A Dynamical Plan Revising for Ambient Systems

Ahmed Chawki Chaouche, Amal El Fallah-Seghrouchni, Jean-Michel Ilié, Djamel Eddine Saidouni

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 32 (2014) 37 – 44

Procedia Computer Science

The 5th International Conference on Ambient Systems, Networks and Technologies

A Dynamical Plan Revising for Ambient Systems

A-C. Chaouche[a,b,*], A. El Fallah Seghrouchni[a], J-M. Ilié[a], D-E. Saïdouni[b]

[a]*LIP6 Laboratory, University Paris VI, France*
[b]*MISC Laboratory, University Constantine II, Algeria*

Abstract

The proposed AgLOTOS formal specification language is dedicated to express BDI agent plans, according to the features and requirements of Ambient Intelligence (AmI). It offers a rich modular approach to express and compose elementary plans in order to execute them concurrently. We show how a plan is built automatically as a system of concurrent processes from the mental attitudes of the agent. In contrast to existing approaches, the plan is viewed as the realization of a whole set of partially ordered intentions. The AgLOTOS semantics accords with the possibility of updating some sub-plans on the fly, as the intention set of the BDI agent is revised.

Selection and Peer-review under responsibility of the Program Chairs.
Keywords: Ambient Intelligence, BDI Agent, Planning Language, Plan Revising, Plan Consistency.

1. Introduction

Ambient Intelligence (AmI) systems are highly dynamical systems where agents can enter or leave the system, and each one can evolve according to its mental attitudes. For the design of such complex systems MAS approaches offer interesting frameworks, since their agents are considered as intelligent, proactive and autonomous[1,2,3,4]. Many modeling approaches are proposed, which partially focus on some of the MAS aspects[5,6,7,8,9]. In fact, the major problem consists in recognizing its environmental contexts, including its locality and the discovery of other agents. In[10], it is shown how autonomous BDI agents[11] can evolve and move within an ambient environment, based on an agent centric approach and a context-awarness. Basically, the model is enriched by AmI primitives like communication and mobility, used over an open system.

This paper aims at proposing a rich and efficient planning process within each AmI agent, such that the specification of a plan is directly and automatically built from a set of intentions considered by the BDI agent. An interesting work in this domain was already proposed according to a standard MAS context[7]. Based on an algebraic recursive specification language of actions, it is demonstrated how to formally produced a goal/plan hierarchy (HTN), from a library of elementary plans, each one considered as a possible realization for some goal.

* Corresponding author. Tel.: +33-144-278-794 ; fax: +33-144-277-000.
E-mail address: ahmed.chaouche@lip6.fr

Selection and Peer-review under responsibility of the Program Chairs.
doi:10.1016/j.procs.2014.05.395

Intention revising [12,13,14] is considered as an important notion in BDI agent conceptual frameworks (see Bratman et al. [1]), actually, agents are dynamic entities which are changing intentions and then plan, throughout their evolutions. The authors of [15], demonstrate a solution which consists in an on-the-fly revising of plans to be executed, depending on the modifications of the agents goals, for whatever reasons and in particular the fact that some plans of intentions can be conflictual. The proposed solution comes from the fact that actions are processed atomically (without concurrency) and plans are strongly scheduled according to a hierarchical alternation of goals and primitive actions. Moreover, the proposed structure simplifies the complex mechanism of a BDI agent: goals are viewed as events and the language of plans allows to directly handle believes and to trigger events.

In contrast, the HoA model presented in [10] offers an explicit separation between the management of plans and the BDI attitudes. The planning process is viewed as a service which directly relates from the set of intentions of the agent, whatever the BDI process is. Our aim is investigating an adapted revising solution enhancing the concurrency of actions, letting the planning process selecting and trying one or more plans from some intention concurrently, and even dealing with several intentions in the same time. The syntax and semantics of the specification algebraic language *AgLOTOS* first presented in [10], is considered again and augmented in order to formally capture the revising of plans. Since our starting point is a set of intentions, we assume that the BDI agent itself can solve conflictual situations that could arise for some context, between intentions, by means of a scheduling process applied on the set of intentions.

The remaining of the paper is organized as follows: Section 2 presents the proposed HoA model able to answer to AmI requirements in MAS. In Section 3, the *AgLOTOS* language is defined to compose and schedule the different sub-plans attached to intentions. According to this principle, the agent is considered of having only one (global) plan. Section 4 details the semantics of *AgLOTOS* which provides a rich mechanism to represent the plan evolutions of the agent while taking into account the change of the context. In Section 5, we define an updating service based on *AgLOTOS* which allows changing the plan according to the revising of the intention set. Throughout the paper, a simple AmI scenario is given as an illustration of our approach. The last section concludes and outlines our perspectives.

