A Minimal Calculus for Situated Multi-Agent Systems *

Chrysafis Hartonas
Technological Education Institute of Larissa, Greece
hartonas@teilar.gr

Abstract

We present a process-algebraic approach to situated multi-agent systems which incorporates syntax for agent systems as well as for the environments (workspaces) they live in. Agent states are characterized by (1) a belief state, (2) a goal (desire, need) state and (3) a capabilities state. Capabilities are determined by a subspace of the workspace the agent lives in, and they are represented in the agent state as a collection of types of objects (tools, in the agent's workspace) the agent has the expertise (or permission) to use. Action capabilities are then just permissible transformations of workspaces, restricted to the particular capabilities space of the agent. Workspaces (certain kinds of collections of objects, modelled here as simply typed records of labelled attributes) and workspace transformations (record update operations) are structured into a transition system. We use a simple model of objects as records and, therefore, of workspaces and thus we only deal in this report with deterministic environments. Transitions are determined by agent actions (modifications of properties of objects in their capabilities space, modelled as record update operations), thus allowing for a formal account of agent systems living in an ever changing environment. Beliefs and desires are formulae of a many-sorted, first-order multi-modal language of properties of workspaces, where sorts are object types and modalities are indexed by workspace transformations (record updates). The logical language is kept simple, allowing only for first-order beliefs (beliefs about properties of workspaces).

We present a language of situated, cooperative, self-interested agent systems, proposing a basic collection of agent behaviors (including ground observation actions, commitments, communication via assertions and requests, recursive behaviors, choice, concurrent behaviors etc), we provide an operational semantics for this language and discuss some examples of useful definable agent behaviors, such as perceiving (performing observations triggered by a statement). We then present and discuss notions of (behavioral) agent preorder and equivalence relations. Using the operational semantics and our notion of behavioral preorder and equivalence we propose an inequational theory for reasoning about agents. We conclude with presenting and discussing a case study for a simple agent system.

1 Introduction

Research on agent systems is vastly spread over the whole spectrum of AI and specifically agent-related issues, investigating decision making algorithms, learning algorithms,

*Research partially supported by the TEI of Larissa Research Committee

distributed planning and problem solving, coordination, cooperation and negotiation models, knowledge representation and communication language paradigms, diversifying classifications of various types of agent, organizational principles and architectures for agents, computational logics and reasoning models, implementation issues, often by extension of some object-oriented language, or by designing specifically agent-oriented languages (after Shoham's AGENT0 language) etc. Such interest in theoretical research on multi-agent systems, and in thereby empowered technologies, is practically very much justifiable, given the wide range of possible applications, from information management and electronic commerce, to air traffic control, telecommunications, manufacturing process control systems, etc.

Despite the extensive research on the subject, what seems to be largely missing in agent research is any systematic attempt to provide a clean semantic approach to agent-oriented computing, analogous to well established results for functional and concurrent programming and, more recently, distributed programming (π-calculus and related research), object oriented programming (ς-calculus and related research, or the actor model of computation, for concurrent objects with state).

There are very few exceptions of agent related research that can be thought of as heading in this direction ([4, 5, 6, 11]), despite the apparent benefits of such an approach, providing a formal forum for a high-level specification and verification of agent systems.

For example, in [4, 5, 6] the authors advocate a modular approach to a high-level specification of agent systems, focusing on communication as mere information passing and on belief revision, following observation actions that bind the agent's expertise variables to values they have in the environment. The only mental attitudes considered are beliefs and agents are viewed as triples $\langle \mathbf{x}, S, B \rangle$, where B is the belief base, $\mathbf{x}$ is a tuple of variables (the expertise variables of the agent, determining its perception window in the world) and S is a program generated by a standard syntax from ground behaviors (actions) of observing the values of the expertise variables, verifying or non-verifying a constraint φ (a formula of an unspecified constraint language), establishing a constraint φ on the variables $\mathbf{x}$ and sending or receiving assertions about the environment. The authors present a clean operational semantics for their language and prove, in [4], a result that belief revision, under the given operational semantics, maintains consistency of the belief base.

We think of our present work as falling within the same line of research. The main issue addressed in this report is the construction of a calculus (a process algebra) for multi-agent systems. This includes a language for the specification of such systems together with suitable notions of agent behavioral equivalence (that can be further exploited to investigate inequational theories for reasoning about agent systems).

A distinctive feature of our approach is with our understanding and treatment of agent-environment interaction. This includes both actions the agent can perform, as well as perception of the environment.

We view environments as certain collections of objects and regard agent actions strictly as modifications of properties (attributes) of objects (pushing an object to another location, changing an object's color, flipping a switch on or off, appending a string to a file object etc). Technically, we use a calculus of simply typed records to model objects, we define a notion of *workspace* (or environment, a collection of object terms closed under the subterm relation and under evaluation) and we demonstrate that workspaces and agent actions

(record update operations) form a deterministic transition system.

In the view we explore here, what concrete actions an agent can perform is determined by, first, the environment it finds itself in and, second, its action *capabilities*. Capabilities are determined by a subspace of the workspace the agent inhabits and they are represented in the agent state as a collection of types of objects (tools, in the agent's workspace), intuitively the types of objects that the agent has the requisite expertise (or permission) to use in provoking modifications of the shared workspace. Action capabilities are then just permissible transformations of workspaces, restricted to the particular capabilities space of the agent. We view agent systems as concurrently executing agents, sharing a common workspace, accessible to observation by all, though each one of them has its own capabilities subspace, which determines the roles the agent can assume in an agent society. Though the current state of the common workspace is visible to all agents, individual capabilities limit what "thought experiments" each agent can perform, in the sense that they limit each agent's ability to see how the environment can be changed, since the actions (workspace transformations) each agent can perform are limited by its own individual capabilities.

Workspaces are finitely generated model structures, supporting or failing to support the agent's beliefs, desires or commitments and intentions. In attempting to satisfy a desire or commitment, the agent sets out to modify its workspace so as to construct a model of the desire or commitment. We put forth a language of properties of workspaces, a many-sorted, multi-modal, first-order language, where sorts are object types and modalities are indexed by agent actions (record update operations). This language can be used to designate agent beliefs, desires etc, as well as persistent (integrity) constraints on the workspace. The semantics of the logical language is given with respect to the transition system of workspaces.

Beliefs an agent has are a partial representation of its workspace. In the view we follow here, the observational portion of an agent's belief base consists of ground sentences about the values of attributes of objects, such as `MyPoint.xpos`=3, or `MyFile.name`="filename". Further beliefs, such as `MyPoint.xpos`<5, or $(\exists x)$(`MyPoint.xpos` $+ x \geq 12$) are derivable from the basic facts and the agent's reasoning theory Θ. Observational beliefs are formed by perception acts (observations). Ground observations are direct observations of the values of attributes of objects, via a basic behavior $\texttt{obs}(M_i : \tau_i)_{i \in n}$. More complex observational behaviors can be then defined and we introduce, by definition, a behavior $\texttt{obs}(\varphi)$ for performing observations triggered by a statement φ.

Apart from workspace transforming actions and observation, agent behaviors include behavior choice, two types of conditional, concurrent and sequenced behaviors as well as recursively defined behaviors. They also include a belief revision and a goal-updating behavior, as well as communication behaviors. For simplicity, we restrict communication to just two types of speech act, namely assertions and requests.

Intentions, in the framework we explore here, are not part of the agent state. Intentions can be variably thought of as either a subset of desires (the ones the agent is currently committed to), leading to a planning behavior, or as a set of concurrently or sequentially executing plans, aiming at the satisfaction of some of the desires (intentions in the first sense). Intentions in the latter sense are just *behaviors*, while we view intentions in the first sense as *commitment behaviors* (committing to ψ is the behavior discharged by calling the planner to supply a plan for bringing it about that ψ). Viewing intentions as commitment behaviors, we place them in the action queue of the agent, rather than in its state. Intentions

appear then organized, sequentially (prioritized), or concurrently. In broad terms, however, the agent architecture underlying our proposed agent calculus can be thought of as a variant of the belief-desire-intention architecture.

In the rest of this report we define workspaces, introduce our language of properties of workspaces and our language of agent systems, we provide operational semantics for the language of agents, and explore some definable behaviors. Finally, having presented our language of agent systems together with its operational semantics, we introduce notions of behavioral preorder and equivalence of agents that can be further exploited to establish an inequational theory for reasoning about agents.

2 Agent Workspaces and their Properties

We develop here a simple model for a workspace transition system, where workspaces are certain collections of object terms and the language of objects is that of a simply typed record calculus. Objects as records may be modified by modifying their properties (updating their attributes) and these modifications may propagate in the entire workspace. In this Section, we work out some technical details involved in showing that workspaces form a deterministic transition system.

2.1 Introducing Workspaces

Agents are situated in some environment or other. Thus, any abstract model of agents will invariably consider (1) an *environment* E, viewed as a set of environment states $E = \{U_1, \ldots, U_n, \ldots\}$ and (2) a set of *actions*, or *environment transformers* $Act = \{\alpha_1, \ldots, \alpha_n, \ldots\}$, where a transition $U \stackrel{\alpha}{\longrightarrow} V$ indicates a state transformation, due to α being performed. An environment state is what we call here a *workspace* the agent lives in. Actions α are not unconditionally applicable to just any environment state, or workspace (you can't paint any objects blue if no brush objects are available in your workspace), but we may assume a (monotone) function $act : E \longrightarrow \mathcal{P}(Act)$, determining just what actions can be applied in what particular workspaces. Consistent with the common-sense view that, by and large, we perceive our macro-environment by reifying it (cutting it up into small discrete chunks, the objects), we think of environment states (workspaces) as certain kinds of sets of objects (thus admitting a partial order structure on the set of environment states) and view the latter as collections of attributes. This is quite meaningful from a computational perspective, too, as objects (viewed as neatly packed sets of attributes) can be encoded in a variety of programming languages as structures (Prolog or C structures, for example), or association lists (as in Scheme and LISP), or records (explicitly so in languages like ML), or as class-instances, i.e. objects in the object-oriented programming sense (C++, Java etc).

Real-world environments can be vastly complicated, better modelled as continuous rather than discrete, with fuzzy rather than clear cut objects, being only partially accessible to perception, governed by nondeterminism in their transformations, changing at high speed rather than stationary or slow evolving, with discrete actions concurrently executed composing (fusing) to produce results that differ from those produced by each action, separately executed etc. Thus designing an agent system depends on having an appropriate model for the type of environment the agents will inhabit.

We restrict here to *discrete, fully accessible, deterministic* environments, with *no action fusion* (no compositional interaction between discrete workspace transforming actions). Environments (workspaces) of this kind can be adequately modelled as certain collections of (simply typed) records (with records modelling objects). In this Section we provide formal definitions and we work out the technical details needed to establish that workspaces, as defined, form a deterministic transition system, where transition actions are record updates (modifications of properties of objects in the workspace).

The framework we propose can be roughly described as follows:

1. We have a typed language (with typed variables $x : \tau$, for each type τ) of expressions denoting objects. These include ground (or "abstract") objects such as numbers, strings etc, as well as composite objects, regarded as clusters of attributes. In our concrete model of this Section, composite objects are precisely proper record expressions. We write $\trianglerighteq M : \tau$ to indicate that M can be shown to be of type τ and use ρ as a type metavariable to stand for types other than those of the ground objects, such as numbers, strings etc (in our concrete workspace model those are proper record types (row types) $\rho = [\ell_1 : \tau_1, \ldots, \ell_n : \tau_n]$)

2. There is a partial evaluation function on object expressions, denoted by $eval(M) = N$ or $M \Downarrow N$ (N is the value of M, such as, for example, the color of a colored object, or its position etc), which is precisely record expression evaluation

3. Workspaces form a partial ordering, under a suitable notion of subspace, indicated by $U \preceq V$, and there is a suitable notion of workspaces being generated by a set of objects (see Definition 2.6, for our concrete workspace model), indicated by $U = \langle\, \{M_i : \rho_i \mid i \in I\} \,\rangle$. By a finitely generated workspace we mean a workspace $U = \langle M_1 : \rho_1, \ldots, M_n : \rho_n \rangle$ where each M_i is provably of some row type ρ_i.

4. For each workspace U, there is a well-defined set of actions $act(U)$ that can be performed on U. We write $U \xrightarrow{\alpha} V$ to indicate that V results by performing α on U. We show that workspace transitions are deterministic, so that actions are really partial functions defined on the set of workspaces. For this reason, we also write $U\{\alpha\}$ for the workspace V such that $U \xrightarrow{\alpha} V$.

2.2 Objects as Records

Record terms M are defined by the syntax

$$M := x\ (x \in X) \mid c\ (c \in C) \mid [\ell_1 = M_1, \ldots, \ell_n = M_n]_{n \geq 0} \mid M.\ell \mid M[\ell \Leftarrow M']$$

where X is a countable set of record variables, C is a set of constants (numbers, strings, booleans etc) and the labels ℓ_i in $[\ell_1 = M_1, \ldots, \ell_n = M_n]$ are pairwise distinct. $M.\ell$ is field selection (extraction of the value of the attribute labelled by ℓ) and $M[\ell \Leftarrow M']$ is record update, redefining the field ℓ to label M'. The syntax may be extended to include operations *op* on ground terms (numbers, strings etc), and we will use such operators in examples, though we do not explicitly include them in the syntax.

A *normal form*, or *value*, is a record term N specified by the syntax

$$N := x\ (x \in X) \mid c\ (c \in C) \mid [\ell_1 = M_1, \ldots, \ell_n = M_n]_{n \geq 0}$$

Adopting a product notation for repeated field selections, $M.\ell_1 \cdots .\ell_n = M \prod_{n=1,\ldots,n} \ell_i$, or updates, $M[\ell_1 \Leftarrow M_1] \cdots [\ell_m \Leftarrow M_m] = M \prod_{i=1,\ldots,m} [\ell_i \Leftarrow M_i]$, it is clear from the syntax that any record term has the general form

$$M = K \prod_{i \in r} (\prod_{j \in s_i} \ell_{ij})(\prod_{k \in t_i} [\ell_{ik} \Leftarrow M_{ik}])$$

where $r, s_i, t_i \in \omega$ and K is either a variable, or a constant, in which cases $r = 0$, or a term of the form $[\ldots, \ell = M, \ldots]$. The proof of this claim is immediate, by structural induction on M.

In the sequel we work with typed records, so that type annotations become part of the syntax of record terms. Thus the language of typed records includes typed variables $x : \tau$, for each type τ, typed constants $c : \kappa_c$, and typed terms

$$[\ell_1 = M_1 : \tau_1, \ldots, \ell_n = M_n : \tau_n] : [k_1 : \tau'_1, \ldots, k_m : \tau'_m]$$
$$(M : \tau).\ell : \tau_1$$
$$(M : \tau)[\ell \Leftarrow L : \tau'] : \tau_1$$

2.3 Types, Subtyping and Operational Semantics for Records

Assuming some ground types κ (for integers, booleans, perhaps strings etc) the types τ for the record language we have presented are simple

$$\begin{array}{lcl} \kappa & := & \texttt{int} \mid \texttt{bool} \mid \cdots \\ \tau & := & \kappa \mid \rho \\ \rho & := & [\ell_1 : \tau_1, \ldots, \ell_n : \tau_n]_{n \geq 0} \end{array}$$

where the labels ℓ_i are pairwise distinct.

We adopt one subtyping axiom

$$[\ell_1 : \tau_1, \ldots, \ell_n : \tau_n, \ldots, \ell_{n+m} : \tau_{n+m}] \sqsubseteq [\ell_1 : \tau_1, \ldots, \ell_n : \tau_n]$$

and the usual reflexivity and transitivity rules

$$\tau \sqsubseteq \tau \qquad\qquad \frac{\tau_1 \sqsubseteq \tau_2 \qquad \tau_2 \sqsubseteq \tau_3}{\tau_1 \sqsubseteq \tau_3}$$

Terms are proven well-typed by the typing rules listed in Table 1. If M is provably of type ρ, then we say it is of a *row type* (proper record type).

Mentioning types in contexts where the type annotations are not of significance is rather awkward and we will omit them when no harm is done.

A big-step operational semantics for the evaluation of well-typed record terms is given in Table 2. We use the notational convention $N[\![\ell = A]\!]$ to indicate that the normal form N is a record with an ℓ label naming A.

In the update axiom our notation is intended to mean that only the field labelled by ℓ gets modified.

