so on. Such primitives at least imply the incorporation of agent-identity; the assignment of unique names to agents. Additionally, as statements govern the behaviours of agents, we should aim to get a hold on the implicit knowledge present in these statements. Related to this topic are the incorporation of goal-directed behaviour, as comprehensively examined in [8], and meta-knowledge, which constitute two essential characteristics of agent-oriented systems. Finally, an interesting aspect for future examination concerns the inheritance of knowledge from agents to complex agent systems. For instance, it is not immediately clear what kind of knowledge bases should be assigned to compound agent systems.

6 Related work

In addition to some connections alluded to, we will lightly touch upon the relation with some other affined approaches. First of all, Concurrent Transaction Logic [3] is a well-founded programming language designed for entities updating the state of a global store (i.e. a relational database, a knowledge base, a collection of communication buffers and so on). The language principally focuses on the interaction between the entities and the global database. In our framework, however, we additionally stress the knowledge each agent has about the store, necessitating communication of this knowledge among the agents. Secondly, whereas our approach concentrates on constraint languages, Transaction Logic leaves the underpinning language unspecified.

The modeling of communication among agents by means of interactions between actors is described in [6]. The approach incorporates the actor model, which is a framework that facilitates the expression of attitudes towards incoming messages. Moreover, in this model, the behaviour of entities is completely governed by the incoming messages (underlined by the fact that actors in order to change their attitude even send messages to *themselves*). The fact that the inhabited environment is left implicit additionally contrasts with our framework.

A *logical* treatment of modular agent systems; called a Logic of Contexts is described in [7]. Giunchiglia *et al.* have developed a formalism in which agent systems are hierarchies of logical theories, called contexts, connected by lifting and lowering bridge rules. A lifting rule ensures that if some specified formula φ holds in a sub-context, the associated formula φ' holds in the encompassing context. A lowering rule establishes the converse.

The DESIRE [4] framework used to design and specify interacting and reasoning components, also propagates modularity. It supports the modeling of modular components, which interact with each other via the transport of information along interconnecting links. The components may be built from smaller ones; the components that cannot be decomposed, are assigned a knowledge base expressing their reasoning capabilities and an information state representing their knowledge. In many applications, the external world is also modeled as a component. Each link in the framework transfers information between two components and as these components use distinct signatures (languages), the transfer necessitates the translation of information. The major distinctions with our framework are the use of signatures, the incorporation of meta-level reasoning, the modeling of the world as a component and the presence of mechanisms to update knowledge. However, as DESIRE currently possesses a semantics based on temporal logic, we believe that our framework may serve as an initial approach towards a structural operational semantics of the DESIRE methodology.

7 Appendix

In this appendix we show the proofs of theorem 22 and theorem 23.

Theorem 22. *For every transition* $\langle A,\sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\rightarrow} \langle A',\sigma'\rangle$ *we have* $\sigma' \vdash \sigma$.

Proof. The proof will proceed by induction on the length of derivation of the transition $\langle A,\sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\rightarrow} \langle A',\sigma'\rangle$. Concerning the induction basis we remark that for the only axiom $\langle(\mathsf{est}(\varphi),B),\sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\rightarrow} \langle(E,B),\sigma\wedge\varphi\rangle$ in which the workspace changes, the condition $\sigma\wedge\varphi\vdash\sigma$ certainly holds. The induction step in all cases immediately follows from an application of the induction hypothesis.

Before we commence the proof of theorem 23, which will consist of the proofs of three lemmas, let us first introduce a convenient abbreviation.

Definition 26. The property P for agent system configurations is defined as

$$P(\langle A,\sigma\rangle) \Leftrightarrow \text{ for all basic agents } (S,B) \text{ in } A \text{ it holds that } \sigma\vdash B.$$

The core of the proof of theorem 23 is constituted by the proof of lemma 29 which subsequently uses two results stated in lemma 27 and lemma 28.

Lemma 27. *If* $\langle A,\sigma\rangle \overset{c\,!\,\varphi}{\Leftrightarrow\!\!\rightarrow} \langle A',\sigma'\rangle$ *and* $P(\langle A,\sigma\rangle)$ *then* $P(\langle A',\sigma'\rangle)$ *and* $\sigma\vdash\varphi$.

Proof. The proof proceeds by induction on the length of derivation of the transition $\langle A,\sigma\rangle \overset{c\,!\,\varphi}{\Leftrightarrow\!\!\rightarrow} \langle A',\sigma'\rangle$. First, we consider the crucial case $A\equiv(\mathsf{send}(c,\varphi),B)$. Its transition is given by the axiom

$$\langle(\mathsf{send}(c,\varphi),B),\sigma\rangle \overset{c\,!\,\varphi}{\Leftrightarrow\!\!\rightarrow} \langle(E,B),\sigma\rangle \qquad \text{if } (\mathit{window}(\sigma,\mathbf{x})\wedge B)\vdash\varphi.$$

To show the claim we assume $(\mathit{window}(\sigma,\mathbf{x})\wedge B)\vdash\varphi$ and additionally $P(\langle(\mathsf{receive}(c,\psi),B),\sigma\rangle)$, that is, $\sigma\vdash B$. From these assumptions, $P(\langle(E,B),\sigma\rangle)$, i.e. $\sigma\vdash B$ immediately follows. We also conclude using the fact $\sigma\vdash \mathit{window}(\sigma,\mathbf{x})$ the other consequent: $\sigma\vdash\varphi$.