2. The Higher-Order Agent Model

We are interested in modeling the evolution of the agent globally. Figure 1 highlights the agents BDI structure we consider in this paper. The reasoning mechanism, made by the so-called BDI Process module, is triggered by the perceived events. It manages/updates the beliefs (B), desires (D), and intentions (I) structures. Then, it calls the Planning Process module in order to produce consistent action plans automatically, helped by a library of plans. Our agent-based model captures two main aspects of the agent: *(1)* the mental reasoning of the agent as a *BDI state* and *(2)* the evolution of the selected plan as a *Planning state*.

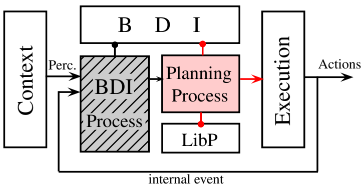

Fig. 1. Agent BDI structure

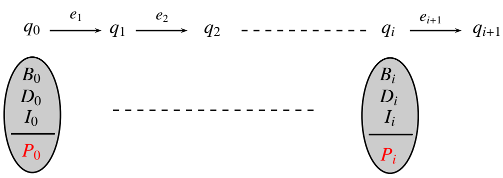

Fig. 2. The agent behavioral changes

Each agent configuration is then composed of a *BDI state* and a *Planning state*, knowing that the operational semantics of plans can yield all the possible evolutions implied by the selected plan. As illustrated in Figure 2, the occurrences of events may cause some changes of configurations. These evolutions are formally represented by the following *Higher-order Agent* model, named *HoA* for short. It is defined over an alphabet of events triggered by the actions being executed and by perception events, namely $Evt = E_{Act} \cup E_{Perc}$. Among the actions, message sendings are available, and message receivings are viewed as specific environmental perceptions. Moreover, mobility is handled as a specific action (*move*).

Definition 2.1. *(Formalization of the HoA model)*

Considering any agent of the AmI system, let BDI be the set of all the possible mental states that can be defined over the BDI structure and let $\mathcal{P}$ be the set of all the possible corresponding plans. Let $\mathcal{CN}$ be the set of all the *possible planning states evolving in some plan and LibP be a subset of $\mathcal{P}$ representing the library of plans. The HoA model of the agent is a transition system Ω, represented by a tuple $\langle Q, q_0, \rightarrow, F_M, F_P, F_C \rangle$, where:*

- *Q is the set of HoA configurations such that any configuration q is a tuple $q = (bdi, C)$ where bdi and C respectively represent the BDI and the* planning *states of the agent in q,*
- *$q_0 \subseteq Q$ is the initial HoA configuration ($q_0 = (bdi_0, C_0)$),*
- *$\rightarrow \subseteq Q \times Evt \times Q$ is the set of transitions between HoA configurations,*
- *$F_M : Q \longrightarrow BDI$ associates a BDI state with each HoA configuration,*
- *$F_P : BDI \times LibP \longrightarrow P$ associates with each BDI state, an agent plan built from the library of plans,*
- *$F_C : Q \longrightarrow C\mathcal{N}$ associates a planning state with each HoA configuration.*

In this paper, a BDI state is composed of three sets of propositions, representing the Beliefs, Desires and Intentions of the agent. The Plan is directly derived from the intentions and is written as an *AgLOTOS* expression. In fact, the semantics of *AgLOTOS* yields all the possible planning states implied by the plan expression (see Section 3).

3. AgLOTOS-Based Planning Construction

Starting from any BDI state and LibP library, we show how to build the associated plan automatically. We assume that the LibP library is indexed by the intention set such that each intention is associated to one or many plans, called *elementary plans*. A plan is often an alternate of several plans, such that each one can satisfy the corresponding intention. Moreover, the plan of an intention, namely *intention plan*, is a composition of elementary plans. Since in our view, the intention set can express a scheduling of intentions, a second level of composition is required to express the whole plan of an intention set, namely the *Agent plan*.

The compositions of sub-plans are specified by using our algebraic description language, namely *AgLOTOS*. *AgLOTOS* strictly extends the Basic *LOTOS* Language[16] since elementary plans are expressed in terms of Basic *LOTOS* expressions. Specific *AmI primitives* allow one to take into account the mobility and the asynchronous communication. Also, location and reception information are assumed to be handled at the BDI level. The next subsections introduce the *AgLOTOS* syntax with the specification of plans.

3.1. Syntax of AgLOTOS Plans

The AgLOTOS Algebraic Language.