Proposition 2.1 (Determinacy) *If $M \Downarrow N$ and $M \Downarrow K$, then $N = K$.*

Table 1: Typing Rules

(K) $\quad \trianglerighteq c : \kappa_c$

(R) $\quad \dfrac{\trianglerighteq M_i : \tau_i \qquad i = 1, \ldots, n \ (n \geq 0)}{\trianglerighteq [\ell_1 = M_1 : \tau_1, \ldots, \ell_n = M_n : \tau_n] : [\ell_1 : \tau_1, \ldots, \ell_n : \tau_n]}$

(Upd) $\quad \dfrac{\trianglerighteq M : \tau \qquad \trianglerighteq L : \tau_i}{\trianglerighteq (M : \tau)[\ell_i \Leftarrow L_i : \tau_i] : \tau} \ (\tau = [\ell_1 : \tau_1, \ldots, \ell_n : \tau_n])$

(Sel) $\quad \dfrac{\trianglerighteq M : \tau}{\trianglerighteq (M : \tau).\ell_i : \tau_i} \ (\tau = [\ell_1 : \tau_1, \ldots, \ell_n : \tau_n])$ $\quad$ (Sub) $\quad \dfrac{\trianglerighteq M : \tau \qquad \tau \sqsubseteq \tau'}{\trianglerighteq M : \tau'}$

Table 2: Big-Step Operational Semantics for Records

(Normal Form) $\quad N \Downarrow N$

(Update-1) $\quad [\ldots, \ell = A, \ldots][\ell \Leftarrow B] \Downarrow [\ldots, \ell = B, \ldots]$

(Update-2) $\quad \dfrac{M_1 \Downarrow N_1[\![\ell = A]\!] \qquad N_1[\ell \Leftarrow B] \Downarrow N}{M_1[\ell \Leftarrow B] \Downarrow N}$

(Selection) $\quad \dfrac{M_1 \Downarrow N_1[\![\ell = A]\!] \qquad A \Downarrow N}{M_1.\ell \Downarrow N}$

PROOF: Immediate, by transition induction. ∎

Subject Reduction and Minimal Types theorems for this type system are well-established in the literature on record calculi.

Theorem 2.2 (Subject Reduction) *If $\trianglerighteq M : \tau$ and $M \Downarrow N$, then $\trianglerighteq N : \tau$.* ∎

Theorem 2.3 (Minimal Types) *If $\trianglerighteq M : \tau$, then there is a minimal type σ, in the $\sqsubseteq$-order, such that $\trianglerighteq M : \sigma$.* ∎

2.4 Workspaces

Roughly put, a workspace is essentially a subset of the full language of objects that is transitively closed under the subterm relation and under evaluation. The precise definition follows.

Definition 2.4 (Workspaces) *A family W of object terms is a* workspace *provided that (1) W includes all the variables and ground object terms (constants and terms of the form $op(\bar{M})$, if included in the syntax), and (2) for any $M \in W$*

- *If $M = [f_1 = K_1, \ldots, f_m = K_m]$ then for each $1 \leq s \leq m$, $M.f_s, K_s \in W$*
- *If $M = B.f$, then $B \in W$ and*
- *If $M = B[f \Leftarrow L]$ then $B, L \in W$.*

- *If $M \Downarrow N$, then $N \in W$.* ∎

Lemma 2.5 *The family of workspaces over the full language of objects is closed under arbitrary unions and intersections (it is a complete lattice under set-theoretic unions and intersections).* ∎

Given the complete lattice structure of the family of workspaces, we can define workspaces generated by a fixed set of objects.

Definition 2.6 ((Finitely) Generated Workspaces) *Let $(M_i)_{i\in I}$ be an arbitrary family of objects. Then the workspace $\langle M_i\rangle_{i\in I}$* generated by the *$M_i$ is the smallest workspace containing all the M_i, in other words*

$$\langle M_i\rangle_{i\in I} = \bigcap\{W |\ W \ \textit{ is a workspace and } \ \forall i \in I\ M_i \in W\}$$

We say that a workspace W is finitely generated *(has a finite base) if there exist $A_i \in W$, where $i \in n$, for some natural number n, such that $W = \langle A_i\rangle_{i\in n}$. We may sometimes abuse terminology by saying that W is a* finite *workspace, meaning that it is finitely generated.*

The "empty" workspace $\langle\ \rangle$ is the space consisting only of abstract entities (variables, numbers etc). The full language $\mathcal{L}$ of objects (modelled here as records) is also a workspace and it constitutes a bounding universe of action and discourse. ∎

In working with collections of objects (the workspaces), rather than with single objects, we need to provide an account of changes brought about to the workspace, due to operations performed on its objects. For a simple example, the workspace may simply be generated by a rectangle object $R = [\texttt{topLeft} = P, \texttt{bottomRight} = Q]$, where P and Q are point objects (modelled as records of coordinates), say $P = [\texttt{xpos} = 1, \texttt{ypos} = 11]$ and $Q = [\texttt{xpos} = 17, \texttt{ypos} = 0]$. The operation $P[\texttt{xpos} \Leftarrow P.\texttt{xpos} + 1]$ moves the top-left corner P of the rectangle. Viewing this as an operation on the workspace we need to take into account the fact that not only the point P, but also the rectangle R, hence the entire workspace, has been modified. The result of an operation performed on a workspace is not always as transparent as our simple example indicates. We clarify this issue, from a technical point of view, in the sequel.

Definition 2.7 *For each converging computation $M \Downarrow N[\![\ell = L]\!]$, we define a sequence of normal forms $\ell_{M,N}$, by transition induction, as follows:*

Case 1: *$M = N$ is a normal form and the reduction is $N \Downarrow N$. We set $\ell_{M,N} =< N >$.*

Case 2: *$M = K[\ell \Leftarrow L] \Downarrow N$, with K a normal form. We set $\ell_{M,N} =< N, K >$.*

Case 3: *$M = M_1[\ell \Leftarrow A] \Downarrow N$, where M_1 is not a normal form. Then the reduction is by the rule*

$$\frac{M_1 \Downarrow N_1[\![\ell = A]\!] \qquad N_1[\ell \Leftarrow L] \Downarrow N}{M = M_1[\ell \Leftarrow L] \Downarrow N}$$

By induction, ℓ_{M_1,N_1} is defined and we set $\ell_{M,N} =< N >^\frown \ell_{M_1,N_1}$.

Case 4: $M = M_1.\ell \Downarrow N$ *and the reduction is by the selection rule*

$$\frac{M_1 \Downarrow N_1[\![\ell = L]\!] \qquad L \Downarrow N}{M = M_1.\ell \Downarrow N}$$

By induction the sequence $\ell_{L,N}$ *is defined and we set* $\ell_{M,N} = \ell_{L,N}$. ∎

The following theorems will be used in determining workspace transitions.

Theorem 2.8 *Let* $M \Downarrow N[\![\ell = L]\!]$ *and* $\ell_{M,N} =< N = N_1, N_2, \ldots, N_n = K >$ *be the sequence of Definition 2.7. Then there exists a unique term* $K_1 = K \prod_{i \in r}[\ell_i \Leftarrow A_i]$ *such that* $K_1 \Downarrow N$ *and* $\ell_{M,N} = \ell_{K_1,N}$.

PROOF: The proof is by transition induction.

Case 1: $M = N \Downarrow N$. Then $\ell_{M,N} =< N >$ and the normal form K of the theorem is just $K = N$. Taking $K_1 = K = N$ and $r = 0$ the claim becomes trivial.

Case 2: $M = K[\ell \Leftarrow L] \Downarrow N$ where K is a normal form. Then $\ell_{M,N} =< N, K >$ and we may take $K_1 = M = K[\ell \Leftarrow L]$, so that the claim is again trivial.

Case 3: $M = M_1[\ell \Leftarrow L] \Downarrow N$, where M_1 is not a normal form, $M_1 \Downarrow N_1[\![\ell = A]\!]$ and $\ell_{M,N} =< N >^\frown \ell_{M_1,N_1} =< N, N_1, \ldots, K >$.

By induction let $K_0 = K \prod_{i \in r}[\ell_i \Leftarrow A_i] \Downarrow N_1$, with $\ell_{M_1,N_1} = \ell_{K_0,N_1}$. Then set $K_1 = K_0[\ell \Leftarrow L]$. Clearly $K_1 \Downarrow N$, by an application of the update rule

$$\frac{K_0 \Downarrow N_1[\![\ell = A]\!] \qquad N_1[\ell \Leftarrow L] \Downarrow N}{K_0[\ell \Leftarrow L] \Downarrow N}$$

Using the definition of the ℓ-sequences and the induction hypothesis it follows that

$$\ell_{M,N} =< N >^\frown \ell_{M_1,N_1} =< N >^\frown \ell_{K_0,N_1} = \ell_{K_1,N}$$

Case 4: $M = M_1.\ell \Downarrow N$ by the selection rule

$$\frac{M_1 \Downarrow N_1[\![\ell = L]\!] \qquad L \Downarrow N}{M = M_1.\ell \Downarrow N}$$

and $\ell_{M,N} = \ell_{L,N} =< N, \ldots, K >$. By induction, let $K_1 = K \prod_{i \in r}[\ell_i \Leftarrow A_i] \Downarrow N$, such that $\ell_{K_1,N} = \ell_{L,N}$. Then by definition of $\ell_{M,N}$ we have $\ell_{M,N} = \ell_{K_1,N}$ and K_1 satisfies the required property. ∎

Theorem 2.9 *If* $M \Downarrow N[\![\ell = L]\!]$, *then (1)* $L \in \langle M \rangle$ *and (2) if* $\ell_{M,N} =< N, \ldots, K >$, *then* $K \in \langle M \rangle$.

PROOF: The proof of (1) is trivial, since by definition $N \in \langle M \rangle$. For (2), we proceed by transition induction.

Case 1: $M = N \Downarrow N[\![\ell = L]\!]$, obvious.

Case 2: $M = K[f \Leftarrow A] \Downarrow N[\![\ell = L]\!]$, where K is a normal form. Since $\ell_{M,N} =< N, K >$, the claim is obvious.

Case 3: $M = M_1[f \Leftarrow A] \Downarrow N[\![\ell = L]\!]$, where M_1 is not a normal form. We have $\ell_{M,N} =< N >^\frown \ell_{M_1,N_1} =< N, N_1, \ldots, K >$ and by induction $K \in \langle M_1 \rangle \subseteq \langle M \rangle$.

Case 4: $M = M_1.f \Downarrow N[\![\ell = L]\!]$ and the reduction is by the selection rule

$$\frac{M_1 \Downarrow N_1[\![f = B]\!] \qquad B \Downarrow N[\![\ell = L]\!]}{M = M_1.f \Downarrow N[\![\ell = L]\!]}$$

We have $\ell_{M,N} = \ell_{B,N} =< N, \ldots, K >$. By induction, $K \in \langle B \rangle$. By part 1, $B \in \langle M \rangle$ and so $K \in \langle M \rangle$. ∎

The meaning and use of the preceding technical clarifications is that when an update operation $M[\ell \Leftarrow A]$ is performed, then if $M[\ell \Leftarrow A] \Downarrow N$ and $\ell_{M[\ell \Leftarrow A],N} =< N, \ldots, K >$, then N (the value of the updated term) results by modifying the normal form K of the workspace, by an application of finitely many operations of update.

2.5 A Transition System of Workspaces

Agents can interact with their workspaces in two different ways, namely by (1) observing the value of some attribute of an object, by a field extraction operation, which naturally leaves the workspace unaffected, even though some side computation may be needed in evaluating the attribute, and (2) by modifying some object in the workspace. In the latter case, the effect of the modification may propagate in the entire workspace. Thus, an operation on a workspace U is merely an update of some member of U. More precisely,

Definition 2.10 *1. If $M : \tau, L : \tau' \in U$ are closed and well-typed terms and, in addition, $\trianglerighteq(M : \tau)[\ell \Leftarrow L : \tau'] : \tau$, then (omitting the type annotations, for simplicity of expression) $\alpha_{M,\ell,L} = M[\ell \Leftarrow L]$ is an operation of type τ on U. We write $\mathtt{act}(U)$ for the set of operations on the workspace U.*

2. If $M[\ell \Leftarrow L] \Downarrow N$ and $\ell_{M[\ell \Leftarrow L],N} =< N, \ldots, K >$, then the workspace produced by applying the operation, denoted by $U\{\alpha\}$ (omitting the subscripts, for simplicity), is the space $\langle \{A[N/K] |\ A \in U\} \rangle$, where the replacement operation is defined by

- $x[N/K] = x$ *and* $c[N/K] = c$
- $[(\ell_i = A_i)_{i=1,\ldots,n\ (n \geq 0)}][N/K] = [(\ell_i = A_i[N/K])_{i=1,\ldots,n\ (n \geq 0)}]$
- $(M.\ell)[N/K] = (M[N/K]).\ell$
- $(M[\ell \Leftarrow B])[N/K] = (M[N/K])[\ell \Leftarrow B[N/K]]$

3. We also write $U \xrightarrow{\alpha} V$, if $V = U\{\alpha\}$.

Note that by Theorem 2.9, the normal form K of the Definition belongs to the workspace U and that, by Subject Reduction, it is of the same type as N.

Lemma 2.11 *If U is a finitely generated workspace and $\alpha \in \mathtt{act}(U)$, then $U\{\alpha\}$ is also finitely generated.*

PROOF: Obvious, from the definition. ∎

Lemma 2.12 *Let V be a subspace of U, $V \preceq U$. If α is an operation on V and $V\{\alpha\} = V'$ is defined, then $U\{\alpha\} = U'$ is defined and $V' \preceq U'$.*

PROOF: The hypothesis that $\alpha = M[\ell \Leftarrow L]$, for some M, ℓ, L, is defined on V means that $M, L \in V \subseteq U$, hence the operation is also defined on U. The claim that $V\{\alpha\} \preceq U\{\alpha\}$ follows from the way the image of a workspace is defined. ∎

Thus if $A = \langle \Delta, \delta, \kappa \rangle$ is a U-agent, hence $\kappa | U = V \preceq U$, and A uses objects M, L in its capabilities subspace V to perform the operation $\alpha = M[\ell \Leftarrow L]$, then $A\{\alpha\}$, after evaluation, becomes a $U\{\alpha\}$-agent without apparent effect in its internal state, though its concrete capabilities $\kappa | U\{\alpha\}$ may well be different.

2.6 Examples

Example 1 Let R be a rectangle object,

$$R = [topLeft = P, bottomRight = Q, area = (P.ypos - Q.ypos) * (Q.xpos - P.xpos)]$$

where P, Q are point objects,

$$P = [xpos = 1, ypos = 14] \text{ and } Q = [xpos = 7, ypos = 0]$$

Let $U = \langle R \rangle$ be the workspace generated by the rectangle. Then $\alpha = P[xpos \Leftarrow P.xpos + 3]$ is an operation defined on the workspace and which, if executed, results in moving the point P to the right, by 3 units. This change affects the entire workspace, since the rectangle object contains a reference to the point P, and so its relative position, size, area and perimeter also change. Given the computation $M = P[xpos \Leftarrow P.xpos + 3] \Downarrow [xpos = 4, ypos = 14] = N$, the sequence $\ell_{M,N}$ is simply $< N, P >$. By our definition

$$U\{\alpha\} = \langle \{A[N/P] |\ A \in U\} \rangle$$

so that if

$$T = R[N/P] = [topLeft = N, bottomRight = Q, area = (N.ypos - Q.ypos) * (Q.xpos - N.xpos)]$$

then $U\{\alpha\} = \langle T \rangle$. ∎

Example 2 Suppose, in the previous example, that the objects are colored, where we set $R.color = Q.color = P.color$ and the color attribute of P is initially set to "white". Repainting P, by the operation $P[color \Leftarrow \text{"}blue\text{"}]$, results in the obvious modification of the colors of all objects in the workspace. ∎

2.7 A Language of Properties of Workspaces

An agent situated in some environment forms beliefs about properties of its environment. These may be beliefs about properties of objects in its environment, or beliefs about ways in which the environment can change, as a result of actions performed and affecting this

environment. We specify here a language of workspace properties that can be used by an agent to reason about its environment. This same language can be also used to express persistent properties (integrity constraints) of the environment (such as, for example, that no two points can have the same coordinates while being in the same workspace), ones that limit the actions an agent can perform, while trying to transform its environment in ways that better suit its own desires.

We define a many-sorted first-order modal language $\mathcal{L}$ as follows: Sorts are types and the set of terms is exactly the set of well-typed object terms. This includes typed variable terms $x : \tau$, all ground terms, i.e. constants and any terms of the form $op(\bar{M})$ that may have been included in the syntax of objects. Atomic statements are built with standard typed predicates like $=, \leq, <, >, \geq$: $\texttt{int} \times \texttt{int}$ for numbers and similarly for strings, and perhaps additional typed predicates depending on the intended application domain. Modal statements are built with modalities $\langle\alpha\rangle$ indexed by actions, where we assume that actions compose, so that if α_1, α_2 are actions, then each of $\langle\alpha_1\rangle$, $\langle\alpha_2\rangle$ and $\langle\alpha_1; \alpha_2\rangle$ is a valid modality. Terms and formulae are defined by the recursive scheme

$$\begin{array}{rcl} t & ::= & M : \tau \\ \varphi & ::= & R(t_1 : \tau_1, \ldots, t_n : \tau_n) \mid \neg\varphi \mid \varphi \wedge \varphi \mid (\exists x : \tau)\varphi \mid \langle\alpha\rangle\varphi \end{array}$$

where R is a typed predicate and α is an operation on workspaces. For example, in our concrete workspace model of Section 2, $(M : \tau).f + (K : \tau').g \leq 5 : \texttt{int}$ is an atomic statement, though we will not always be as pedantic with type annotations.