Additionally, we will work out the case $A\equiv(\mathsf{loc}^{\rho}_{\mathbf{y}}S,B)$ and omit all other cases as these require a similar, straightforward use of the induction hypothesis. The transition of this basic agent is given by the rule

$$\frac{\langle(S,B_1),\rho_1\rangle \overset{c\,!\,\varphi}{\Leftrightarrow\!\!\rightarrow} \langle(S',B_2),\rho'\rangle}{\langle(\mathsf{loc}^{\rho}_{\mathbf{y}}S,B),\sigma\rangle \overset{c\,!\,\exists\mathbf{y}\varphi}{\Leftrightarrow\!\!\rightarrow} \langle(\mathsf{loc}^{\rho'}_{\mathbf{y}}S',B'),\sigma'\rangle}$$

where $B_1=\exists\mathbf{y}B,\ \rho_1=\rho\wedge\exists\mathbf{y}\sigma,\ B'=B\wedge\exists\mathbf{y}B_2,\ \sigma'=\sigma\wedge\exists\mathbf{y}\rho'$.

In the following, we will frequently employ the fact that $\varphi\vdash\psi$ implies $\exists\mathbf{y}\varphi\vdash\exists\mathbf{y}\psi$. From the assumption $\sigma\vdash B$ and the above mentioned fact we derive that $P(\langle(S,B_1),\rho_1\rangle)$ holds. As the length of derivation of $\langle(S,B_1),\rho_1\rangle \overset{c\,!\,\varphi}{\Leftrightarrow\!\!\rightarrow} \langle(S',B_2),\rho'\rangle$ is shorter, we subsequently apply the induction hypothesis, yielding $\rho'\vdash B_2$ and $\rho_1\vdash\varphi$. From the former and $\sigma\vdash B$ we conclude $P(\langle(\mathsf{loc}^{\rho'}_{\mathbf{y}}S',B'),\sigma'\rangle)$. Secondly, from $\rho_1\vdash\varphi$ we conclude (as by theorem 22 we have $\rho'\models\rho_1$) that $\sigma'\vdash\exists\mathbf{y}\varphi$ holds, which completes the right-hand side of the claim.

Lemma 28. *If* $\langle A,\sigma\rangle \overset{c\,?\,\varphi}{\Leftrightarrow\!\!\rightarrow} \langle A',\sigma'\rangle$ *and* $P(\langle A,\sigma\rangle)$ *and* $\sigma\vdash\varphi$ *then* $P(\langle A',\sigma'\rangle)$.

Proof. We will prove this employing induction on the length of derivation of the transition $\langle A,\sigma\rangle \overset{c\,?\,\varphi}{\Leftrightarrow\!\!\rightarrow} \langle A',\sigma\rangle$. We consider the most relevant case $A \equiv (\mathsf{receive}(c,\psi),B)$. The applicable transition is given by

$$\langle(\mathsf{receive}(c,\psi),B),\sigma\rangle \overset{c\,?\,\varphi}{\Leftrightarrow\!\!\rightarrow} \langle(E,B\wedge\varphi),\sigma\rangle.$$

To show the claim we assume $\sigma \vdash B$ and $\sigma \vdash \varphi$. From these assumptions $P(\langle(E,B\wedge\varphi),\sigma\rangle)$, i.e. $\sigma \vdash B\wedge\varphi$ immediately follows.

In all other cases, the claim follows by a straightforward application of the induction hypothesis.

Lemma 29. *If $\langle A,\sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\rightarrow} \langle A',\sigma'\rangle$ and $P(\langle A,\sigma\rangle)$ then $P(\langle A',\sigma'\rangle)$.*

Proof. We prove this claim by induction on the length of derivation of the transition $\langle A,\sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\rightarrow} \langle A',\sigma\rangle$. We consider the most interesting case $A \equiv A_1 \parallel A_2$. One of its possible transitions is given by the inference rule

$$\frac{\langle A_1,\sigma\rangle \overset{c\,?\,\varphi}{\Leftrightarrow\!\!\rightarrow} \langle A_1',\sigma\rangle \quad \langle A_2,\sigma\rangle \overset{c\,!\,\varphi}{\Leftrightarrow\!\!\rightarrow} \langle A_2',\sigma\rangle}{\langle A_1 \parallel A_2,\sigma\rangle \overset{\tau}{\Leftrightarrow\!\!\rightarrow} \langle A_1' \parallel A_2',\sigma\rangle}$$

for communication. To show the lemma we assume $P(\langle A_1 \parallel A_2,\sigma\rangle)$, from which we derive $P(\langle A_1,\sigma\rangle)$ and $P(\langle A_2,\sigma\rangle)$. Using lemma 27 we conclude from the latter $P(\langle A_2',\sigma\rangle)$ and $\sigma \vdash \varphi$. Additionally, using lemma 28 we conclude from $P(\langle A_1,\sigma\rangle)$ and $\sigma \vdash \varphi$ that $P(\langle A_1',\sigma\rangle)$ holds. Combining both results we deduce $P(\langle A_1' \parallel A_2',\sigma\rangle)$.