We now define the syntax of elementary plans which are written using the algebraic language *AgLOTOS*[10]. This language extends the LOTOS language[16] in order to deal with the concurrency of actions in plans.

Let O be the (finite) set of observable actions which are viewed as instantiated predicates, ranged over $a, b, ...$ and let L be any subset of O. Let $\mathcal{H} \subset O$ be the subset of actions which represent the Ami primitives:

- In *AgLOTOS*, actions are refined to make the AmI primitives observable:*(1)* an agent can perceive the enter and leave of other agents in the AmI system, *(2)* it can move between the AmI system localities and *(3)* an agent can communicate with another agent in the system.
- An *AgLOTOS* expression refers to contextual information with respect to the (current) BDI state of the agent: *(1)* Θ is a finite set of space localities, *(2)* Λ is a set of agents with which it is possible to communicate, and $\mathcal{M}$ is the set of possible messages to be sent and received.
- The agent mobility is expressed by the primitive $move(\ell)$ which is used to handle the move of the agent to some locality ℓ ($\ell \in \Theta$). The syntax of the communication primitives is inspired from the semantics of the π-calculus primitives, however considered within a totally dynamic communication support, hence without specification of predefined channels: the expression $x!(\nu)$ specifies the emission to the agent x ($x \in \Lambda$) of some message ν ($\nu \in \mathcal{M}$), whereas, the expression $x?(\nu)$ means that ν is received from some agent x.

Let $Act = O \cup \{\tau, \delta\}$, be the set of actions, where $\tau \notin O$ is the internal action and $\delta \notin O$ is a particular observable action which features the successful termination of a plan.

The *AgLOTOS* language specifies pairs for each elementary plan consisting of a name to identify it and an *AgLOTOS* expression to feature its behavior. Consider that elementary plan's names are ranged over $P, Q, ...$ and that the set of all possible behavior expressions is denoted $\mathcal{E}$, ranged over $E, F, ...$ The *AgLOTOS* expressions are written by composing (observable) actions through LOTOS operators. The syntax of an *AgLOTOS* elementary plan P is defined inductively as follows:

$$
\begin{array}{rlll}
P & ::= E & & \textit{Elementary plan} \\
E & ::= & exit \mid stop & \\
& \mid & a; E \mid E \odot E & (a \in \mathcal{O}) \\
& \mid & hide\ L\ in\ E & \\
\mathcal{H} & ::= \mid & move(\ell) & (\mathcal{H} \subset \mathcal{O}, \ell \in \Theta) \\
& \mid & x!(v) \mid x?(v) & (x \in \Lambda, v \in \mathcal{M}) \\
\odot & = & \{\ |||, |[L]|, ||, [\,], \gg, [> \} &
\end{array}
$$

The elementary expression *stop* specifies a plan behavior without possible evolution and *exit* represents the successful termination of some elementary plan. In the syntax, the set $\odot$ represents the standard LOTOS operators: $E\ [\,]\ E$ specifies a non-deterministic choice, *hide L in E* a hiding of the actions of L that appear in E, $E \gg E$ a sequential composition and E [> E the interruption. The LOTOS parallel composition, denoted $E\,|[L]|\,E$, can model both synchronous composition, $E \,||\, E$ if $L = \mathcal{O}$, and asynchronous composition, $E \,|||\, E$ if $L = \emptyset$. In fact, the *AgLOTOS* language exhibits a rich expressivity such that the sequential executions of plans appears to be only a particular case.

Formal Specification of AgLOTOS Plans.

The building of an agent plan requires the specific *AgLOTOS* operators: *(1)* at the agent plan level, the *parallel* $|||$ and the *sequential* $\gg$ *composition* operators are used to build the agent plan, in respect with the intentions of the agent and the associated weights, *(2)* the *alternate composition* operator, denoted $\Diamond$, allows to specify an alternation of elementary plans. In particular, an intention is satisfied iff at least one of the associated elementary plans is successfully terminated.

Let $\widehat{\mathcal{P}}$ be the set of names used to identify the possible intention plans: $\widehat{P} \in \widehat{\mathcal{P}}$ and let $\overline{\mathcal{P}}$ be the set of names qualifying the possible agent plans: $\overline{P} \in \overline{\mathcal{P}}$.

$$
\begin{array}{lll}
\widehat{P} ::= P \mid \widehat{P} \Diamond \widehat{P} & & \textit{Intention plan} \\
\overline{P} ::= \widehat{P} \mid \overline{P} \,|||\, \overline{P} \mid \overline{P} \gg \overline{P} & & \textit{Agent plan}
\end{array}
$$

AgLOTOS Plan Building From Intentions.