Definition 2.13 *The well-formed formulae of the language are defined as follows*

- *If $R : \tau_1 \times \cdots \times \tau_n$ and $\trianglerighteq t_i : \tau_i$, then $R(t_1 : \tau_1, \ldots, t_n : \tau_n)$ is well formed*
- *If φ, ψ are well-formed, then so are $\neg\varphi, \varphi \wedge \psi, (\exists x : \tau)\varphi$*
- *If α is a valid atomic or composite action (workspace transformation) expression and φ is well-formed, then $\langle\alpha\rangle\varphi$ is also well-formed. In our concrete workspace model of Section 2, atomic actions α are record update terms and validity of the action expression $\alpha = (M : \tau)[\ell \Leftarrow L : \tau'] : \tau$ means that no free variables of ground type (integers, strings etc) occur in α and that $\trianglerighteq(M : \tau)[\ell \Leftarrow L : \tau'] : \tau$ (i.e. the object update expression is well-typed) .*

Formulae of the logical language express properties of workspaces. The semantics is given in a more or less standard compositional way, given an interpretation of predicates and function symbols and a type respecting interpretation σ of the variables as objects $[\![x : \tau]\!]_\sigma = \sigma(x) : \tau$ and $[\![c]\!]_\sigma = c$ for the constants. Operators on ground terms (function symbols), such as integer addition or multiplication, string concatenation etc, are interpreted in the standard way. If R is a predicate, then we let its interpretation in a workspace U be simply denoted by $R|_U$.

$$\begin{array}{lcll} U & \models_\sigma & R(t_1, \ldots, t_n) & \text{iff} \quad ([\![t_1]\!]_\sigma, \ldots, [\![t_n]\!]_\sigma) \in R|_U \\ U & \models_\sigma & \neg\varphi & \text{iff} \quad U \not\models_\sigma \varphi \\ U & \models_\sigma & \varphi \wedge \psi & \text{iff} \quad U \models_\sigma \varphi \text{ and } U \models_\sigma \psi \\ U & \models_\sigma & (\exists x : \tau)\varphi & \text{iff} \quad (\exists M : \tau \in U)(\ U \models_\sigma \varphi[M/x]\ ) \\ U & \models_\sigma & \langle\alpha\rangle\varphi & \text{iff} \quad (\exists V)(\ U \xrightarrow{\alpha[\sigma]} V \text{ and } V \models_\sigma \varphi\ ) \end{array}$$

In our concrete workspace model, actions α are terms of the logical language and may therefore be open (though only closed action terms can be workspace transformations). This implies that, in $(\exists x : \tau)\langle\alpha\rangle\varphi$, quantification can capture a variable in the modality as well as in φ. It also explains the rule for the satisfaction of modal formulae, where the transition is by the action $\alpha[\sigma]$, the closure of α by the substitution σ. Disjunctions, the universal quantifier and "necessity" operators $[\alpha]$ can be introduced in the usual way.

Remark 2.14 *Note that, because of our hypothesis that workspace transformations are functional (if there exists a V such that $U \xrightarrow{\alpha} V$, then $V = U\{\alpha\}$ is unique), it follows that there is a semantic equivalence $\langle\alpha\rangle(\varphi \wedge \psi) \equiv \langle\alpha\rangle\varphi \wedge \langle\alpha\rangle\psi$.*

In addition, because actions compose, every formula of the type $\langle\alpha_1\rangle \cdots \langle\alpha_n\rangle\varphi$ is semantically equivalent to the formula $\langle\alpha_1; \cdots; \alpha_n\rangle\varphi$.

Finally, operators $[\alpha]$ can be defined in the usual way, with resulting semantic clause for satisfaction

$$U \models_\sigma [\alpha]\varphi \quad \textit{iff} \quad (\forall V)(U \xrightarrow{\alpha} V \implies V \models_\sigma \varphi)$$

Thus $U \models_\sigma [\alpha]\varphi$ if either $\alpha \notin \mathtt{act}(U)$, or else $U\{\alpha\} \models_\sigma \varphi$. It then follows that if $\alpha \in \mathtt{act}(U)$, then $U \models_\sigma \langle\alpha\rangle\varphi$ iff $U \models_\sigma [\alpha]\varphi$. ∎

3 A Language and Semantics for Situated Agent Systems

It is perhaps fair to say that most researchers would broadly agree today that

1. Agents are the *hosts (subjects) of diverse mental (propositional) attitudes* (beliefs, commitments, desires, intentions, obligations, fears, uncertainties etc). Distinct mental attitudes may have the same propositional content but enjoy a different *functional status*, affecting the agent's behavior in different ways

2. Agents are the *subjects of actions* that they perform and which result in possibly modifying the environments they live in. What actions an agent performs depends on what *capabilities* it has. On the other hand, what agent capabilities are actually realizable is something that varies with the particular environment the agent finds itself in.

3. Agents are also *centers of perception*, they are endowed with perhaps diverse representation capabilities that allow them to form representations of both the state the environment is in and the state they, themselves, are in. The environment is observable via the agent's sensors. Ground facts about the environment, gathered by direct observation, form the basis for learning and thus agents as perceivers are thereby also *subjects of learning* and of knowledge accumulation

4. Agents are *deliberating subjects*, they exhibit what can be conceived as logical reasoning behavior, which vastly underlies their decision-making

5. Agents and their environment are never in a perfect equilibrium state, there are tensions arising from the fact that what an agent needs (desires) is not automatically catered for by the environment. Thus agents are, or must be, efficient *problem solvers*,

scrupulously *planning* solutions for detected (perceived) problems, given their action capabilities

6. Agents are *social beings*, they are *subjects of communication acts*, which includes both various types of *speech act* (assertions, queries, requests, confirmations, denials or rejections etc), as well as various kinds of *social exchange*, whose content is not informational but physical (handing over or receiving *objects*, to be consumed or used as tools by some other agent)

Some of the features mentioned above are aspects of the *agent state*, whereas the rest are distinctive *agent behaviors*. Mental attitudes (beliefs, desires, fears etc) and capabilities are aspects of the agent state. Perceiving and learning, deliberating and planning, communicating etc are agent behaviors. As in [4, 5, 6], we propose to proceed in a modular fashion, examining some features of agency and leaving it for further research to examine other features or compose partial approaches. However, rather than focusing exclusively on beliefs, as in [4, 5, 6], we propose and study a calculus of agents whose state is determined by (1) *beliefs*, (2) *desires* and (3) *capabilities*. We regard intentions and commitments as special agent behaviors, as we explain in the sequel.

Beliefs and desires are expressed by formulae of the language of properties of workspaces we have presented and which depends on having a formal account of a workspace transition system. The workspace inhabited by agents is a model-structure that validates or invalidates their beliefs, desires, intentions or commitments. To satisfy a goal, agents attempt to reconstruct the workspace so as to obtain a model of the formula expressing the goal. We view *actions as modifications of properties of objects in the environment* (pushing an object to another location, filling in or emptying a container object, painting a colorable object, turning a switch on or off etc). And we view *capabilities as expertise to use certain types of objects*. Thus capabilities are expressed in the agent state as a collection of types (of objects the agent can use). More precisely, we assume that capabilities always include ground types (any agent can manipulate numbers, strings and other ground objects) and if $\rho = [\ell_1 : \tau_1, \ldots, \ell_m : \tau_m]$ is a capability type for agent A, then each of the τ_i is one, too. Note that, given capabilities $\bar{\rho}$, it is the particular environment (workspace) which then provides, or fails to provide, concrete objects of the given types for the agent to use. If $\bar{\rho} = \rho_1, \ldots, \rho_n$ are row types and U is a workspace, then the capabilities types $\bar{\rho}$ determine a subspace

$$\bar{\rho}|U = \langle\ \{M \in U|\ \trianglerighteq M : \rho_i, \text{ for some } i = 1, \ldots, n\}\ \rangle$$

which we refer to as the *capabilities subspace* of the agent A in the workspace U. Conversely, too, any finitely generated subspace

$$K = \langle M_1 : \rho_1, \ldots, M_n : \rho_n \rangle \preceq U$$

determines uniquely the capabilities $\bar{\rho} = \rho_1, \ldots, \rho_n$. Note that, when an agent migrates from one workspace to another, its concrete capabilities subspaces may be different in the two distinct environments (an architect at the building site maintains his capabilities to use drawing equipment to draw a floor plan, though he cannot exercise such capabilities at his current workspace). Learning a new capability (perhaps by acquiring sufficient information on a certain type of object) is an issue we will not deal with in this report.

3.1 A Language for Agent Systems

Definition 3.1 (Situated Agent Systems) *Fix a finitely generated workspace U. Situated U-agent systems are defined by*

$$\begin{array}{rrl} \textit{Single Agents:} & A & := \texttt{nil} \mid \langle \Delta, \delta, \kappa \rangle \mid A\{p\} \\ \textit{Agent Systems:} & \mathbf{A} & := A \mid \mathbf{A} \| \mathbf{A} \\ \textit{Situated Systems:} & \mathcal{A} & := (U, \mathbf{A})_\Theta \end{array}$$

where

- *p is a behavior, defined by the syntax of Table 3*
- `nil` *is the nil agent, whose only function is to be a message-holder in asynchronous communication,*
- *$\langle \Delta, \delta, \kappa \rangle$ is a basic agent (often denoted in the rest of this paper as simply the agent $\Delta\delta\kappa$) with*
  - *belief base $\Delta = \varphi_1, \ldots, \varphi_n$, where we assume that Δ is consistent*
  - *desires $\delta = \psi_1, \ldots, \psi_n$, for some n, where we also assume that δ is consistent and that its members are deductively independent ($i \neq j$ implies $\psi_i \nvdash \psi_j$) and*
  - *capabilities $\kappa = \bar{\rho}$, determining a subspace $\kappa|U = K \preceq U$ (a workspace of tools (objects) the agent can use).*
- *Finally, $A\{p\}$ is an agent intending (to behave like) p, as soon as prior behavior commitments have been discharged. In $A\{p\}$, where $A = \langle \Delta, \delta, \kappa \rangle$ is a basic agent, the behavior p may be a sequence $p = p_1; \cdots; p_n$ and we think of it as the* action queue *of the agent.*
- *An agent system is either a single agent A, or the concurrent composition $\mathbf{A} \| \mathbf{A}$ of two systems.*
- *A situated agent system is a system of agents $\mathbf{A}$ living in a common workspace U. Θ is a reasoning theory in a language that may extend the logical language we have presented by the addition of belief $\mathcal{B}$, desire $\mathcal{D}$ etc modalities and appropriate reasoning axioms and rules. We usually leave Θ implicit, writing simply $(U, \mathbf{A})$.*

The language of agent behaviors p is specified by the syntax in Table 3, where we assume a countable stock of recursion constants p, definable by equations of the form $p = q(p)$, or, making the agent parameter explicit, $A\{p\} = A\{q(p)\}$. We also make the convention to write $\varphi?p$ and $\varphi??p$ when the `else` action q is simply `skip`.

$$\varphi?p \equiv \varphi?p \texttt{ else skip} \qquad \varphi??p \equiv \varphi??p \texttt{ else skip}$$

Some more explanations on the syntax of agents follow.

The idle behavior `skip` is discharged without any effect to the agent state.

The only behaviors modifying the workspace are the workspace operations α. Those are object update actions, $(M : \tau)[\ell \Leftarrow L : \tau_1] : \tau$ (modifications of properties of objects in the

Table 3: Agent Behaviors in a Multi-Agent Environment

$p =$

	ATOMIC BEHAVIORS
`skip`,	idle action, denoting termination;
α,	action transforming the common workspace;
$\texttt{obs}(M_i : \tau_i)_{i \in n}$,	concurrent observations of the values (if any) of objects ;
$a![\varphi]$,	assert φ on channel a;
$r![\varphi]$,	request that action be taken so that φ becomes true;
$c?(\psi)p$,	receive a message from channel c, then behave like p;
$\texttt{revise}(\bar{\theta})$,	belief revision utility, given fresh observations $\bar{\theta}$;
$\texttt{update}(\delta)$,	goal updating, replacing old goals with new;
	COMPOUND BEHAVIORS
ψ,	commitment to bring it about that ψ;
$\varphi?\ p\ \texttt{else}\ q$,	logical query based conditional;
$\varphi??\ p\ \texttt{else}\ q$,	goal based conditional;
$p + p$,	choice;
$p \mid p$,	parallel;
$p; p$,	sequential behavior.
	RECURSIVE BEHAVIORS
$(A\{p\} = A\{q(p)\})$,	recursive behavior p, with defining equation $p = q(p)$;

agent's environment, discussed in Section 2). What actions can be performed by what agent depends on the agent's capabilities, in the sense that the objects involved in the action must be objects of types that the agent is capable of handling. This restriction is captured in the operational semantics of our language, see rule (α). As actions are designated by object expressions, action-patterns can be expressed in our framework by open expressions.

We say that an agent of the form $A\{p; \psi\}$ *intends* to bring it about that ψ. After the behavior p is discharged, leading to a possibly revised agent state A', the agent $A'\{\psi\}$ becomes *committed* to bringing it about that ψ.

This distinction between intentions and commitments (intentions refer to the future, whereas commitments are for now) is not sharp, since intentions may appear as delayed (conditional) commitments. For example, the behavior $p = \varphi?\psi\ \texttt{else}\ p$ is an intention to (try and) bring it about that ψ, as soon as some condition φ is believed to hold. Thus $A\{p\}$ is the agent committed to bringing it about that ψ, as soon as the precondition φ holds, which is hardly different from intending that ψ, provided that φ (being now committed to entertaining my child, provided he is back from school, is no different from intending to entertain him, when he comes back from school).

Incidentally, an intention expressed with a recurrent behavior like p above is unconditionally persistent. However, if the agent ever comes to the conclusion that the condition will never get satisfied, or that it is too costly to pursue it, or that its significance, given

new circumstances, is not great, it is rational that it should drop the intention. This is better captured with a behavior of the form

$$p = \varphi?\psi \texttt{ else } (\texttt{skip} + p)$$

For example, if I intend to spend my next vacations in Hawai, provided I have saved enough money for the expenses, then I should persist with my intention until I do indeed have enough money saved for it. If, however, I ever become convinced that there will always come up a more urgent need, devoring my savings, or if it so happens that Trans-Atlantic airlines offers me a vacation in Tenerife, then it is rational that I should drop my intention to spend my next vacations in Hawai.

Intentions or commitments are handled by invoking the reasoning module, which attempts to elaborate a plan for satisfying the intention. A plan is a sequence of atomic actions $\alpha_1; \cdots; \alpha_n$ such that, if $U = U_1$ is the current workspace of the agent and if κ are its capabilities, then $\kappa | U_i = K_i \xrightarrow{\alpha_i} K_{i+1}$, and $U_i \xrightarrow{\alpha_i} U_{i+1}$, for $i = 1, \ldots, n$. We will not endeavor in this report on how the planner actually works.

In defining rules for the operational semantics of our language we assume a belief revision utility $\texttt{revise}(\Delta, \bar{\theta})$, revising an old belief base Δ, given fresh observations $\bar{\theta}$. In the subsequent discussion of the rules we give a detailed discussion of how the belief base is revised.

Goal selection and updating is a utility very much depending on the design objectives and application details, though some general principles for goal updating (e.g. maximizing a utility function) have been proposed and investigated. We think that, in the context of our approach in this report, we may simply assume some update utility, revising the agent's desires, without going into any details on the subject.

For communication, we restrict to only two types of communication acts, assertions and requests, and we use, for simplicity, two different types of communication channel, ranged over by channel names a, a', a_i etc (for assertions) and r, r', r_j etc (for requests), thus avoiding to have to devise an extension of our logical language to a communication language, with explicit speech act operators. We use c for a communication channel when it is not relevant whether assertions or requests are meant.

Compound agent behaviors include recursively defined behaviors. Since a behavior, from a denotational point of view, is merely a function from agent states to agent states, recursively defined behaviors are simply recursively defined functions on agent states, definable by equations of the form $A\{p\} = A\{q(p)\}$. We make the usual assumption that p is guarded in q, thus disallowing "definitions" of recursive behaviors with equations of the form $A\{p\} = A\{p\}$. The guardedness assumption in our context is simply that $q(p) \neq p$, while any other behavior context is allowed for q. With recursive behaviors available, familiar constructs such as while loops can be easily defined, in a standard way.

Note that we have allowed for extended parallelism in our language, both at the level of concurrently executing behaviors and at the level of concurrently executing agents. The former is given an operational semantics by interleaving basic (atomic) behaviors.