The other possible transition is given by the inference rule

$$\frac{\langle A_1,\sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\rightarrow} \langle A_1',\sigma'\rangle}{\langle A_1 \parallel A_2,\sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\rightarrow} \langle A_1' \parallel A_2,\sigma'\rangle}$$

for parallel composition. From the assumption $P(\langle A_1 \parallel A_2,\sigma\rangle)$ we derive $P(\langle A_1,\sigma\rangle)$ and $P(\langle A_2,\sigma\rangle)$. As the length of derivation of $\langle A_1,\sigma\rangle \overset{\alpha}{\Leftrightarrow\!\!\rightarrow} \langle A_1',\sigma'\rangle$ is shorter, we subsequently apply the induction hypothesis and obtain $P(\langle A_1',\sigma'\rangle)$. Secondly, from theorem 22 we infer $\sigma' \vdash \sigma$, and hence we deduce from $P(\langle A_2,\sigma\rangle)$ that $P(\langle A_2,\sigma'\rangle)$ holds. If we combine both results we conclude $P(\langle A_1' \parallel A_2,\sigma'\rangle)$, which was to be shown.

We remark that in all other cases the claim follows from a simple application of the induction hypothesis and hence, these cases are omitted.

We are now in the position to prove theorem 23, which we will repeat below.

Theorem 23. *If all communication bases B of the basic agents (S,B) in the initial agent system A_0 are empty then it holds for each configuration $\langle A_i,\sigma_i\rangle$ $(i = 0,1,2,\ldots)$ in the computation*

$$\langle A_0,\sigma_0\rangle \overset{\tau}{\Leftrightarrow\!\!\rightarrow} \langle A_1,\sigma_1\rangle,\ \langle A_1,\sigma_1\rangle \overset{\tau}{\Leftrightarrow\!\!\rightarrow} \langle A_2,\sigma_2\rangle,\ \cdots$$

that for all (S,B) in A_i : $\sigma_i \vdash B$.

Proof. For each configuration the claim directly follows from i consecutive applications of lemma 29.

References

1. J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60:109–137, 1984.
2. C. Beyssade, P. Enjalbert, and C. Lefèvre. Cooperating logical agents. In *Proceedings of IJCAI'95 Workshop (ATAL)*, volume 1037 of *LNAI*, pages 299–314. Springer-Verlag, 1995.
3. A. Bonner and M. Kifer. Concurrency and communication in transaction logic. In *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 142–156, Bonn, 1996. MIT Press.
4. F. Brazier, B. Dunin-Keplicz, N. Jennings, and J. Treur. Formal specification of multi-agent systems: a real-world case. In *Proceedings of ICMAS-95*, pages 25–32. MIT Press, 1995.

5. P. Gärdenfors. *Knowledge in flux : Modelling the dynamics of epistemic states*. Bradford Books, MIT press, Cambridge, 1988.
6. M. Gaspari. Modelling interactions in agent system. In *Proceedings of the 4th Congres of the Italian Association for Artificial Intelligence*, pages 426–438. LNAI 992, 1995.
7. F. Giunchiglia, L. Serafini, E. Giunchiglia, and M. Frixione. Non-omniscient belief as context-based reasoning. In *IJCAI-93*, pages 548–554, 1993.
8. K.V. Hindriks, F.S. de Boer, W. van der Hoek, and J.-J.Ch. Meyer. Formal semantics for an abstract agent programming language. Technical report, Universiteit Utrecht, Department of Computer Science, 1997.
9. C.A.R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.
10. Th. Huibers. *An Axiomatic Theory for Information Retrieval*. PhD thesis, Universiteit Utrecht, 1996.
11. Y. Lespérance, H.J. Levesque, F. Lin, D. Marcu, R. Reiter, and R.B. Scherl. Foundations of a logical approach to agent programming. In *Proceedings of IJCAI'95 Workshop (ATAL)*, volume 1037 of *LNAI*, pages 331–346. Springer-Verlag, 1995.
12. R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.
13. G. Plotkin. A structured approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.
14. V.A. Saraswat and M. Rinard. Concurrent constraint programming. In *Proceedings of Seventeenth ACM Symposium on Principles of Programming Languages*, 1990.
15. Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.
16. B. van Linder, W. van der Hoek, and J.-J.Ch. Meyer. Communicating rational agents. In *KI-94: Advances in AI*, volume 861 of *LNCS*, pages 202–213. Springer-Verlag, 1994.
17. M. Wooldridge. A knowledge-theoretic semantics for concurrent METATEM. In *Proceedings of ECAI'96 Workshop (ATAL)*, volume 1193 of *LNAI*, pages 357–374. Springer-Verlag, 1996.
18. M. Wooldridge and N. Jennings. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.