With respect to the set of intentions I of the agent, the agent plan is formed in two steps: *(1)* by an extraction mechanism of elementary plans from the library, *(2)* by using the composition functions called *options* and *plan*:

- $libP : \mathcal{I} \to 2^{\mathcal{P}}$, features the library of elementary plans.
- $options : \mathcal{I} \to \widehat{\mathcal{P}}$, yields for any $i \in \mathcal{I}$, an intention plan of the form: $\widehat{P_i} = \Diamond_{P \in libP(i)} P$.
- $plan : 2^{I} \to \overline{\mathcal{P}}$, creates the final agent plan $\overline{P}$ from the set of intentions I. Depending on how I is ordered, the intention plans yielded by the different mappings $\widehat{P_i} = options(i)$ $(i \in I)$ are composed by using the *AgLOTOS* composition operators $|||$ and $\gg$.

To be pragmatic considering any BDI state of the agent, we propose that the agent can label the different elements of the set I of intentions by using a weight function $W : I \longrightarrow \mathbb{N}$. This allows us to weight the corresponding intention plans yielded by the mapping *options*. The ones having the same weight are composed by using the concurrent parallel operator $|||$. In contrast, the intention plans corresponding to distinct weights are ordered by using the sequential operator $\gg$. For instance, let $I = \{i_0^1, i_1^2, i_2^1, i_3^0\}$ be the considered set of intentions, such that the superscript information denotes a weight value, and let $\widehat{P_0}, \widehat{P_1}, \widehat{P_2}, \widehat{P_3}$ be their corresponding intention plans, the constructed agent plan could be viewed (at a plan name level) as: $plan(I) = \widehat{P_1} \gg (\widehat{P_0} |||\widehat{P_2}) \gg \widehat{P_3}$.

Table 1. A state evolution for Alice and Bob

Alice's scenario		Bob's scenario	
q_0^A	$B_0 = \{in(me, \ell_1), in(copies, \ell_2)\}$ $D_0 = \{meeting(Bob, \ell_1), getting\_copies(\ell_2)\}$ $I_0 = \{meeting(Bob, \ell_1)\}$ $\overline{P_0} = meet(Bob); exit$	q_0^B	$B_0 = \{in(me, \ell_2)\}$ $D_0 = \{waiting(v), meeting(Alice, \ell_1)\}$ $I_0 = \{waiting(v), meeting(Alice, \ell_1)\}$ $\overline{P_0} = Alice?(v); exit \mid\mid\mid move(\ell_1); meet(Alice); exit$
q_1^A	$B_1 = \{in(me, \ell_1), in(copies, \ell_2), in(Bob, \ell_2)\}$ $D_1 = \{meeting(Bob, \ell_1), asking(Bob, get\_copies(\ell_2))\}$ $I_1 = \{meeting(Bob, \ell_1), asking(Bob, get\_copies(\ell_2))\}$ $\overline{P_1} = meet(Bob); exit \mid\mid\mid Bob!(get\_copies(\ell_2)); exit$	q_1^B	$B_1 = \{in(me, \ell_2), in(copies, \ell_2)\}$ $D_1 = \{meeting(Alice, \ell_1), getting\_copies(\ell_2)\}$ $I_1 = \{meeting(Alice, \ell_1), getting\_copies(\ell_2)\}$ $\overline{P_1} = get\_copies(\ell_2); exit \gg move(\ell_1); meet(Alice); exit$

3.2. A Simple AmI Example

Let us briefly recall the scenario presented in[10] where Alice and Bob are two agents of an AmI Universitary system. Such a system is clearly open since agents can enter and leave. The fact that Bob is entering the system can be perceived by Alice in case she is already in. Since Alice is context-aware, she can take advantage of this information, together with other information like the fact she is able to communicate with Bob through the system.

Let $\Theta = \{\ell_1, \ell_2\}$ be two localities of the system where the agents behave. The proposed problem of Alice is that she cannot make the two following tasks in the same period of time: *(1)* to meet with Bob in ℓ_1, and *(2)* to get her exam copies from ℓ_2. Clearly, the Alice's desires are conflictual since Alice cannot be in two distinct localities simultaneously.

Table 1 represents a possible evolution of some *HoA* configurations for these agents, highlighting how to solve the Alice's problem. Alice and Bob are specified separately, and it is the proper tasks of Bob and Alice to coordinate, at their BDI process levels. In order to express the agent configurations, BDI propositions and plan actions are simply expressed by using instantiated predicates, like $get\_copies(\ell_2)$. Sub-plans are viewed as concurrent processes, terminated by the specific *exit* process, *a la LOTOS*.