3.2 Operational Semantics

Axioms and rules of the operational semantics are presented in Tables 4 and 5. Comments and explanations for each axiom or rule follow. The general form of a transition

is $(U, \mathbf{A}) \xrightarrow{e} (V, \mathbf{B})$, where e is empty (a silent transition) or an observable action (a workspace transformation α, or a message-send action $c![\vartheta]$), U, V are workspaces and $\mathbf{A}, \mathbf{B}$ are agent systems. Most of the rules refer to single-agent transformations.

Discharging Behavior Commitments: First, behavior commitments are discharged and fulfilled as soon as any prior behavior commitment has been discharged. Given that we view behaviors as functions on agent states, this simply means that to evaluate $\Delta\delta\kappa\{p; q\}$, one first needs to evaluate $\Delta\delta\kappa\{p\} = \Delta'\delta'\kappa'$ and then feed this as an argument to the subsequent behavior q, as in $\Delta'\delta'\kappa'\{q\}$. Reduction to basic agents before a behavior commitment is fulfilled follows the rules (D1)-(D3).

In all of the following cases $A = \langle \Delta, \delta, \kappa \rangle$ is a basic U-agent, denoted, for transparency of notation, as $\Delta\delta\kappa$.

skip: The action `skip` is an idle action, $(U, \Delta\delta\kappa\{\texttt{skip}\}) \longrightarrow (U, \Delta\delta\kappa)$

Workspace Transformations: If the agent performs an operation on the workspace, then this must be by updating an object in its capabilities space, where the update is again in the capabilities space $\kappa|U = K \preceq U$. The effect of the action, if it is defined on the capabilities space, is to provoke a transformation of the surrounding workspace of the agent: $(U, \Delta\delta\kappa\{\alpha\}) \longrightarrow (U\{\alpha\}, \Delta\delta\kappa)$. This transition depends on the possibility of the transformation of the capabilities space and so it is best expressed as a rule:

$$\frac{\kappa|U = K \xrightarrow{\alpha} K' \quad U \xrightarrow{\alpha} U'}{(U, \Delta\delta\kappa\{\alpha\}) \xrightarrow{\alpha} (U', \Delta\delta\kappa)}$$

We also write $\trianglerighteq \alpha : \kappa$ to indicate that α is a well-typed term and that all the types of objects in α are in the capabilities space κ.

Intentions, Commitments and Planning: A commitment to a goal, as in $\Delta\delta\kappa\{\psi\}$, is discharged by calling the planner, which returns some plan p. A plan is just a behavior, which may be simply a sequence of actions, or (if the planning task fails) the idle action `skip` for dropping the commitment, or even a request action $r![\psi]$ for requesting help in establishing ψ. The relevant rule is the rule (int), which presumes an available planning utility `plan`, the details of which we shall leave unspecified in this report. We do presume some normalizing assumptions, however, such as that deductively equivalent goals lead to the same planning behavior.

Observing Objects: Agents form new beliefs or update existing ones by interacting with their environment, by perceiving it through their sensors. The behavior `obs` corresponds to ground observations where an agent directly observes the value of an object in its workspace (and not only in its capabilities space). New observations may contradict existing beliefs. Thus observing gives rise to a belief revision behavior.

$$\frac{M_i \in U \quad M_i \Downarrow N_i \quad (i \in n > 0)}{(U, \Delta\delta\kappa\{\texttt{obs}(M_i : \tau_i)_{i\in n}\}) \longrightarrow (U, \Delta\delta\kappa\{\texttt{revise}(M_i =_{\tau_i} N_i)_{i\in n})\})}$$

Table 4: Operational Semantics I: Axioms

(skip)	$(U, \Delta\delta\kappa\{\texttt{skip}\} \longrightarrow (U, \Delta\delta\kappa)$	
(α)	$(U, \Delta\delta\kappa\{\alpha\}) \xrightarrow{\alpha} (V, \Delta\delta\kappa)$	Provided $\trianglerighteq\alpha : \kappa$ and $U \xrightarrow{\alpha} V$
(obs0)	$(U, \Delta\delta\kappa\{\texttt{obs}(M_i : \tau_i)_{i\in 0}\}) \longrightarrow (U, \Delta\delta\kappa)$	
(obs)	$(U, \Delta\delta\kappa\{\texttt{obs}(M_i : \tau_i)_{i\in n>0}\}) \longrightarrow (U, \Delta\delta\kappa\{\texttt{revise}(M_i =_{\tau_i} N_i)_{i\in n>0}\})$	Provided $M_i : \tau_i \in U$ and $M_i \Downarrow N_i$, for each $i \in n > 0$
(upd)	$(U, \Delta\delta\kappa\{\texttt{update}(\delta')\}) \longrightarrow (U, \Delta\delta'\kappa)$	
(rev)	$(U, \Delta\delta\kappa\{\texttt{revise}(\bar{\vartheta})\}) \longrightarrow (U, \Delta'\delta\kappa)$	Provided $\texttt{revise}(\Delta, \bar{\vartheta}) = \Delta'$
(query1)	$(U, \Delta\delta\kappa\{\varphi?p\ \texttt{else}\ q\}) \longrightarrow (U, \Delta\delta\kappa\{p\})$	Provided $\Delta \vdash_\Theta \varphi$
(query2)	$(U, \Delta\delta\kappa\{\varphi?p\ \texttt{else}\ q\}) \longrightarrow (U, \Delta\delta\kappa\{q\})$	Provided $\Delta \nvdash_\Theta \varphi$
(goal1)	$(U, \Delta\delta\kappa\{\vartheta??p\ \texttt{else}\ q\}) \longrightarrow (U, \Delta\delta\kappa\{p\})$	Provided $\vartheta \vdash_\Theta \delta_{i\ (i=1,or\ldots,or\ i=n)}$
(goal2)	$(U, \Delta\delta\kappa\{\vartheta??p\ \texttt{else}\ q\}) \longrightarrow (U, \Delta\delta\kappa\{q\})$	Provided $\vartheta \nvdash_\Theta \delta_{i\ (i=1,\cdots,i=n)}$
(+1)	$(U, \Delta\delta\kappa\{p + q\}) \longrightarrow (U, \Delta\delta\kappa\{p\})$	
(+2)	$(U, \Delta\delta\kappa\{p + q\}) \longrightarrow (U, \Delta\delta\kappa\{q\})$	
(Rec)	$(U, \Delta\delta\kappa\{p\}) \longrightarrow (U, \Delta\delta\kappa\{q(p)\})$	Provided p is defined by $p = q(p)$
(int1)	$(U, \Delta\delta\kappa\{\psi\}) \longrightarrow (U, \Delta\delta\kappa)$	Provided $\Delta \vdash_\Theta \psi$
(int)	$\dfrac{\texttt{plan}(\Delta, \kappa, \psi) = p}{(U, \Delta\delta\kappa\{\psi\}) \longrightarrow (U, \Delta\delta\kappa\{p\})}$	
(CAss)	$(U, \Delta\delta\kappa\{a![\vartheta]\} \xrightarrow{a![\vartheta]} (U, \Delta\delta\kappa \| a![\vartheta])$	Provided $\Delta \vdash_\Theta \vartheta$
(CReq)	$(U, \Delta\delta\kappa\{r![\vartheta]\} \xrightarrow{r![\vartheta]} (U, \Delta\delta\kappa \| r![\vartheta])$	Provided $\vartheta \vdash_\Theta \delta_{i\ i=1,or\cdots or\ i=n}$
(Msg)	$(U, \Delta\delta\kappa\{c?(\psi)p\} \| c![\vartheta]) \longrightarrow (U, \Delta\delta\kappa\{p[\vartheta/\psi]\})$	

Table 5: Operational Semantics II: Rules

(D1) $\dfrac{(U,A) \longrightarrow (U,B)}{(U,A\{p\}) \longrightarrow (U,B\{p\})}$ (D2) $\dfrac{(U,A) \xrightarrow{\alpha} (V,B)}{(U,A\{p\}) \xrightarrow{\alpha} (V,B\{p\})}$

(D3) $\dfrac{(U,A) \xrightarrow{c![\vartheta]} (U,B\|c![\vartheta])}{(U,A\{p\}) \xrightarrow{c![\vartheta]} (U,B\{p\}\|c![\vartheta])}$

(Seq1) $\dfrac{(U,A\{p\}) \longrightarrow (U,A'\{p'\})}{(U,A\{p;q\}) \longrightarrow (U,A'\{p';q\})}$ (Seq2) $\dfrac{(U,A\{p\}) \xrightarrow{\alpha} (U',A\{p'\})}{(U,A\{p;q\}) \xrightarrow{\alpha} (U',A\{p';q\})}$

(SeqAt1) $\dfrac{(U,A\{p\}) \longrightarrow (U,\Delta\delta\kappa)}{(U,A\{p;q\}) \longrightarrow (U,\Delta\delta\kappa\{q\})}$ (SeqAt2) $\dfrac{(U,A\{p\}) \xrightarrow{\alpha} (U',\Delta\delta\kappa)}{(U,A\{p;q\}) \xrightarrow{\alpha} (U',\Delta\delta\kappa\{q\})}$

(Seq!) $\dfrac{(U,A\{p\}) \xrightarrow{c![\vartheta]} (U,A'\{p'\}\|c![\vartheta])}{(U,A\{p;q\}) \xrightarrow{c![\vartheta]} (U,A'\{p';q\}\|c![\vartheta])}$ (Seq!At) $\dfrac{(U,A\{p\}) \xrightarrow{c![\vartheta]} (U,\Delta\delta\kappa)}{(U,A\{p;q\}) \xrightarrow{c![\vartheta]} (U,\Delta\delta\kappa\{q\}\|c![\vartheta])}$

(ParL1) $\dfrac{(U,A\{p\}) \longrightarrow (U,A'\{p'\})}{(U,A\{p|q\}) \longrightarrow (U,A'\{p'|q\})}$ (ParR1) $\dfrac{(U,A\{q\}) \longrightarrow (U,A'\{q'\})}{(U,A\{p|q\}) \longrightarrow (U,A'\{p|q'\})}$

(ParAtL1) $\dfrac{(U,A\{p\}) \longrightarrow (U,\Delta\delta\kappa)}{(U,A\{p|q\}) \longrightarrow (U,\Delta\delta\kappa\{q\})}$ (ParAtR1) $\dfrac{(U,A\{q\}) \longrightarrow (U,\Delta\delta\kappa)}{(U,A\{p|q\}) \longrightarrow (U,\Delta\delta\kappa\{p\})}$

(ParL2) $\dfrac{(U,A\{p\}) \xrightarrow{\alpha} (U',A'\{p'\})}{(U,A\{p|q\}) \xrightarrow{\alpha} (U',A'\{p'|q\})}$ (ParR2) $\dfrac{(U,A\{q\}) \xrightarrow{\alpha} (U',A'\{q'\})}{(U,A\{p|q\}) \xrightarrow{\alpha} (U',A'\{p|q'\})}$

(ParAtL2) $\dfrac{(U,A\{p\}) \xrightarrow{\alpha} (U',\Delta\delta\kappa)}{(U,A\{p|q\}) \xrightarrow{\alpha} (U',\Delta\delta\kappa\{q\})}$ (ParAtR2) $\dfrac{(U,A\{q\}) \xrightarrow{\alpha} (U',\Delta\delta\kappa)}{(U,A\{p|q\}) \xrightarrow{\alpha} (U',\Delta\delta\kappa\{p\})}$

(Par!L1) $\dfrac{(U,A\{p\}) \xrightarrow{c![\vartheta]} (U,A'\{p'\}\|c![\vartheta])}{(U,A\{p|q\}) \xrightarrow{c![\vartheta]} (U,A'\{p'|q\}\|c![\vartheta])}$ (Par!R1) $\dfrac{(U,A\{q\}) \xrightarrow{c![\vartheta]} (U,A'\{q'\}\|c![\vartheta])}{(U,A\{p|q\}) \xrightarrow{c![\vartheta]} (U,A'\{p|q'\}\|c![\vartheta])}$

(Par!L2) $\dfrac{(U,A\{p\}) \xrightarrow{c![\vartheta]} (U,\Delta\delta\kappa\|c![\vartheta])}{(U,A\{p|q\}) \xrightarrow{c![\vartheta]} (U,\Delta\delta\kappa\{q\}\|c![\vartheta])}$ (Par!R2) $\dfrac{(U,A\{q\}) \xrightarrow{c![\vartheta]} (U,\Delta\delta\kappa\|c![\vartheta])}{(U,A\{p|q\}) \xrightarrow{c![\vartheta]} (U,\Delta\delta\kappa\{p\}\|c![\vartheta])}$

(SParL1) $\dfrac{(U,\mathbf{A}) \longrightarrow (U,\mathbf{A}')}{(U,\mathbf{A}\|\mathbf{B}) \longrightarrow (U,\mathbf{A}'\|\mathbf{B})}$ (SParR1) $\dfrac{(U,\mathbf{B}) \longrightarrow (U,\mathbf{B}')}{(U,\mathbf{A}\|\mathbf{B}) \longrightarrow (U,\mathbf{A}\|\mathbf{B}')}$

(SParL2) $\dfrac{(U,\mathbf{A}) \xrightarrow{\alpha} (U',\mathbf{A}')}{(U,\mathbf{A}\|\mathbf{B}) \xrightarrow{\alpha} (U',\mathbf{A}'\|\mathbf{B})}$ (SParR2) $\dfrac{(U,\mathbf{B}) \xrightarrow{\alpha} (U',\mathbf{B}')}{(U,\mathbf{A}\|\mathbf{B}) \xrightarrow{\alpha} (U',\mathbf{A}\|\mathbf{B}')}$

(SPar!L) $\dfrac{(U,\mathbf{A}) \xrightarrow{c![\vartheta]} (U,\mathbf{A}'\|c![\vartheta])}{(U,\mathbf{A}\|\mathbf{B}) \xrightarrow{c![\vartheta]} (U,\mathbf{A}'\|c![\vartheta]\|\mathbf{B})}$ (SPar!R) $\dfrac{(U,\mathbf{B}) \xrightarrow{c![\vartheta]} (U,\mathbf{B}'\|c![\vartheta])}{(U,\mathbf{A}\|\mathbf{B}) \xrightarrow{c![\vartheta]} (U,\mathbf{A}\|\mathbf{B}'\|c![\vartheta])}$

and when $n = 0$

$$\texttt{obs}(M_i)_{i\in 0} = \texttt{skip}$$

In Section 3.3.1 we generalize the observing behavior, allowing for observations triggered by any formula (the definition is by structural induction on the formula), thus obtaining a more complete account of perception.

Revising Beliefs: An agent may be activated with a minimal set of beliefs, including no observational facts, or communicated assertions, at all. The belief base of the agent is informed by (1) observe actions, or (2) assertions made by other agents. According to the operational semantics of our agent language, revisions due to observations attempt to assert into the belief base just two kinds of positive fact, namely (a) conjunctions of ground facts, of the form $\bigwedge_{i\in r}(M_i = v_i)$ (where v_i are values) and (b) conjunctions of modal ground facts $\bigwedge_i \langle\alpha\rangle(M_i = v_i)$. Facts of type (a) and (b) constitute the *observational core* of the belief base. Thus, a more refined representation of the belief base would be to distinguish

- Ground observational beliefs $\Delta_o = \bigwedge_{k\in s}(M_k = v_k)$,
- Ground modal observational beliefs Δ_{α_i}, for each workspace transforming action α_i for which there are any relevant beliefs, $\Delta_{\alpha_i} = \bigwedge_{j\in r}\langle\alpha_i\rangle(M_j = v_j)$
- Communication beliefs Δ_c

where the semantics of belief revision should guarantee that $\Delta_o \wedge (\bigwedge_{i\in n} \Delta_{\alpha_i}) \wedge \Delta_c = \Delta$ remains consistent.

In the view we are presenting here, beliefs in the belief base Δ are to be restricted to the above forms alone and any other belief is a result of deduction. For example, an agent may believe that the color of this ball is not red, just because there is a ground belief in the belief base that $\texttt{thisBall.color} = \mathit{blue}$. Or the agent may believe that there is a colored object in its environment, i.e. that $(\exists x : [\texttt{color} : \mathit{str}])(\exists y : \mathit{str})(x.\texttt{color} = y)$, just because the ground belief $\texttt{thisBall.color} = \mathit{blue}$ is in its belief base. Or, still, the agent may believe that colinear objects A, B are closer together than objects C, D, just because there are ground beliefs in the belief base, for example $\texttt{A.pos} = 2$, $\texttt{B.pos} = 3$ and $\texttt{C.pos} = 1$, $\texttt{D.pos} = 12$, so that $\Delta \vdash_\Theta \texttt{B.pos} - \texttt{A.pos} < \texttt{D.pos} - \texttt{C.pos}$.

Revising observational beliefs in this context is relatively straightforward, since we are here letting any beliefs about rules (implicational connections) be thrown into the non-revisable reasoning theory Θ. Learning new constraints, such as implicational connections, and revising the reasoning theory is an issue we do not deal with in this report.