Within each agent, we assume that BDI process can order the set of intentions to be considered. For instance, in the configuration q_1^B of Bob, the intention set $I_1 = \{meeting(Alice, \ell_1), getting\_copies(\ell_2)\}$ is ordered such that $weight(meeting(Alice, \ell_1)) < weight(getting\_copies(\ell_2))$. In the intention set I_1, the plan expression of Bob is: $\overline{P_1} = get\_copies(\ell_2); exit \gg move(\ell_1); meet(Alice); exit$, which is built by using the *options* and *plan* mappings. Pay attention that some actions can be processed concurrently, so is the case in the configuration q_0^B, for the sub-plans $get\_copies(\ell_2); exit$ and $move(\ell_1); meet(Alice); exit$.

The reader may notice that the initial plans of Bob and Alice accords with the formal building of plans presented in Section 4. Moreover, the revising of their plans follows the techniques presented in Section 5.

4. Semantics of AgLOTOS Plans

The *AgLOTOS* operational semantics is basically derived from the one of Basic LOTOS, which is able to capture the evolution of a concurrent processes. A configuration (E, P) represents a process identified by P, such that its behavior expression is E, moreover, the notation $P ::= E$ means that the behavior expression E is assigned to P. In case P is an elementary plan, its expression is called an elementary expression. The reader may refer to[10] for a detailed semantics of elementary plans, viewed as Basic LOTOS processes.

Further, the behavior expression of the agent plan $\overline{P}$ is denoted $[\overline{P}]$. Definition 4.1 generically specifies how $[\overline{P}]$ is formed compositionally from the intention plan configuration of the agent, like $(E, \widehat{P})$ (see *rule 2*), themselves built from an alternate of *elementary plans* configuration, like (E_k, P_k) (see *rule 1*).

Definition 4.1. *Any plan configuration $[\overline{P}]$ has a generic representation defined by the following two rules:*

$$(1) \quad \frac{\overline{P} ::= \widehat{P} \qquad \widehat{P} ::= \Diamond^{k=1..n} P_k \qquad P_k ::= E_k}{[\overline{P}] ::= (\Diamond^{k=1..n} E_k,\ \widehat{P})} \qquad\qquad (2) \quad \frac{\overline{P} ::= \overline{P_1} \odot \overline{P_2} \qquad \odot \in \{\mid\mid\mid, \gg\}}{[\overline{P}] ::= [\overline{P_1}] \odot [\overline{P_2}]}$$

Table 2 shows the operational semantic rules defining the possible planning state changes for the agent. The rules apply from any HoA configuration $q = (bdi, C)$, where the planning state C is directly specified as an agent plan expression, like $[\overline{P}]$. In each row in the table, there are three kinds of derivations: *(1)* the left column shows the nominal case considering the execution of any action a ($a \in O \cup \{\tau\}$), *(2)* and *(3)* the other two columns focus on the termination action of some intention plan, $\widehat{P}$. In the middle one, the considered intention plan is successfully terminated whereas in the right one, the failure termination case is treated. With respect to any intention plan $\widehat{P}$, $\widehat{P}$ and $\neg\widehat{P}$ respectively represent the successful and failure termination cases of $\widehat{P}$. Hence, if $\mathcal{CN}$ is the set of all the possible planning states for the agent, then the transition relation between the planning states is a subset of $\mathcal{CN} \times Act \times (\widehat{\mathcal{P}} \cup \neg\widehat{\mathcal{P}}) \times \mathcal{CN}$. The transitions $(C_1, a, \widehat{P}, C_2)$ and $(C_1, a, \neg\widehat{P}, C_2)$ ($\widehat{P} \in \widehat{\mathcal{P}}$) provoke an internal event informing the BDI process of the termination of the intention plan $\widehat{P}$. For sake of clarity, the transition (C_1, a, nil, C_2) is simply denoted $C_1 \xrightarrow{a} C_2$, representing the execution of a non termination action a.

- The two first rows concern the derivations of the behavior expression of an intention plan $\widehat{P}$, under the execution of some action. The (Action) rules exhibit the simple case where E is an elementary expression of $\widehat{P}$, whereas the (Alternate) rules focus on the execution of an alternate of elementary expressions, like $\Diamond^{k=1..n} E_k$. The middle rule captures the successful termination of $\widehat{P}$, under the execution of the action δ, while the right one captures the failure, in case the behavior expression of $\widehat{P}$ is equivalent to $fail$. In this paper, $fail$ represents the fact that the execution of some behavior expression E fails due to the dynamical environment of the agent. In the (Alternate) row, the behavior expression $E = \Diamond^{k=1..n} E_k$ of an intention plan $\widehat{P}$, is refined by using the mapping *select* which selects one of the elementary expression among the ones of E, e.g. $E_j = select(\widehat{P})$. The alternate operation is semantically defined by introducing a new semantic operator $\rhd$, in order to take this selection into account: $E_j \rhd (\Diamond E_{k \neq j}^{k=1..n} E_k)$. Observe that $E \rhd F$, yields E if E is a success and F if E fails.
- In the two last rows, the sequential and parallel LOTOS operators are re-considered to take into account the compositions of intention plan configurations, in a sequential or parallel way.

Table 2. Semantic rules of agent plan configurations

(Action)	$\dfrac{E \xrightarrow{a} E'}{(E,\widehat{P}) \xrightarrow{a} (E',\widehat{P})}$	$\dfrac{E \xrightarrow{\delta} stop}{(E,\widehat{P}) \xrightarrow[\widehat{P}]{\tau} (stop,\widehat{P})}$	$\dfrac{E \equiv fail}{(E,\widehat{P}) \xrightarrow[\neg\widehat{P}]{\tau} (stop,\widehat{P})}$
(Alternate)	$\dfrac{E_j \xrightarrow{a} E'_j \quad E_j = select(\widehat{P})}{(\Diamond^{k=1..n} E_k,\widehat{P}) \xrightarrow{a} (E'_j \rhd (\Diamond_{k \neq j}^{k=1..n} E_k),\widehat{P})}$	$\dfrac{E \xrightarrow{\delta} stop}{(E \rhd F,\widehat{P}) \xrightarrow[\widehat{P}]{\tau} (stop,\widehat{P})}$	$\dfrac{E \equiv fail \quad F \xrightarrow{a} F'}{(E \rhd F,\widehat{P}) \xrightarrow{a} (F',\widehat{P})}$
(Sequence)	$\dfrac{C_1 \xrightarrow{a} C'_1}{C_1 \gg C_2 \xrightarrow{a} C'_1 \gg C_2}$	$\dfrac{C_1 \xrightarrow[\widehat{P}]{\tau} C'_1}{C_1 \gg C_2 \xrightarrow[\widehat{P}]{\tau} C_2}$	$\dfrac{C_1 \xrightarrow[\neg\widehat{P}]{\tau} C'_1}{C_1 \gg C_2 \xrightarrow[\neg\widehat{P}]{\tau} C_2}$
(Parallel)	$\dfrac{C_1 \xrightarrow{a} C'_1}{C_1 \vert\vert\vert C_2 \xrightarrow{a} C'_1 \vert\vert\vert C_2 \quad C_2 \vert\vert\vert C_1 \xrightarrow{a} C_2 \vert\vert\vert C'_1}$	$\dfrac{C_1 \xrightarrow[\widehat{P}]{\tau} C'_1}{C_1 \vert\vert\vert C_2 \xrightarrow[\widehat{P}]{\tau} C_2 \quad C_2 \vert\vert\vert C_1 \xrightarrow[\widehat{P}]{\tau} C_2}$	$\dfrac{C_1 \xrightarrow[\neg\widehat{P}]{\tau} C'_1}{C_1 \vert\vert\vert C_2 \xrightarrow[\neg\widehat{P}]{\tau} C_2 \quad C_2 \vert\vert\vert C_1 \xrightarrow[\neg\widehat{P}]{\tau} C_2}$

4.1. Application to the scenario

Consider Table 1 where $\overline{P_1}$ is the agent plan considered for Bob in the configuration q_1^B. The evolution of this plan $[\overline{P_1}]$ is expressed by: $[\overline{P_1}] = ((E_g, \widehat{P_g}) \gg (E_m, \widehat{P_m}))$, where $(E_m, \widehat{P_m})$ and $(E_g, \widehat{P_g})$ are two intention plan configurations of Bob. The first one corresponds to the intention *meeting*(*Alice*, ℓ_1) and the second to *getting_copies*(ℓ_2), such that $E_m = move(\ell_1); meet(Alice); exit$ and $E_g = get\_copies(\ell_2); exit$.