In our restricted context, belief revision is then a rather straightforward matter. If we represent conjunctions set-theoretically, i.e. let $\{\theta_1, \ldots, \theta_n\}_{n\geq 0}$ stand for $(\theta_1 \wedge \cdots \wedge \theta_n)_{n\geq 0}$, then we can define

$$\texttt{revise}(\Delta_o, M = v) = \{M = v\} \cup (\Delta_o \setminus \{(M_i = v_i) \in \Delta_o |\ M_i = M\})$$

and

$$\texttt{revise}(\Delta_o, M = v, \bar{M} = \bar{v}) = \texttt{revise}(\texttt{revise}(\Delta_o, M = v), \bar{M} = \bar{v})$$

Similarly, for ground modal observations

$$\texttt{revise}(\Delta_\alpha, \langle\alpha\rangle(M = v)) = \{\langle\alpha\rangle(M = v)\} \cup (\Delta_\alpha \setminus \{\langle\alpha\rangle(M_i = v_i) \in \Delta_\alpha |\ M_i = M\})$$

and

$$\texttt{revise}(\Delta_\alpha, \langle\alpha\rangle(M = v), \langle\alpha\rangle(\bar{M} = \bar{v})) = \texttt{revise}(\texttt{revise}(\Delta_\alpha, \langle\alpha\rangle(M = v)), \langle\alpha\rangle(\bar{M} = \bar{v}))$$

Once ground observational beliefs and/or ground modal beliefs are revised, due to fresh observations $\bar{\theta}$, the agent will proceed to revise its communication beliefs $\Delta_c = \vartheta_1, \ldots, \vartheta_k$, rejecting ϑ_j if it conflicts with the revised observational belief base augmented with any ϑ_i, with $i < j$, already accepted, and accepting ϑ_j otherwise .

Assuming such a background belief revision utility $\texttt{revise}(\Delta, \bar{\theta})$, which revises the belief base Δ given fresh observations $\bar{\theta}$, we add the rule

$$\frac{\texttt{revise}(\Delta, \bar{\vartheta}) = \Delta'}{(U, \Delta\delta\kappa\{\texttt{revise}(\bar{\vartheta})\}) \longrightarrow (U, \Delta'\delta\kappa)}$$

Goal Updating: For goal updating we assume the axiom

$$(U, \Delta\delta\kappa\{\texttt{update}(\delta')\}) \longrightarrow (U, \Delta\delta'\kappa)$$

without going into the modalities of goal-selection.

Queries and Conditional: For the conditional $\vartheta?\ p$ `else` q, if the agent $A = \Delta\delta\kappa$ believes that ϑ is the case (i.e. if $\Delta \vdash_\Theta \vartheta$), then it will behave like p, else like q.

Satisfiability of Desires and Conditional: The conditional $\vartheta??p$ `else` q tests whether satisfying ϑ would further satisfaction of some goal ψ_i of the agent $A = \Delta\delta\kappa$, where $\delta = \psi_1, \ldots, \psi_n$, and selects a behavior appropriately.

$$\frac{\vartheta \vdash_\Theta \psi_{i\ (i=1,or\ldots,or\ i=n)}}{(U, \Delta\delta\kappa\{\vartheta??p\ \texttt{else}\ q\}) \longrightarrow (U, \Delta\delta\kappa\{p\})} \qquad \frac{\vartheta \nvdash_\Theta \psi_{i\ (i=1,\cdots,i=n)}}{(U, \Delta\delta\kappa\{\vartheta??p\ \texttt{else}\ q\}) \longrightarrow (U, \Delta\delta\kappa\{q\})}$$

Behavior Choice: Internal choice of behaviors can be resolved in either direction, nondeterministically and at run time.

$$(U, A\{p + q\}) \longrightarrow (U, A\{p\}) \qquad\qquad (U, A\{p + q\}) \longrightarrow (U, A\{q\})$$

Recursive Behaviors: If p is a recursive behavior defined by the equation $B\{p\} = B\{q(p)\}$, then for any agent A we have the corresponding unfolding axiom

$$(U, A\{p\}) \longrightarrow (U, A\{q(p)\})$$

While loops, such as

$$\texttt{While}\ \varphi\ \texttt{do}\ q\ \ \texttt{endWhile}$$

can be easily defined as the behavior

$$p = \varphi?\ (q; p)\ \texttt{else skip}$$

Sequenced Behaviors: For sequencing, the following natural rules apply

$$\text{(Seq1)}\quad \frac{(U, A\{p\}) \longrightarrow (U, A'\{p'\})}{(U, A\{p;q\}) \longrightarrow (U, A'\{p';q\})} \qquad \text{(Seq2)}\quad \frac{(U, A\{p\}) \xrightarrow{\alpha} (U', A\{p'\})}{(U, A\{p;q\}) \xrightarrow{\alpha} (U', A\{p';q\})}$$

$$\text{(SeqAt1)}\quad \frac{(U, A\{p\}) \longrightarrow (U, \Delta\delta\kappa)}{(U, A\{p;q\}) \longrightarrow (U, \Delta\delta\kappa\{q\})} \qquad \text{(SeqAt2)}\quad \frac{(U, A\{p\}) \xrightarrow{\alpha} (U', \Delta\delta\kappa)}{(U, A\{p;q\}) \xrightarrow{\alpha} (U', \Delta\delta\kappa\{q\})}$$

$$\text{(Seq!)}\quad \frac{(U, A\{p\}) \xrightarrow{c![\vartheta]} (U, A'\{p'\} \| c![\vartheta])}{(U, A\{p;q\}) \xrightarrow{c![\vartheta]} (U, A'\{p';q\} \| c![\vartheta])} \qquad \text{(Seq!At)}\quad \frac{(U, A\{p\}) \xrightarrow{c![\vartheta]} (U, \Delta\delta\kappa)}{(U, A\{p;q\}) \xrightarrow{c![\vartheta]} (U, \Delta\delta\kappa\{q\} \| c![\vartheta])}$$

In the first two rules the behavior p may be discharged by provoking some changes to the agent state or to the workspace, while leaving behind some residual behavior p'. The next two rules refer to the case where no residual behavior is left. Both possible cases need to be treated again for the case of output (the last two rules).

Concurrent Behaviors: Table 5 provides rules for reductions of concurrent behaviors, as well as for concurrent systems of agents. Both cases mentioned for sequencing may occur and for each of the left or right behaviors in $p|q$, hence the rather large number of needed rules.

Asynchronous Message Exchange: Communication is by asynchronous message passing, where messages are either assertions (information exchange) or requests (collaboration). A message, when passed asynchronously by an agent, is held by the message holder `nil`. Thus a sent message is represented by the agent $\texttt{nil}\{c![\vartheta]\}$, which we typically write, as is usual practice, as merely $c![\theta]$. Thus, if $A = \Delta\delta\kappa$ and a, r are, respectively, assertion and request channels, then we have the rules

$$\frac{\Delta \vdash_\Theta \vartheta}{(U, \Delta\delta\kappa\{a![\vartheta]\} \xrightarrow{a![\vartheta]} (U, \Delta\delta\kappa \| a![\vartheta])} \qquad \frac{\vartheta \vdash_{\Theta,\Delta} \bigvee \delta}{(U, \Delta\delta\kappa\{r![\vartheta]\} \xrightarrow{r![\vartheta]} (U, \Delta\delta\kappa \| r![\vartheta])}$$

The first rule means that agents are assumed to be truthful, as they never assert anything they do not believe to be true. By the second rule, agents never make random requests, they only request something that may increase the chances of satisfying their own desires.

A message can be consumed by any basic agent listening on the same channel

$$(U, \Delta\delta\kappa\{c?(\psi)p\} \| c![\vartheta]) \longrightarrow (U, \Delta\delta\kappa\{p[\vartheta/m]\})$$

To be more precise, writing messages in their full notation (making the message-holder explicit) we would have

$$(U, \Delta\delta\kappa\{c?(\psi)p\} \| \texttt{nil}\{c![\theta]\} \longrightarrow (U, \Delta\delta\kappa\{p[\vartheta/m]\} \| \texttt{nil})$$

However, it is a consequence of our operational semantics that $\texttt{nil} \| \mathbf{A}$ and $\mathbf{A}$ are essentially identical (behaviorally equivalent), as we explain in the sequel, and so there is no harm done by using the abbreviated notation for messages.

Concurrently Executing Agents: Other than the asynchronous communication axiom above, the rules (SPar), presented in Table 5, are the natural rules to adopt for concurrent systems.

3.3 Perceiving and Experimenting

Agents often need to decide on a course of action by testing whether some condition holds (and not merely by querying their belief base on the issue). Testing, or experimenting, involves the performance of ground observations, determined by the condition in question. In the sequel, we first generalize the behavior of observing to a behavior of performing observations triggered by a formula. Subsequently we introduce a testing behavior.

3.3.1 Observations triggered by Formulae

We define reduction rules for the action $\texttt{obs}(\phi)$, for a U-agent $A = \langle \Delta, \delta, \kappa \rangle$, by induction on the structure of ϕ

1. If $\phi = R\bar{t}$ is atomic, where $\bar{t} = t_1, \ldots, t_n$, for some $n > 0$, then let $\texttt{obs}(R\bar{t}) = \texttt{obs}(\bar{t})$
2. For conjunction, let $\texttt{obs}(\phi \wedge \psi) \equiv \texttt{obs}(\phi)|\texttt{obs}(\psi)$
3. There can be no observation of negative facts, such as, for example, that it is not the case that the color of this object is red. Rather, an observation of the color of the object is made and it is then decided by a logical query whether the statement "it is not the case that the color of this object is red" is supported by observation or not. Thus we set $\texttt{obs}(\neg\phi) \equiv \texttt{obs}(\phi)$
4. Observing existential statements is unproblematic if the type of the bound variable is a row type $\rho = [\ell_1 : \tau_1, \ldots, \ell_n : \tau_n]$. This is because the observation takes place in a finitely generated workspace U and thus there can only be finitely many objects of type ρ in U, so that observation of the existential statement may be safely reduced to such of a disjunctive statement. Thus, to observe $(\exists x : [\texttt{color} : str](x.\texttt{color} = red)$ one observes colored objects in the workspace in order to ascertain whether one with red color exists.

   By contrast, consider a statement of the form $(\exists x : \texttt{int})(x + P.\texttt{xpos} \leq 12379)$. The only pertinent observation in trying to ascertain the truth or falsity of this statement is obs(P,xpos). Everything else is a matter of logical deduction. Thus it suffices to $\texttt{obs}(P.\texttt{xpos} + x \leq 12379)$, which reduces precisely to $\texttt{obs}(P, \texttt{xpos})$.

   With these clarifications, if $A = \langle \Delta, \delta, \kappa \rangle$ is a U-agent, then let

$$\texttt{obs}((\exists x : \tau)\phi) = \begin{cases} \texttt{obs}(\phi), & \text{If } \tau = \kappa \text{ is a ground type;} \\ \texttt{obs}(\bigvee_{(M:\rho)\in U} \phi[M/x]), & \text{If } \tau = \rho \text{ is a row type.} \end{cases}$$

5. Finally, if $\phi \equiv \langle\alpha\rangle\theta$ then proceed by induction on θ

   - If $\theta = R\bar{t}$ is atomic, then the reduction obeys the rule

$$\frac{(U, A\{\alpha\}) \longrightarrow (V, B) \qquad (V, B\{\texttt{obs}(R\bar{t})\}) \longrightarrow (V, B\{\texttt{revise}(\bar{\vartheta})\})}{(U, A\{\texttt{obs}(\langle\alpha\rangle R\bar{t})\}) \longrightarrow (U, A\{\texttt{revise}(\langle\alpha\rangle\bar{\vartheta})\})}$$

   where if $\bar{\theta} = \theta_1, \ldots, \theta_n$, then $\langle\alpha\rangle\bar{\theta} = \langle\alpha\rangle\theta_1, \ldots, \langle\alpha\rangle\theta_n$.

- Consistent with our view that there can be no observation of negatives facts, if $\theta = \neg\phi$ we set $\texttt{obs}(\langle\alpha\rangle\neg\phi) = \texttt{obs}(\langle\alpha\rangle\phi)$ and so this case is reduced to the other cases. For further justification of our current definition note that $\langle\alpha\rangle\neg\phi$ is semantically equivalent to $\neg[\alpha]\phi$, so that observing $\langle\alpha\rangle\neg\phi$ is the same as observing $[\alpha]\phi$. By the observations we made in Remark 2.14, if α is a performable action on the workspace U, then U cannot distinguish between $[\alpha]\phi$ and $\langle\alpha\rangle\phi$.
- If $\theta = \theta_1 \wedge \theta_2$, so that $\langle\alpha\rangle(\theta_1 \wedge \theta_2) \equiv \langle\alpha\rangle\theta_1 \wedge \langle\alpha\rangle\theta_2$, because transition relations are functions, we may set $\texttt{obs}(\langle\alpha\rangle(\theta_1 \wedge \theta_2)) \equiv \texttt{obs}(\langle\alpha\rangle\theta_1 \wedge \langle\alpha\rangle\theta_2)$ and so this case is reduced to the other cases
- If $\theta = (\exists x : \kappa)\chi$, where κ is a ground type, then, consistent with our view that there can be no observation involving abstract entities (numbers) we may set $\texttt{obs}(\langle\alpha\rangle\theta) \equiv \texttt{obs}(\langle\alpha\rangle\chi)$ and so this case is also reduced to the other cases
- If $\theta = (\exists x : \rho)\chi$, where ρ is a row type, then we let $\texttt{obs}(\langle\alpha\rangle(\exists x : \rho)\chi) \equiv \texttt{obs}(\langle\alpha\rangle \bigvee_{(M:\rho)\in U\{\alpha\}} \chi[M/x])$ and so this case reduces to the other cases as well.

6. If $\phi \equiv \langle\alpha\rangle\theta$, where $x : \rho$ occurs free in α, then set $\texttt{obs}(\langle\alpha\rangle\theta) = \texttt{obs}(\bigvee_{(M:\rho)\in U}\langle\alpha[M/x]\rangle\theta$.

3.3.2 Testing

Testing whether a property ψ holds in the current workspace, we propose, is answered by a logical query on a belief base first revised by ground observations triggered by ψ. Thus we simply define

$$\texttt{test}(\psi)?\ p\ \texttt{else}\ q \equiv \texttt{obs}(\psi); (\psi?\ p\ \texttt{else}\ q)$$

For example, and where $P = [xpos = 5, ypos = 11]$ is a point object, to test whether $P.xpos \leq P.ypos - 1$, the observation $\texttt{obs}((P, xpos), (P, ypos))$ is made, the belief base is appropriately informed (and revised if necessary), and the truth or falsity of the tested statement is decided by a logical query on the revised belief base.

3.4 Some Example Definable Agent Behaviors

Using the basic agent behaviors so far presented we can define more complex ones, as in the following examples.

3.4.1 Satisfying Goals

We define a compound behavior `satG`, for goal satisfaction, where goals are tested and if they currently fail to hold, a plan for modifying the agent's workspace is elaborated and executed, the goal list is updated by simply dropping the satisfied goal and a recursive call is made to `satG`. We may assume that ground routines for testing whether a list is null or not, or for extracting the head of a list are available and we will not care to explicitly present them. For example, we could have incorporated a basic behavior `null`? p `else` q, for doing p if the goal list is empty and doing q otherwise. Then the behavior `satG` can be defined by the following equation, where $A = \langle\Delta, \delta, K\rangle$ and δ is either empty or of the form $\psi; \delta_1$.

$$A\{\texttt{satG}\} = A\{\texttt{null}?\ \texttt{skip}\ \texttt{else}\ (\texttt{test}(\psi)?\ (\texttt{update}(\delta_1); \texttt{satG})\ \texttt{else}\ \psi; \texttt{update}(\delta_1); \texttt{satG}))\}$$

However, in the sequel, we will not care to make such ground routines explicit and we will give relevant definitions in the following style, for $A = \langle \Delta, \delta, K \rangle$:

$$A\{\texttt{satG}\} = \begin{cases} A, & \text{if } \delta = \langle\,\rangle \\ \texttt{test}(\psi)?\ (\texttt{update}(\delta_1); \texttt{satG})\ \texttt{else}\ (\psi; \texttt{update}(\delta_1); \texttt{satG}), & \text{if } \delta = \psi; \delta_1 \end{cases}$$

3.4.2 Recurrent Behavior

As a standard application of recursive definitions we can define recurrent behaviors by setting, for each behavior p, $!p = p|!p$, or, making the agent parameter explicit, $A\{!p\} = A\{p|!p\}$.

3.4.3 Continuous Monitoring of the Environment

With the replication operator of the previous example we can easily define a U-agent $A = \langle \Delta, \delta, \kappa \rangle$ that continuously monitors its environment (observes the values of attributes of the objects in its environment), while also executing, concurrently, some program p, as

$$A\{!\texttt{obs}(M)_{(M:\rho)\in U} \mid p\}$$

3.4.4 Persistent (Vital) Goals

There are vital needs an agent must make sure to satisfy at all times, while also executing other actions. Because the environment may change as a result of the agent's own behavior and in a way that the vital goals become temporarily unsatisfied, an agent with persistent goals $\bar{\vartheta}$ will have to recurrently modify the environment so as to re-establish satisfiability of its persistent (vital) goals $\bar{\vartheta}$.

There is the option of prescribing that the desires δ mentioned in an agent's state are precisely its vital goals. Or we can simply define a new behavior $\texttt{pers}(\bar{\vartheta})$, as we do below.

$$A\{\texttt{pers}(\vartheta; \bar{\vartheta}_1)\} = A\{\texttt{test}(\vartheta)?\ \texttt{pers}(\bar{\vartheta}_1; \vartheta)\ \texttt{else}\ \vartheta; \texttt{pers}(\bar{\vartheta}_1; \vartheta)\}$$

so that satisfying persistent goals while also behaving like p can be defined by $A\{\texttt{pers}(\bar{\vartheta}) \mid p\}$.

3.5 Constrained Agent Systems

In what we have said up to now, actions transforming the workspace can be unconditionally executed. This, however, is rarely (if ever) the case. By a *global constraint* (for any action α transforming a workspace) we mean a precondition-postcondition pair (φ, ψ) . A constrained action will typically have the form

$$\texttt{test}(\varphi \wedge \langle\alpha\rangle\psi)?\alpha\ \texttt{else skip}$$

so that the agent itself will have to decide whether the precondition and post-condition of the action hold in the current workspace. Global constraints (such as that the capacity of a receptor type of object should not be exceeded, or that some numerical attribute of some type of object should only be changed within preset bounds, etc) apply to all workspaces. Local constraints can be expressed as conditional global constraints $\varphi \rightarrow (\vartheta \wedge \langle\alpha\rangle\psi)$. Global

constraints can be dressed-up as local ones: true $\longrightarrow (\varphi \wedge \langle\alpha\rangle\psi)$. So that constraints have the general form $\chi \longrightarrow (\varphi \wedge \langle\alpha\rangle\psi)$.

In a constrained system, the rule for a transition of $(U, A\{\alpha\})$, where A is a basic agent, takes the form

$$\frac{\begin{array}{l}\kappa|U = K \xrightarrow{\alpha} K' \\ U \xrightarrow{\alpha} V \\ (U, \Delta\delta\kappa\{\texttt{test}(\chi \rightarrow (\varphi \wedge \langle\alpha\rangle\psi))?a![true] \texttt{ else } a![false]\}) \longrightarrow (U, \Delta'\delta\kappa\{a![true]\})\end{array}}{(U, \Delta\delta\kappa\{\alpha\}) \xrightarrow{\alpha} (V, \Delta\delta\kappa)}$$

4 Behavioral Equivalence of Agents

We define, in this Section, several notions of agent equivalence, and explore the relations amongst them.

Definition 4.1 (Deductive Equivalence) *Let $A = \Delta\delta\kappa$ and $B = \Delta'\delta'\kappa'$ be U-agents. Define $A \prec_U B$ iff (1) $\kappa|U \subseteq \kappa'|U$, (2) $\Delta' \vdash_\Theta \Delta$ and (3) $\forall j \exists i\ \delta_i \vdash_\Theta \delta'_j$. We say that A, B are U-equivalent, denoted by $A \approx_U B$, iff $A \prec_U B$ and $B \prec_U A$.*

Lemma 4.2 *Let $A_1 = \Delta_1\delta_1\kappa_1, A_2 = \Delta_2\delta_2\kappa_2$ and assume $A_1 \approx_U A_2$. If p is a behavior such that $(U, A_1\{p\}) \Longrightarrow (U, A'_1)$, where $A'_1 = \Delta'_1\delta'_1\kappa'_1$, then $(U, A_2\{p\}) \Longrightarrow (U, A'_2)$, where $A'_2 = \Delta'_2\delta'_2\kappa'_2$ and $A'_1 \approx_U A'_2$.*

Proof: We proceed by induction on the syntax of behaviors. If p is an atomic behavior and given the hypothesis of the lemma, then p is one of the following (the cases α and $c![\vartheta]$ lead to observable actions, which is why they are ommitted.)

$$\texttt{skip} \mid \texttt{obs}(M_i)_{i\in n} \mid \texttt{revise}(\bar{\theta}) \mid \texttt{update}(\delta)$$

The case `skip` is trivial. Observations $\texttt{obs}(M_i)_{i\in n}$ lead to revision by the same list of formulae $\bar{\theta}$. Revision of deductively equivalent belief bases by deductively equivalent new facts leads to deductively equivalent revisions, and so the claim of the lemma holds. The case for $\texttt{update}(\delta)$ is obvious, by the operational semantics.

If the behavior p is compound, then (1) in the case of a recursively defined behavior p, the first step is a silent unfolding and so this case reduces to non-recursive behaviors. Similarly, if the behavior is a commitment, then the first step is a silent reduction to a new behavior and so this case reduces to the other cases. If (2) p is a conditional, then the hypothesis implies (in both cases of the conditional $\vartheta?p \texttt{ else } q$ and $\vartheta??p \texttt{ else } q$) that the same choice p or q is made by both agents, and the claim holds by induction. The cases of (3) summation, (4) concurrent behavior and (4) sequencing are obvious, by induction. ∎

Most behaviors of a situated multi-agent system are silent (internal). The only observable behaviors in the operational semantics are (1) actions α affecting the common workspace and (2) message-send actions of the form $c![\vartheta]$, where c is an assertion or request communication channel. If β is an observable behavior, $\beta := \alpha \mid a![\vartheta] \mid r![\vartheta]$, then we write $(U, \mathbf{A}) \stackrel{\beta}{\Longrightarrow} (V, \mathbf{B})$ if the behavior β is preceded and followed by some (possibly none) silent moves. We write $(U, \mathbf{A}) \Downarrow \beta$, if there is $(V, \mathbf{B})$ such that $(U, \mathbf{A}) \stackrel{\beta}{\Longrightarrow} (V, \mathbf{B})$. If

$\sigma = \beta_1 \cdots \beta_n \cdots$ is a sequence of observable actions, then we say that $(U, \mathbf{A})$ *converges* on σ, written $(U, \mathbf{A}) \Downarrow \sigma$, just in case there is a computation sequence

$$(U, \mathbf{A}) \stackrel{\beta_1}{\Longrightarrow} (U_1, \mathbf{A}_1) \stackrel{\beta_2}{\Longrightarrow} \cdots \stackrel{\beta_n}{\Longrightarrow} (U_n, \mathbf{A}_n) \stackrel{\beta_{n+1}}{\Longrightarrow} \cdots$$

Note that for a finite sequence σ, $(U, A) \Downarrow \sigma$, if and only if, for some $m \geq 0$, we have $(U, A) \stackrel{\sigma}{\Longrightarrow} (U', A' \| c_1![\vartheta_1] \| \cdots \| c_m![\vartheta_m])$.

Corollary 4.3 *Let $A_1 = \Delta_1\delta_1\kappa_1, A_2 = \Delta_2\delta_2\kappa_2$ be basic agents and assume $A_1 \approx_U A_2$. If $(U, A_1\{p\}) \stackrel{\sigma}{\Longrightarrow} (U', A' \| c_1![\vartheta_1] \| \cdots \| c_m![\vartheta_m])$, for some $m \geq 0$, and where $A' = \Delta_1'\delta_1'\kappa_1'$, then $(U, A_2\{p\}) \stackrel{\sigma}{\Longrightarrow} (U', B' \| c_1![\vartheta_1] \| \cdots \| c_m![\vartheta_m])$, for some $B' = \Delta_2'\delta_2'\kappa_2'$ such that $A' \approx_{U'} B'$.*

PROOF: Immediate from Lemma 4.2, from the fact that observable actions do not affect the state of the agent and from the fact that if $\kappa_1|U = \kappa_2|U$, then for any state transformation $\alpha \in \kappa_1|U = \kappa_2|U$, if $U \stackrel{\alpha}{\longrightarrow} V$, then $\kappa_1|V = \kappa_2|V$. This is because the transformation $\alpha = M[\ell \Leftarrow L]$ of the workspace does not affect the types of objects in the workspace (the effect of the transformation is that an object of some type is being replaced by another object *of the same type*). ∎

Next we define behavior-equivalence $\sim_U$ of agents and we show that it coincides with $\approx_U$ on basic agents. Behavioral equivalence depends on observable behaviors.

Definition 4.4 (Behavioral Equivalence) *Let A, B be any U-agents. Define $A \preceq_U B$ if and only if for any behavior p and any observable action β, if $(U, A\{p\}) \Downarrow \beta$, then $(U, B\{p\}) \Downarrow \beta$. Define $A \sim_U B$ iff $A \preceq_U B$ and $B \preceq_U A$.*

Theorem 4.5 *If $A = \langle \Delta, \delta, \kappa \rangle$ and $B = \langle \Delta', \delta', \kappa' \rangle$ are two basic U-agents, then $A \sim_U B$ iff $A \approx_U B$.*

PROOF: Assume first that $A \sim_U B$. To see that $\Delta' \equiv_\Theta \Delta$ let $p = \Delta?a![\mathtt{true}]$. Then $(U, A\{p\}) \Downarrow a![\mathtt{true}]$ and therefore, by $A \preceq_U B$, $(U, B\{p\}) \Downarrow a![\mathtt{true}]$. By the rules of the operational semantics the latter is only possible if $\Delta' \vdash_\Theta \Delta$. By similar argument it also follows that $\Delta \vdash_\Theta \Delta'$.

For pairwise equivalence of goals, let $p = \psi_i??a![\mathtt{true}]$. Then clearly $(U, A\{p\}) \Downarrow a![\mathtt{true}]$ and so $(U, B\{p\}) \Downarrow a![\mathtt{true}]$. It follows, by the operational semantics, that $\psi_i \vdash_\Theta \psi_j'$, for some j. By similar argument, there is k such that $\psi_j' \vdash_\Theta \psi_k$ and, thereby, $\psi_i \vdash_\Theta \psi_k$. By our assumption that desires are deductively independent, it follows that $\psi_i \equiv_\Theta \psi_j'$.

Finally, to see that $\kappa|U = K$ and $\kappa'|U = K'$ are identical, let $\tau \sqsubseteq [\ell : \tau']$ and $M : \tau$, $L : \tau' \in K$. Let $\alpha = M[\ell \Leftarrow L]$. Then $(U, A\{\alpha\}) \Downarrow \alpha$ and so, by $A \preceq_U B$, it must be that $(U, B\{\alpha\}) \Downarrow \alpha$. This means that $\alpha = M[\ell \Leftarrow L]$ is an action defined on the capabilities space K' of B, i.e. $M, L \in K'$ and so $K \subseteq K'$. The other direction of the inclusion is proven similarly. It follows from this that $\kappa = \kappa'$.

Conversely, suppose $A \approx_U B$. We proceed by structural induction on the syntax of behaviors. If p is an atomic behavior

$$p := \mathtt{skip} \mid \alpha \mid \theta \mid \mathtt{obs}(M_i)_{i \in n} \mid a![\theta] \mid r![\theta] \mid \mathtt{revise}(\bar{\theta}) \mid \mathtt{update}(\delta') \mid c?(\psi)p$$

then either (1) p does not lead to any observable behavior (cases $\texttt{skip}$, $\texttt{obs}(M_i)_{i\in n}$, $\texttt{revise}(\bar{\theta})$, $\texttt{update}(\delta')$, $c?(\psi)p$) and the claim is vacuously true, or (2) p is a commitment θ, leading to execution of a plan p', or else (3) p is either an output behavior, or a workspace transformation.

If $p = \theta$ is a commitment, then given deductive equivalence of the belief bases it is clear that $\Delta \vdash_\Theta \langle\alpha\rangle\theta$ iff $\Delta' \vdash_\Theta \langle\alpha\rangle\theta$. The rest follows by observing the rules (int1, int2) for commitments.

Remains to examine the cases of output and workspace transforming behaviors. In the latter case, since the agents have the same capabilities spaces, it follows that $(U, A\{\alpha\}) \Downarrow \alpha$ iff $(U, B\{\alpha\}) \Downarrow \alpha$.

In the case of output, consider first assertions $a![\vartheta]$. By the operational semantics, agents are only willing to assert ϑ provided it follows from what they believe. By the hypothesis that $A \approx_U B$, it then follows that A is willing to assert ϑ iff B is willing to assert it. In other words, $(U, A\{a![\vartheta]\}) \Downarrow a![\vartheta]$ iff $(U, B\{a![\vartheta]\}) \Downarrow a![\vartheta]$.

If the output behavior is a request, then it follows again from the relevant rule of the operational semantics and from the hypothesis that $A \approx_U B$ that A will launch a request ϑ if and only if B can do the same, since agents communicate requests only when satisfying them may further satisfaction of their own desires. In other words, $(U, A\{r![\vartheta]\}) \Downarrow r![\vartheta]$ iff $(U, B\{r![\vartheta]\}) \Downarrow r![\vartheta]$. Hence the claim is true for atomic behaviors.

We now proceed to compound (including recursively defined) behaviors, assuming as our induction hypothesis that the claim holds for their constituent behaviors. The case of recursive behaviors is resolved easily since the agents first reduce by unfolding the definition of the behavior, and so this case resolves into one of the cases of non-recursive behaviors. The cases of the conditionals $\vartheta?p$ $\texttt{else}$ q and $\vartheta??p$ $\texttt{else}$ q are immediate, since both agents will select p, or both will select q, given the rules of the operational semantics and the hypothesis that $A \approx_U B$. The rest is by induction. The cases of behavior choice and parallel is also resolved by induction, given the rules of the operational semantics. There only remains the case where $p = p_1; p_2$ is a sequence. If $(U, A\{p_1; p_2\}) \Downarrow \beta$ because $(U, A\{p_1\}) \Downarrow \beta$, then the claim is true by induction. Otherwise, $(U, A\{p_1\}) \Longrightarrow (U, A')$, in which case the claim follows by Lemma 4.2 and the induction hypothesis for p_2.

We may then conclude that $A \sim_U B$ iff $A \approx_U B$. ∎

A stronger notion than behavioral equivalence is *trace equivalence*, which we define below, focusing on sequences of observable actions, rather than single observable actions. However, it can be shown, as we do in Theorem 4.7, that the two notions coincide on basic agents.

Definition 4.6 (Trace Equivalence) *Let A, B be any U-agents. Define $A \sqsubseteq_U B$ iff for any behavior p and sequence σ of observable actions, if $(U, A\{p\}) \Downarrow \sigma$, then $(U, B\{p\}) \Downarrow \sigma'$, for some $\sigma' = \sigma_0^\frown\sigma$. We say that A, B are* trace equivalent*, denoted by $A \cong_U B$, iff $A \sqsubseteq_U B$ and $B \sqsubseteq_U A$.*

Our next proposition shows that trace and behavioral equivalence coincide on basic agents.

Theorem 4.7 *If $A = \Delta\delta\kappa$, $B = \Delta'\delta'\kappa'$, then $A \sim_U B$ iff $A \stackrel{\sim}{=}_U B$. This equivalence fails for agents A, B that are not ground. In fact, trace is finer than behavioral equivalence (if $A \stackrel{\sim}{=}_U B$, then $A \sim_U B$, but not conversely).*

PROOF: The direction $A \stackrel{\sim}{=}_U B \Longrightarrow A \sim_U B$ is immediate, by taking sequences σ consisting of a single observable action. For the converse, we may use Theorem 4.5, so that it suffices to show that $A \approx_U B \Longrightarrow A \stackrel{\sim}{=}_U B$. The arguments for atomic behaviors and for any compound behavior other than a sequence $p = p_1; p_2$ are essentially the same as in the proof of $A \approx_U B \Longrightarrow A \sim_U B$, of Theorem 4.5. We examine the case of a sequenced behavior $p = p_1; p_2$ separately.

Suppose $(U, A\{p_1; p_2\} \Downarrow \sigma$ and let σ_1, σ_2 be subsequences so that $\sigma = \sigma_1^\frown \sigma_2$ (where any of σ_1, σ_2 may be the empty sequence ϵ) and $(U, A\{p_1\}) \Downarrow \sigma_1$.