An example of execution derived from the planning state C_0 is the following, expressing that Bob fails to get the copies but this does not prevent him to move and perform the meeting with Alice:

$$((E_g, \widehat{P_g}) \gg (E_m, \widehat{P_m})) \xrightarrow[\neg\widehat{P_g}]{\tau} (E_m, \widehat{P_m}) \xrightarrow{move(\ell_1)} (E'_m, \widehat{P_m}) \xrightarrow{meet} (E''_m, \widehat{P_m}) \xrightarrow[\widehat{P_m}]{\tau} (stop, \widehat{P_m}).$$

5. Dynamical Plan Revising

In our model, the BDI process drives the planning process such that adding or removing intentions possibly provoke the change of the agent plan. Of course, the BDI process cannot ask for such a change, if this could imply an incoherent state for the agent. In fact, the BDI process must be informed by the planning process about the terminations of intention plans in order to act with consistency. At this point, we assume that the only dependencies within the intention set are due to the weighting of intentions required by the BDI process. A rough approach would consist in waiting the termination of the whole plan, meaning that the planning process reaches the final planning state of the current plan, before taking any intention change into account. In this paper, we propose an improved method which consists in updating the agent plan as the revising of intentions are required by the BDI process. This update consists in adding new intention plans and removing some of the remaining intention plans in progress. We take profit from the compositional nature of *AgLOTOS*, that allows the planning process to manage the different intention plans distinctly. Recall that any planning state structurally specifies the different remaining *AgLOTOS* expressions to execute, corresponding to the intention plans to be achieved.

In some *HoA* configuration $q = (bdi, C)$, $I(bdi)$ represents the intention set in this configuration. Let us consider some planning state $C = [\overline{P}]$, such that $[\overline{P}] = \odot_{i\in 1..n} (E_i, \widehat{P_i})$ represents the remaining intention plan configurations to execute, where each $\widehat{P_i}$ corresponds to the plan of the intention i, E_i is its associated *AgLOTOS* expression and $\odot \in \{|||, \gg\}$. The updating of the intention set and the associated agent plan relies on the following principles:

- according to the semantics, the termination of any intention plan produces an internal event which changes the BDI state, in particular by removing the achieved intention from the intention set of the agent.
- it is easy to build some mappings which relates every intention $i \in I$ to the corresponding pair $(E, \widehat{P})$ and vice versa: *(1) remain* : $\overline{\mathcal{P}}\times I \to \mathcal{E}\times\widehat{\mathcal{P}}$ maps intention $i \in I$ to the corresponding pair $(E, \widehat{P})$ of $[\overline{P}]$; *(2) index* : $\widehat{\mathcal{P}} \to I$ maps each $\widehat{P}$ to the corresponding intention $i \in I$. It is worth noting that from $weight(index(\widehat{P_i}))$ such that $i \in I$, one yields the weight of the intention i.

The add and remove update operations are formalized by the following two mappings $add, remove$. These mappings defined from $2^{\mathcal{E}\times\widehat{\mathcal{P}}} \times \mathcal{I}$ yields a new agent plan whose expression $[\overline{P}]$ is (re)built from the given set of intention plan configurations and the intention to be added or removed. The *Adding* of a new intention k, assuming its intention plan configuration is $(E_k, \widehat{P_k})$, means:

- adding k in I and rebuild the weight mapping to take k into account, then
- building a new agent plan expression, from the set of remaining intention plan configurations $\cup_{i\in I}\, remain(\overline{P}, i)$ and their respective weights $weight(i)$.

Formally, let C be the current planning state of the agent and k be the intention to be added, the planning state C' obtained after the adding operation is defined by: $C' = (add(\cup_{i\in I}\, remain(C, i), k))$. The explicit removing of a (non-terminated) intention k from I, means that the corresponding $(E_k, \widehat{P_k}) = remain(\overline{P}, k)$ must be removed from $\overline{P}$. As for the adding function, the resulting planning state C' after removing the intention k is: $C' = (remove(\cup_{i\in I}\, remain(C, i), k))$.

5.1. Application to the scenario

Consider the example of Table 1 again, the changes of the presented initial *HoA* configurations for Alice and Bob (taken separately) are due to the respective perceptions of Alice and Bob and the fact they are anticipative. Actually, after having perceived that Bob is in ℓ_2 ($e_1 = perc(in(Bob, \ell_2))$), which is the locality of the exam copies, Alice enriches her beliefs, desires and intentions, aiming at communicating with Bob and asking for his help to bring her the copies. Consequently, she evolves to the new *HoA* configuration q_1^A, where the generated plan suggests that Alice sends the message $Bob!(get\_copies(\ell_2))$.

Notice that Bob is able to receive any message from Alice, which is denoted $Alice?(v)$ in q_0^B. The reception of the message sent by Bob triggers an event at its BDI process level. Since here, Bob accepts to bring the copies with him to

Alice, he expands his beliefs ($in(copies, \ell_2)$) and also take into account a new intention $getting\_copies(\ell_2)$, in fact consistent with the previous one. The HoA configuration of Bob is changed to q_1^B i.e. $I_1 = I_0 \cup \{getting\_copies(\ell_2)\}$, and the plan expression $[\overline{P_0}]$ of Bob is updated by using the add mapping: $[\overline{P_1}] = add(\cup_{i \in I_0} remain(\overline{P_0}, i), getting\_copies(\ell_2))$, in order to satisfy all of his desires, getting first the copies then going to meet Alice.