If $\sigma_1 = \epsilon$, then the result follows by Lemma 4.2 and by the inductive hypothesis applied to the sub-behavior p_2. If $\sigma_2 = \epsilon$, then the result follows by the inductive hypothesis applied to the sub-behavior p_1. Hence we may assume that $\sigma_1 \neq \epsilon \neq \sigma_2$. This implies that σ_1 is finite (since $\sigma = \sigma_1^\frown \sigma_2$ and $\sigma_2 \neq \epsilon$). Let

$$(U, A\{p_1\}) \stackrel{\sigma_1}{\Longrightarrow} (V, A' \| c_1![\vartheta_1]\| \cdots \| c_k![\vartheta_k])$$

The hypothesis $A \approx_U B$ implies that A, B can execute exactly the same workspace transformations α and, since they execute the same behavior p_1 it follows, by the inductive hypothesis on the sub-behavior p_1, that

$$(U, B\{p_1\}) \stackrel{\sigma_1}{\Longrightarrow} (V, B' \| c_1![\vartheta_1]\| \cdots \| c_k![\vartheta_k])$$

By Corollary 4.3, it follows that $A' \approx_V B'$. Now apply the inductive hypothesis for the sub-behavior p_2 to the basic agents A', B'.

Now if A, B are not ground agents, then this equivalence of the two behavioral relations fails. For example, let

$$A = \Delta\delta\kappa\{(\alpha; \alpha')|(\alpha_1; \alpha_1')\} \quad \text{and} \quad B = \Delta\delta\kappa\{(\alpha; \alpha'); (\alpha_1; \alpha_1') + (\alpha_1; \alpha_1'); (\alpha; \alpha')\}$$

Then it can be easily seen that $A \sim_U B$. However, $A \stackrel{\sim}{\neq}_U B$. This is because B has only two possible traces, $\alpha; \alpha'; \alpha_1; \alpha_1'$ and $\alpha_1; \alpha_1'; \alpha; \alpha'$, whereas any interleaving of the α actions (such as, for example, $\alpha; \alpha_1; \alpha_1'; \alpha'$) is a possible trace for A.

It is easily seen that trace equivalence is finer than behavioral equivalence, by restricting sequences to single actions. ∎

Next, we examine bisimulations and bisimilarity and compare with the notions of agent equivalence we have presented so far.

Definition 4.8 (Bisimulation) *A relation R on U-agents is a* simulation*, provided that $R(A, B)$ implies that*

- *For any behavior p and any observable action β, if*

$$(U, A\{p\}) \stackrel{\beta}{\Longrightarrow} (V, A' \| c_1![\vartheta_1]\| \cdots \| c_k![\vartheta_k])$$

for some $k \geq 0$, then there exists an agent B' such that

$$(U, B\{p\}) \stackrel{\beta}{\Longrightarrow} (V, B' \| c_1![\vartheta_1]\| \cdots \| c_k![\vartheta_k])$$

and $R(A', B')$.

The relation R is a bisimulation *if both R and R^{-1} are simulations. Bisimilarity $A \stackrel{=}{\sim}_U B$ is defined as the largest bisimulation relation on U-agents.*

Theorem 4.9 *Bisimilarity and deductive equivalence coincide on basic agents. In other words, if $A = \Delta\delta\kappa$, $B = \Delta'\delta'\kappa'$, then $A \approx_U B$ iff $A \stackrel{=}{\sim}_U B$.*

PROOF: From the definition of deductive equivalence in Definition 4.1 and by Corollary 4.3 it follows that $\approx_U$ is a bisimulation on basic agents and thus $A \approx_U B$ implies $A \stackrel{=}{\sim}_U B$. The proof of the converse is the same as that of $A \sim_U B \Longrightarrow A \approx_U B$ of Theorem 4.5. ∎

Corollary 4.10 *Let A, B be any agents. Then*

1. *If A, B are bisimilar, then they are trace equivalent*
2. *If A, B are trace equivalent, then they are behaviorally equivalent*
3. *If A, B are ground agents, then the three notions of equivalence coincide.*

PROOF: Parts 2 and 3 have been already proven in the previous theorems. For the first part, if A, B are bisimilar and $\sigma = \beta_1 \cdots \beta_n \cdots$ is a trace for A, and A' is the descendant of A after execution of the action β_1, then B can do β_1 with a descendant B' that is bisimilar to A'. By induction, any finite initial segment of σ is a trace for B, and therefore σ is a trace for B, which establishes trace equivalence of A and B. ∎

Thus we have $\stackrel{=}{\sim}_U \subseteq \stackrel{\sim}{=}_U \subseteq \sim_U$, i.e.

$$A \stackrel{=}{\sim}_U B \implies A \stackrel{\sim}{=}_U B \implies A \sim_U B$$

In the sequel, we focus only on behavioral equivalence and preorder. Our next definition generalizes behavioral equivalence to agent systems. As a notational convention we write $\mathbf{A}\{p\}$ for $A_1\{p\}\| \cdots \|A_n\{p\}$, when $\mathbf{A} = A_1\| \cdots \|A_n$.

Definition 4.11 (Behavioral Equivalence of Agent Systems) *Let $\mathbf{A}, \mathbf{B}$ be any U-agent systems. Define $\mathbf{A} \preceq_U \mathbf{B}$ if and only if for any behavior p and any observable action β, if $(U, \mathbf{A}\{p\}) \Downarrow \beta$, then $(U, \mathbf{B}\{p\}) \Downarrow \beta$. Define $\mathbf{A} \sim_U \mathbf{B}$ iff $\mathbf{A} \preceq_U \mathbf{B}$ and $\mathbf{B} \preceq_U \mathbf{A}$.*

As we comment in the sequel, our definition validates some natural identities, like

$$\texttt{nil}\|\mathbf{A} \sim_U \mathbf{A} \qquad \mathbf{A}\|\mathbf{B} \sim_U \mathbf{B}\|\mathbf{A} \qquad (\mathbf{A}\|\mathbf{B})\|\mathbf{C} \sim_U \mathbf{A}\|(\mathbf{B}\|\mathbf{C})$$

Note that the notion of agent (system) equivalence we have explored will identify agent terms if no behavior can distinguish between them. In particular, it will identify any message with the `nil` agent, since we get, in general, $\texttt{nil} \sim_U \texttt{nil}\{p\}$, for any behavior p. This is perhaps unproblematic, since messages are really not agents. However, a stronger notion is needed if we wish to distinguish messages $\texttt{nil}\{c![\vartheta]\}$ as well.

Definition 4.12 (Refined Behavioral Equivalence of Agents) *Let A, B be any agents and define $A \sqsubseteq_{\lesssim U} B$ if and only if*

$$\forall p \forall C \forall \beta \ \ (\ (U, A\{p\} \| C) \Downarrow \beta \ \ \textit{implies} \ \ (U, B\{p\} \| C) \Downarrow \beta\ )$$

Let then $A \eqsim_U B$ iff $A \sqsubseteq_{\lesssim U} B$ and $B \sqsubseteq_{\lesssim U} A$. Similarly for agent systems.

Theorem 4.13 *1. If A, B are any agents, then $A \eqsim_U B$ implies $A \sim_U B$, but not conversely.*

2. If A, B are not messages, then $A \eqsim_U B$ iff $A \sim_U B$.

3. If A, B are messages $A = \mathtt{nil}\{c![\vartheta]\}$ and $B = \mathtt{nil}\{\bar{c}![\psi]\}$, then $A \eqsim_U B$ iff (a) $c = \bar{c}$ and $\vartheta \equiv_\Theta \psi$.

PROOF: For the first part, assume the hypothesis and take the agent C of the definition to be $\mathtt{nil}$ to see that $A \sim_U B$ follows. The converse fails because $\mathtt{nil}\{c_1![\vartheta_1]\} \sim_U \mathtt{nil}\{c_2![\vartheta_2]\}$. The relation $\sqsubseteq_{\lesssim U}$ distinguishes between the two, by choosing C of an appropriate form $D\{c_1?(\psi)p\}$.

Now assume that A, B are not messages. Given the first part, proven above, we only need to show that if $A \sim_U B$, then $A \eqsim_U B$. Assuming the hypothesis, let p be any behavior and C any agent. We need to show that for any observable action β, $(U, A\{p\} \| C) \Downarrow \beta$ iff $(U, B\{p\} \| C) \Downarrow \beta$. If the justification of the action β is either that $(U, C) \Downarrow \beta$ or that $(U, A\{p\}) \Downarrow \beta$, then by $A \sim_U B$ the claim follows. There only remains the case of communication, where $C = c![\vartheta]$ and $p = c?(\psi)q$. Then both agents evolve to $A\{q[\vartheta/m]\}$ and $B\{q[\vartheta/m]\}$, respectively, and the conclusion follows by $A \sim_U B$. These are the only possible cases and therefore the two notions of agent equivalence coincide for agents that are not messages.

For messages, it is clear that if their communication channels are distinct then they are discriminated by an agent performing input on one of the channels. So suppose that $A = \mathtt{nil}\{c![\vartheta]\}$ and $B = \mathtt{nil}\{c![\psi]\}$, where we treat both cases $c = a, r$ at the same time. If the agent C in the definition of $\sqsubseteq_{\lesssim U}$ can perform no input on c, then it cannot distinguish between the two, so we may assume $C = D\{c?(\psi)p\}$. The only behaviors p that may depend on the input are

$$p = \mathtt{revise}(\psi) \mid \mathtt{update}(\psi) \mid \psi \mid d![\psi] \mid \psi?p_1 \ \mathtt{else}\ p_2 \mid \psi??p_1 \ \mathtt{else}\ p_2$$

If $\vartheta \equiv_\Theta \psi$, then none of these behaviors can distinguish between the two messages and $A \eqsim_U B$. Conversely, if $\vartheta \not\equiv_\Theta \psi$, then either belief revision, $\mathtt{revise}(\vartheta)$, $\mathtt{revise}(\psi)$, or update, $\mathtt{update}(\vartheta)$, $\mathtt{update}(\psi)$, will distinguish between the two messages. Thus messages as agents are equivalent just in case they share the same communication channel and their message contents are deductively equivalent. ∎

Corollary 4.14 (Context Lemma) *$A \sqsubseteq_{\lesssim U} B$ iff for any behavior p, any observable action β and any agent C of the form $C = \Delta_1\delta_1\kappa_1\{c?(\psi)q\}$, if $(U, A\{p\} \| C) \Downarrow \beta$, then $(U, B\{p\} \| C) \Downarrow \beta$.*

PROOF: By the proof of the previous theorem, the only agents C relevant in detecting differences in the behavior of A, B are precisely the agents C mentioned in the statement of this Corollary. ∎

Table 6: An Equational Base Theory for Record Terms

$$M =_\tau M \qquad \frac{M =_\tau M'}{M' =_\tau M} \qquad \frac{M =_\tau M' \quad M' =_\tau M''}{M =_\tau M''}$$

$$[\ldots, \ell = A : \tau_1, \ldots][\ell \Leftarrow B : \tau_1] =_{[\ldots,\ell:\tau_1,\ldots]} [\ldots, \ell = B : \tau_1, \ldots]$$

$$\frac{M_1 =_{[\ell:\tau]} N_1 \qquad N_1[\ell \Leftarrow B : \tau] =_{[\ell:\tau]} N}{M_1[\ell \Leftarrow B : \tau] =_{[\ell:\tau]} N} \qquad \frac{M_1 =_{[\ell:\tau]} N_1 \qquad N_1.\ell =_\tau N}{M_1.\ell =_\tau N}$$

5 An (In)Equational Theory for Reasoning about Agents

Based on our operational semantics we propose axioms and rules for an inequational theory, for reasoning about agents and agent systems. The axioms and rules we propose make reference to record term identities and to derivability results in the language of properties of workspaces. Identities of record terms are established in a theory containing the axioms and rules in Table 6.

Axioms and rules for an inequational theory of agent systems are presented in Table 7.

In the sequel, we show that the theory is sound with respect to refined behavioral preorder and equivalence of agents .

Theorem 5.1 *The inequational axioms and rules of Table 7 are valid with respect to behavioral preorder and equivalence, in the sense of Definition 4.12. In other words, if $A \leq_U B$, then $A \sqsubseteq_U B$ and if $A =_U B$, then $A \approx_U B$.*

Proof: The relation $\sqsubseteq_U$ is a preorder and $\approx_U$ is defined by $A \approx_U B$ iff $A \sqsubseteq_U B$ and $B \sqsubseteq_U A$, from which it follows that the partial order axiom and rules ($\leq$1, $\leq$2, $\leq$3) are sound.

Soundness for (def) is immediate. For (seq), to show that $A\{p;q\} \approx_U (A\{p\})\{q\}$ let r be any behavior and assume that $(U, (A\{p;q\})\{r\}) \Downarrow \beta$. If this follows from $(U, A\{p\}) \Downarrow \beta$, by the (seq) rules of the operational semantics, then using the rules (D1)-(D3) it follows that $(U, ((A\{p\})\{q\})\{r\}) \Downarrow \beta$. The converse is similar, going from the (D)-rules to the (seq)-rules. Otherwise, $(U, A\{p\}) \Longrightarrow (U, \Delta\delta\kappa)$, so that by the (seq) rules we have both

$$(U, (A\{p;q\})\{r\}) \Longrightarrow (U, (\Delta\delta\kappa\{q\})\{r\}) \quad \text{and} \quad (U, ((A\{p\})\{q\})\{r\}) \Longrightarrow (U, (\Delta\delta\kappa\{q\})\{r\})$$

and thus one agent converges on β iff the other does.

The case for (skip) is straightforward.

For (p1), suppose $A \sqsubseteq_U B$ and let q be any behavior. By (seq), $(A\{p\})\{q\} = A\{p;q\}$, and similarly for B. Letting $p_1 = p;q$, the conclusion $A\{p\} \sqsubseteq_U B\{p\}$ follows now directly from the hypothesis.

For (p2), assume that for any behavior p it holds that $A\{p\} \sqsubseteq_U B\{p\}$. Then in particular we obtain $A \approx_U A\{\texttt{skip}\} \sqsubseteq_U B\{\texttt{skip}\} \approx_U B$ and so we obtain $A \sqsubseteq_U B$.

The cases for $(\Delta\delta\kappa)$, (upd), (rev), (obs0), (obs), (φ?1), (φ?2), φ??1), (φ??2) and (def) are immediate and we leave them to the interested reader.

Soundness of the axioms (+L), (+R) for behavior choice is straightforward and we thus only deal with the rule (+). Assume then that $A\{p\} \sqsubseteq_U B$ and $A\{q\} \sqsubseteq_U B$, let r be any

Table 7: Axioms and Rules

(≤ 1)	$A \leq_U A$	(≤ 2)	$\dfrac{A \leq_U B \quad B \leq_U A}{A =_U B}$
(≤ 3)	$\dfrac{A \leq_U B \quad B \leq_U C}{A \leq_U C}$	(def)	$\dfrac{p =^{def} q}{A\{r\} =_U A\{r[q/p]\}}$
(seq)	$A\{p;q\} =_U (A\{p\})\{q\}$	(skip)	$A\{\mathtt{skip}\} =_U A$
(p1)	$\dfrac{A \leq_U B}{A\{p\} \leq_U B\{p\}}$	(p2)	$\dfrac{A\{p\} \leq_U B\{p\}}{A \leq_U B}$
$(\Delta\delta\kappa)$	$\dfrac{\Delta' \vdash_\Theta \Delta, \quad \forall j \exists i\ \delta_i \vdash_\Theta \delta'_j \quad \kappa\|U \preceq \kappa'\|U}{\Delta\delta\kappa \leq_U \Delta'\delta'\kappa'}$		
(upd)	$\Delta\delta\kappa\{\mathtt{update}(\delta')\} =_U \Delta\delta'\kappa$	(rev)	$\dfrac{\mathtt{revise}(\Delta,\bar{\theta}) = \Delta'}{\Delta\delta\kappa\{\mathtt{revise}(\bar{\theta})\} =_U \Delta'\delta\kappa}$
(obs0)	$A\{\mathtt{obs}(M_i)_{i\in 0}\} =_U A$		
(obs)	$\dfrac{M_i \in U \qquad \vdash M_i =_{\tau_i} N_i \quad (i \in n > 0)}{A\{\mathtt{obs}(M_i)_{i\in n>0}\} =_U A\{\mathtt{revise}(M_i =_{\tau_i} N_i)_{i\in n>0}\}}$		
$(\varphi?1)$	$\dfrac{\Delta \vdash_\Theta \varphi}{\Delta\delta\kappa\{\varphi?p\ \mathtt{else}\ q\} =_U \Delta\delta\kappa\{p\}}$	$(\varphi?2)$	$\dfrac{\Delta \nvdash_\Theta \varphi}{\Delta\delta\kappa\{\varphi?p\ \mathtt{else}\ q\} =_U \Delta\delta\kappa\{q\}}$
$(\varphi??1)$	$\dfrac{\exists i\ \varphi \vdash_\Theta \delta_i}{\Delta\delta\kappa\{\varphi??p\ \mathtt{else}\ q\} =_U \Delta\delta\kappa\{p\}}$	$(\varphi??2)$	$\dfrac{\forall i\ \varphi \nvdash_\Theta \delta_i}{\Delta\delta\kappa\{\varphi??p\ \mathtt{else}\ q\} =_U \Delta\delta\kappa\{q\}}$
(def)	$A\{p\} =_U A\{q(p)\}$		If p is defined by $p = q(p)$
(+L)	$A\{p\} \leq_U A\{p+q\}$	(+R)	$A\{q\} \leq_U A\{p+q\}$
(+)	$\dfrac{A\{p\} \leq_U B \quad A\{q\} \leq_U B}{A\{p+q\} \leq_U B}$	(pskip)	$A\{\mathtt{skip}\|p\} =_U A\{p\}$
(pcom)	$A\{p\|q\} =_U A\{q\|p\}$	(pass)	$A\{(p\|q)\|r\} =_U A\{p\|(q\|r)\}$
(pseq)	$A\{p;(q\|r)\} \leq_U A\{(p;q)\|r\}$	(p+)	$A\{(p+q)\|r\} =_U A\{p\|r + q\|r\}$
$(p\varphi?1)$	$\dfrac{\Delta \vdash_\Theta \varphi}{\Delta\delta\kappa\{(\varphi?p\ \mathtt{else}\ q)\|r\} =_U \Delta\delta\kappa\{p\|r\}}$		
$(p\varphi?2)$	$\dfrac{\Delta \nvdash_\Theta \varphi}{\Delta\delta\kappa\{(\varphi?p\ \mathtt{else}\ q)\|r\} =_U \Delta\delta\kappa\{q\|r\}}$		
$(p\varphi??1)$	$\dfrac{\exists i\ \varphi \vdash_\Theta \delta_i}{\Delta\delta\kappa\{(\varphi??p\ \mathtt{else}\ q)\|r\} =_U \Delta\delta\kappa\{p\|r\}}$		
$(p\varphi??2)$	$\dfrac{\forall i\ \varphi \nvdash_\Theta \delta_i}{\Delta\delta\kappa\{(\varphi??p\ \mathtt{else}\ q)\|r\} =_U \Delta\delta\kappa\{q\|r\}}$		
(inter)	$\dfrac{p, q \text{ atomic}}{A\{(p;p')\|(q;q')\} =_U A\{p;(p'\|q;q') + q;(p;p'\|q')\}}$		