6. Conclusion

The proposed AgLOTOS agent-based algebra appears to be a powerful and intuitive way to express an agent plan. In contrast to existing approaches, plans are composed as concurrent processes and the sub-plans corresponding to different intentions can be executed concurrently, in a unified way.

In this context, we show how to build an agent plan automatically from the set of intentions of the agent. To solve conflict, it is possible to take into account a scheduling of the intentions. Lastly, updates are made possible from a notion of intention plans used to build the agent plan. Up to our knowledge, the presented work seems the first one to deal with the revising of plans on-the-fly, as the intentions of the agent are modified.

At the agent level, the planning state of the agent is also expressed as an AgLOTOS expression, representing the state evolution of the agent plan. This is considered as part of the configuration of the agent. The resulting model called the Higher-order agent model (HoA) formally represents a BDI-AmI open system where agents can reason, communicate and move. Agent dynamicity and context-awareness are handled due to the fact that agents can change their mental state adequately to the perceptions of new events.

Our next perspective is studying the evolution of the HoA model in order to analyze the updates of plans with respect to some applications. We also deal with higher concerns like learning and rationality aspects in relation with the successful and failure intention plan executions highlighted in this paper.

References

1. Bratman, M.E., Israel, D.J., Pollack, M.E.. Plans and resource-bounded practical reasoning. *Computational Intelligence* 1988;**4**:349–355.
2. Doyle, J.. Rationality and its roles in reasoning. *Computational Intelligence* 1992;**8**:376–40.
3. Dennis, L.A., Fisher, M., Webster, M.P., Bordini, R.H.. Model checking agent programming languages. *Automated Software Engg* 2012; **19**(1):5–63.
4. Shoham, Y.. Agent-oriented programming. *Artif Intell* 1993;**60**(1):51–92.
5. El Fallah Seghrouchni, A., Suna, A.. Claim: A computational language for autonomous, intelligent and mobile agents. In: Dastani, M., Dix, J., El Fallah Seghrouchni, A., editors. *PROMAS*; vol. 3067 of *Lecture Notes in Computer Science*. Springer; 2003, p. 90–110.
6. Bordini, R.H., Hübner, J.F., Vieira, R.. Jason and the golden fleece of agent-oriented programming. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A., editors. *Multi-agent programming : languages, platforms and applications.*; no. 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations. New York: Springer; 2005, p. 3–37.
7. Sardina, S., de Silva, L., Padgham, L.. Hierarchical planning in bdi agent programming languages: a formal approach. In: *Proceedings of AAMAS '06*. New York, NY, USA: ACM; 2006, p. 1001–1008.
8. Kouah, S., Saïdouni, D.E., Ilié, J.M.. Synchronized petri net: A formal specification model for multi agent systems. *Journal of Software* 2013;**8**(3):587–602.
9. Schild, K.. On the relationship between bdi logics and standard logics of concurrency. *Autonomous Agents and Multi-Agent Systems* 2000; **3**(3):259–283.
10. Chaouche, A.C., El Fallah Seghrouchni, A., Ilié, J.M., Saïdouni, D.E.. A higher-order agent model for ambient systems. *Procedia Computer Science* 2013;**21**(0):156 – 163. The 4th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2013).
11. Rao, A.S., Georgeff, M.P.. An abstract architecture for rational agents. In: Nebel, B., Rich, C., Swartout, W.R., editors. *KR*. Morgan Kaufmann. ISBN 1-55860-262-3; 1992, p. 439–449.
12. Icard III, T.F., Pacuit, E., Shoham, Y.. Joint revision of beliefs and intention. In: *Principles of Knowledge Representation and Reasoning*. 2010, p. 572–574.
13. Grant, J., Kraus, S., Perlis, D., Wooldridge, M.. Postulates for revising bdi structures. *Synthese* 2010;**175**(1):39–62.
14. Hoek, W., Jamroga, W., Wooldridge, M.. Towards a theory of intention revision. *Synthese* 2007;**155**(2):265–290.
15. Shapiro, S., Sardina, S., Thangarajah, J., Cavedon, L., Padgham, L.. Revising conflicting intention sets in bdi agents. In: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*; AAMAS '12. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 0-9817381-2-5, 978-0-9817381-2-3; 2012, p. 1081–1088.
16. Brinksma, E., editor. *ISO 8807, LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*. 1988.