behavior and suppose, further, that $(U, (A\{p+q\})\{r\}) \Downarrow \beta$. By the operational semantics and by the (seq) axiom, this is either by a computation of the form

$$(U, (A\{p+q\})\{r\}) \longrightarrow (U, A\{p; r\}) \Downarrow \beta$$

or by a computation of the form

$$(U, (A\{p+q\})\{r\}) \longrightarrow (U, A\{q; r\}) \Downarrow \beta$$

If either $(U, A\{p\}) \Downarrow \beta$ or $(U, A\{q\}) \Downarrow \beta$, then the hypothesis implies that $(U, B\{r\}) \Downarrow \beta$. Otherwise, if $(U, A\{p; r\}) \Downarrow \beta$ while $(U, A\{p\}) \not\Downarrow \beta$, then $(U, A\{p\}) \Longrightarrow (U, \Delta\delta\kappa)$, so that the computation has the form

$$(U, (A\{p+q\})\{r\}) \longrightarrow (U, A\{p; r\}) \Longrightarrow (U, \Delta\delta\kappa\{r\}) \Downarrow \beta$$

and since we assume $A\{p\} \sqsubseteq_{\approx U} B$ it follows that $(U, B\{r\}) \Downarrow \beta$. Similarly if the choice is resolved in favor of q. This shows that $A\{p+q\} \sqsubseteq_{\approx U} B$.

Soundness for each of ((pskip), (pcom), (pass), (p+), (pφ?1), (pφ?2), (pφ??1), (pφ??2) and (pseq) is immediate.

Finally, soundness of the interleaving law (int), for atomic p, q, is immediate given the rules of the operational semantics ∎

Remark 5.2 If $p_1, \ldots, p_n, q_1, \ldots, q_m$ are atomic actions, then

$$A\{(p_1; \cdots; p_n)|(q_1; \cdots; q_m)\} =_U A\{ \sum_{j=1}^{j=n+m} r_{j_1}; \cdots; r_{j_{n+m}}\}$$

where

- For each $1 \leq j_s \leq n+m$, either $r_{j_s} = p_i$, for some $1 \leq i \leq n$, or $r_{j_s} = q_k$, for some $1 \leq k \leq m$
- If $r_{j_s} = p_i$, then for each $i < i' \leq n$ there exists an index $j_s < j_{s'} \leq n+m$ such that $r_{j_{s'}} = p_{i'}$, and
- If $r_{j_s} = q_k$, then for each $k < k' \leq m$ there exists an index $j_s < j_{s'} \leq n+m$ such that $r_{j_{s'}} = q_{k'}$

For example

$$\begin{aligned} A\{(\alpha_1; \alpha_2)|(\beta_1; \beta_2)\} =_U A\{&(\alpha_1; \alpha_2; \beta_1; \beta_2) + \\ &(\alpha_1; \beta_1; \beta_2; \alpha_2) + \\ &(\alpha_1; \beta_1; \alpha_2; \beta_2) + \\ &(\beta_1; \beta_2; \alpha_1\alpha_2) + \\ &(\beta_1; \alpha_1; \alpha_2; \beta_2) + \\ &(\beta_1; \alpha_1; \beta_2; \alpha_2)\}. \end{aligned}$$

6 A Simple Case Study

In this Section we present a simple two-agent system, used as an example in [4], and which we describe in the agent language we have developed in this report.

The system consists of a two-room building, a waiting room and an examination room. Only one person can be in the examination room at each particular time. Several persons are initially waiting in the waiting room. Our two agents (with shared workspace the building) are a physician and a secretary. The physician observes the number of persons in his examination room, examines the person currently in, and then sends her out. If the examination room becomes empty, the physician notifies the secretary, trying to get all the patients in for examination. The secretary observes the waiting room and his goal is to send all waiting persons, one by one, in the examination room. If he ever comes to believe that the examination room is empty and there are more patients waiting, he notifies the physician that it is possible to send one patient in for examination, and does indeed send one person out of the waiting room. Having received the secretary's notification, the physician admits the patient into the examination room and begins examination. The system will terminate when all patients from the waiting room have gone into and then out of the examination room.

First, we describe the objects and workspaces involved. There are only three objects, the building, whose two attributes are the two rooms, each of which has only one attribute, the number of people currently in. The building is the two agents' shared workspace. But they have distinct capabilities spaces as the secretary can only modify the number of people in the waiting room, but it is the physician's role to admit a person in the examination room.

Thus we have some initial situation

$$\begin{aligned} BLDG &:= [w = WR, e = ER] \\ WR &:= [inW = 32] \\ ER &:= [inE = 0] \end{aligned}$$

Our two $BLDG$-agents are, first, the physician A, who comes into the examination room with no observational beliefs (he only has the trivial belief $true$), whose goals are to keep the examination room full, as long as there are people in the waiting room

$$\psi \;=\; WR.inW = 0 \;\vee\; ER.inE = 1$$

and whose capabilities space is the examination room ER, determined by the capabilities $\kappa_E = [inE : \texttt{int}]$, so that the initial state of the physician A is

$$A := \langle true, \psi, \kappa_E \rangle$$

And second, the secretary B, also with trivial initial beliefs $true$, whose capabilities space is the waiting room WR, determined by the capabilities $\kappa_W = [inW : \texttt{int}]$, and whose goal is to empty the waiting room,

$$\psi' := WR.inW = 0$$

by sending the patients into the examination room, one by one. The constraint for not just sending everybody away (!) is

$$\vartheta_\alpha := (WR.inW = x) \wedge (x > 0) \wedge (ER.inE = 0) \wedge \langle\alpha\rangle(WR.inW = x - 1)$$

so that any modification $\alpha = WR[inW \Leftarrow k]$ is really a constrained action $\vartheta_\alpha?\alpha$. Thus the initial state of the secretary B is

$$B := \langle true, \psi', \kappa_W \rangle$$

The system is controlled by recursive behaviors p_A and p_B for the two agents

$$\begin{array}{lll} p_B = & a_1?(\psi)\texttt{revise}(\psi); & \texttt{test}(WR.inW > 0)? \\ & & (ER.inE = 0)? \\ & & \quad WR[inW \Leftarrow WR.inW - 1]; \\ & & \quad a_2![\langle WR[inW \Leftarrow WR.inW - 1]\rangle(WR.inW \geq 0)]; \\ & & \quad p_B \\ & & \texttt{else}\ p_B \\ & & \texttt{else}\ a_2![WR.inW = 0] \end{array}$$

$$\begin{array}{ll} p_A = & \texttt{test}(ER.inE = 1)? \\ & \quad \texttt{skip}^{10}; \\ & \quad ER[inE \Leftarrow ER.inE - 1] \\ & \quad \texttt{revise}(ER.inE = 0); \\ & \quad a_1![ER.inE = 0]; \\ & \quad q \\ & \texttt{else}\ a_1![ER.inE = 0]; \\ & \quad q \end{array}$$

where q is the behavior

$$\begin{array}{lll} q = & a_2?(\psi)\texttt{revise}(\psi); & \\ & \langle WR[inW \Leftarrow WR.inW - 1]\rangle(WR.inW \geq 0)? & ER[inE \Leftarrow ER.inE + 1];\ p_A \\ & & \texttt{else skip} \end{array}$$

The system is then $(BLDG, A\{p_A\} \| B\{p_B\})$. To see how the system works, observe that initially it has the form

$$(BLDG, A\{\texttt{test}(ER.inE = 1)?p_1\ \texttt{else}\ p_2\} \| B\{a_1?(\psi)\texttt{revise}(\psi); p_3\}) \tag{1}$$

Given the initial state of the workspace

$$BLDG = [w = WR, e = ER] \qquad WR = [inW = 32] \qquad ER = [inE = 0]$$

and the operational semantics for the (definable) behavior for testing, an observation $\texttt{obs}(ER.inE)$ is made, the belief base (initially trivial) of agent A is revised to $\Delta_A = (ER.inE = 0)$, the test fails and the system evolves to the state

$$(BLDG, A_1\{a_1![ER.inE = 0]; q\} \| B\{a_1?(\psi)\texttt{revise}(\psi); p_3\}) \tag{2}$$

where

$$A_1 = \langle \Delta_A, \psi, \kappa_E \rangle$$

A message is then sent

$$(BLDG, A_1\{a_2?(\psi)\texttt{revise}(\psi); q_1\} \| \texttt{nil}\{a_1![ER.inE = 0]\} \| B\{a_1?(\psi)\texttt{revise}(\psi); p_3\}) \quad (3)$$

The message is consumed by the secretary (agent B), its belief base is appropriately revised to $\Delta_B^1 = (ER.inE = 0)$, a test to find out if there are any patients in WR is performed by B, it succeeds, the belief base of B is revised by asserting the fact $(WR.inW = 32)$, transforming the state of B to

$$B_1 = \langle (ER.inE = 0) \wedge (WR.inW = 32), \psi', \kappa_W \rangle$$

whence the query $(ER.inE = 0)$ succeeds and the system evolves to

$$(BLDG, A_1\{a_2?(\psi)\texttt{revise}(\psi); q_1\} \| B_1\{WR[inW \Leftarrow WR.inW - 1]; a_2![\theta]; p_3\}) \quad (4)$$

Agent B_1 then sends a patient for admission into the examination room and notifies the physician (the message θ is $\langle WR[inW \Leftarrow WR.inW - 1]\rangle(WR.inW \geq 0)$). The message is consumed by agent A (the physician), leading to revising A's belief base so that the query

$$\langle WR[inW \Leftarrow WR.inW - 1]\rangle(WR.inW \geq 0)$$

succeeds, following which the physician admits the person in the examination room. The capabilities spaces and the workspace have now changed to

$$BLDG_1 = [w = WR_1, e = ER_1] \qquad WR_1 = [inW = 31] \qquad ER_1 = [inE = 1]$$

The physician then observes that there is a patient in the examination room, examines her (indicated by 10 successive `skip` actions), sends the person out, thus transforming the workspace to the state $BLDG_2 = [w = WR_1, e = ER]$ and the system gets to the state

$$(BLDG_2, A_2\{a_1![ER.inE = 0]; q\} \| B_2\{a_1?(\psi)\texttt{revise}(\psi); p_3\}) \quad (5)$$

following which the secretary becomes informed that a new patient can be moved from the waiting room to the examination room etc.

The process will continue until all patients are moved out of the building, having been examined.

7 Conclusions and Further Research

This report has focused on proposing a process algebraic approach to agent systems, rather than investigating partial aspects such as planning, belief revision, goal updating or communication languages and agent coordination. Particular solutions to partial issues such as those mentioned above can be incorporated in the framework explored here, by suitable additions or extensions and this task constitutes the substance of further research. We have provided an account of agent systems by incorporating a calculus of the environments

agents live in, which has allowed us to provide a simple account of perception (performing observations triggered by a formula) and action (modification of object properties in the agent environment). We have kept the language of properties of workspaces (language of beliefs and desires) simple, in the context of this report, but it would certainly be worth extending the framework we have proposed here by investigating more complex logical languages, such as fixpoint logics or logics allowing for the representation of time and temporal properties, as well as for higher-order beliefs (beliefs about beliefs of self or of other agents). We leave issues such as these to further research.

Acknowledgements: This research has been made possible by partial financial support from the Research Office of the Technological Education Institute of Larissa, Greece.

References

[1] Amandi, A., Iturregui, R., Zunino, A., "Object-Agent Oriented Programming", Technical Report, 1997?

[2] Barwise, J., Gabbay, D., and Hartonas, C., "Information Flow and the Lambek Calculus", in Logic, Language and Computation, J. Selligman and D. Westerstahl (eds), Proceedings: Information-Oriented Approaches to Logic, Language and Computation, Moraga, California, 1994. CSLI Lecture Notes, No. 58, Stanford (1996), pp 49-64.

[3] Barwise, J., Gabbay, D., and Hartonas, C., "On the Logic of Information Flow", Bulletin of the IGPL , vol 3, No 1, 1995, pp 7-49

[4] van Eijk, R., de Boer, F., van der Hoek, W. and Meyer, J.-J. Ch., "Information Passing and Belief Revision in Multi-Agent Systems", pp. 29-47, in Müller, J.P., Singh, M. P., Rao, A.S. (eds), *Intelligent Agents V*, LNAI 1555, Springer, 1999.

[5] van Eijk, R.M., de Boer, F.S., van der Hoek, W., Meyer, J.-J. Ch., "Generalized Object-Oriented Concepts for Inter-Agent Communication", in Castelfranchi, C., Lespérance, Y., (eds) *Intelligent Agents VII*, LNAI, Springer, 2001.

[6] van Eijk, R.M., de Boer, F.S., van der Hoek, W., Meyer, J.-J. Ch., "A Language for Modular Information-Passing Agents", in K.R.Apt (ed), CWI Quarterly, Special Issue on Constraint Programming, vol 11, pp 273-297, CWI, Amsterdam, 1998.

[7] Guessoum, Z., Briot, J.-P., "From Active Objects to Autonomous Agents", in IEEE Concurrency, vol 7-3, pp. 68-76, 1999.

[8] Hartonas, C., and Hennessy, M., "Full Abstractness for a Functional-Concurrent Language with Higher-Order Value-Passing", *Information and Computation* vol 145, pp. 64-106, 1998.

[9] Hartonas, C., "Duality for Modal μ-Logics", *Theoretical Computer Science*, vol 202 (1-2), 1998, pp 193-222.

[10] van der Hoek, W., "Logical Foundations of Agent-Based Computing", Technical Report, Institute of Informatics and Computing Sciences, Utrecht University, 2001?

[11] Kinny, D., "The Ψ Calculus: An Algebraic Agent Language", in *Intelligent Agents VIII*, LNAI 2333, pp. 32-50, Springer-Verlag, 2002.

[12] Meyer, J.-J. Ch. and Schobbens, P.-Y. (eds), *Formal Models of Agents*, LNAI 1760, Springer, 1999.

[13] Sadri, F., Toni, F., "Computational Logic and Multi-Agent Systems: A Roadmap", Technical Report, Department of Computing, Imperial College, London, 1999.

[14] Shoham, Y., "Agent-Oriented Programming", *Artificial Intelligence*, vol 60-1, pp 51-92, 1993.

[15] Shoham, Y., "AGENT0: A Simple Agent Language and its Interpreter", in *Proceedings of the 9th National Conference on Artificial Intelligence*, AAAI-91, pp. 26-41, Eindhoven, The Netherlands, 1996, Springer LNAI 1038.

[16] Wooldridge, M., Lomuscio, A., "A Computationally Grounded Logic of Visibility, Perception and Knowledge", in *Logic Journal of the IGPL*, vol 9-2, pp. 257-272, 2001.

[17] Wooldridge, M.J., and Jennings, N.R, "Agent Theories, Architectures and Languages: A Survey", in M. Wooldridge and N. R. Jennings, editors: *Intelligent Agents - Theories, Architectures, and Languages I.* Springer-Verlag Lecture Notes in AI Volume 890, February 1995.