A Higher-order Agent Model with Contextual Management for Ambient Systems

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

A Higher-Order Agent Model with Contextual Planning Management for Ambient Systems

Ahmed-Chawki Chaouche[1,2], Amal El Fallah Seghrouchni[1], Jean-Michel Ilié[1] and Djamel Eddine Saïdouni[2]

[1] LIP6 Laboratory, University of Pierre and Marie Curie
4 Place Jussieu, 75005 Paris, France
{ahmed.chaouche,amal.elfallah,jean-michel.ilie}@lip6.fr
[2] MISC Laboratory, University Constantine 2
Ali Mendjeli Campus, 25000 Constantine, Algeria
saidouni@misc-umc.org

Abstract. This paper presents a concrete software architecture dedicated to ambient intelligence (AmI) features and requirements. The proposed behavioral model, called Higher-order Agent (HoA) captures the evolution of the mental representation of the agent and the one of its plan simultaneously. Plan expressions are written and composed using a formal algebraic language, namely AgLOTOS, so that plans are built automatically and on the fly, as a system of concurrent processes. Based on a specific semantics, a guidance service is also proposed to assist the agent in its execution. Moreover due to the specific structure of AgLOTOS expressions, the update of sub-plans is realized automatically accordingly to the revising of intentions, hence maintaining the consistency of the agent.

Keywords: Ambient intelligence, BDI agent, Formal planning language and semantics, Dynamical plan revising, Planning consistency and guidance.

1 Introduction

Ambient Intelligence (AmI) is the vision of ubiquitous electronic environment that is non-intrusive and proactive, when assisting people during various activities [1, 2]. For the design of such complex systems, Multi-agent System (MAS) approaches offer interesting frameworks, since their agents are considered as intelligent, proactive and autonomous [3]. Thanks to their mental attitudes, the Belief-Desire-Intention (BDI) agents of [4] are able to use their Beliefs, Desires, and Intentions rationally, in order to select and execute a plan of actions.

The major problem for AmI agent consists in recognizing its environmental contexts, including its location and the discovery of other agents. Nevertheless, the HoA model proposed recently in [5], outlined how autonomous BDI agents can evolve and move within an ambient environment, based on a context-awareness. The major features and functionalities of AmI are taken into account,

in particular dynamic requirements such that: AmI systems can be open, thus agents can dynamically enter or leave the system.

The presented paper is inspired by the HoA model however an efficient planning management process is introduced in the architecture of the agent. We take profit from the fact that the plan of the agent can be derived from the current set of intentions of the agent. Our approach is based on a formal description language, namely *AgLOTOS*, allowing us to introduce modularity and concurrency aspects to compose sub-plans, viewed as processes. Unlike the formal description of [6], the AgLOTOS semantics overpasses the sequential execution of sub-plans. Rather, the concurrency of sub-plans is fully implemented and is only restrained with the purpose of solving possible inconsistency between intentions.

In this context, the planning process must be able to select and try one or more plans from some intention, and even deal with several intentions at the same time. Moreover, plans must be revised on the fly, in the sense that agents are dynamic entities which are changing the set of intentions and then plan, throughout their evolutions. As underlyied by several authors, this is considered an important notion in BDI agent conceptual frameworks [7, 8, 9].

The planning process we propose also aims at offering to each AmI agent, powerful predictive services, that can run on the fly. Like in other recent approaches which are dedicated to the planning and the validation of BDI MAS systems, e.g. [6, 10], we focus on one agent rather than on the whole MAS, since this eases us to embed agent in whatever environment and to deal with the openness of AmI systems.

The original contributions of this paper are the following *(1)* a well-structure extension of the AgLOTOS language and its semantics, which allows automatic revisions of plans, *(2)* a concrete software architecture allowing the management of the possible actions failures and *(3)* automatic guidance service based on the representation of plans. The paper is organized as follows : Section 2 details the considered AmI features. This allows us to introduce our Agent model in Section 3, namely *Higher-order agent model* (*HoA*), which captures the evolution of the agent in both its mental and planning states. In Section 4, the (structured) *AgLOTOS* language is defined to build plans automatically from the set of intentions, based on a library of elementary plans. then, an efficient plan revising is provided in Section 4, which works accordingly to the revision of intentions. In Section 5, the semantics of AgLOTOS is enriched to automatically produce a *Contextual Planning System* (*CPS*), in order to guide the execution of plans contextually. In Section 6, the description of our experiment project illustrates a concrete use of our approach. Section 7 presents the related works concerning the modeling and the specification of AmI agents and systems. The last section is our conclusion.

2 AmI Requirements

The AmI systems we consider are open space and dynamic systems. Their agents can reason and are assumed to be BDI agents thus have complex features as

described in Figure 1. An agent is assumed to be **autonomous** and **pervasive**, thus operate without the direct intervention of humans or other agents. It is **anticipative** and can process **rational** decisions, based on its own knowledge and beliefs.

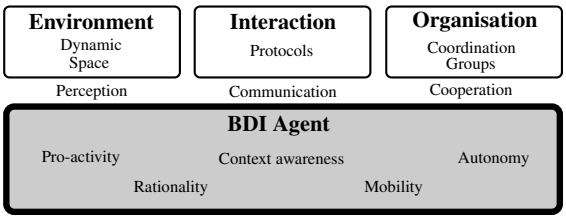

Fig. 1. Agent features for AmI systems

As a corollary of autonomy, an AmI agent is **context-aware**. We see the context of an agent as every environmental information perceived by the agent, in particular vicinity notions in a domain that considers space, time and social relationships. Thus, determining if a piece of information is relevant for an agent should be done based on its local context information.

To improve behavior and knowledge, an AmI agent can **communicate** and **cooperate** with its neighbors. Also, AmI agent can be **mobile** moving from one location to another one in a given space.

The BDI architecture is one of the major approaches to design pro-active agents in MAS [11]. Inspired from [4] and [12], the following functions describe the reasoning mechanism in the BDI agent, in order to produce a plan of actions. It is triggered by the perceived events (Evt) and often helped by a library of plans ($LibP$).

- $revs : 2^B \times Evt \rightarrow 2^B$ is the belief revision function applied when the agent receives a new event.
- $des : 2^B \times 2^D \times 2^I \rightarrow 2^D$ is the Desire update function that maintains consistency with the selected desires,
- $filter : 2^B \times 2^D \times 2^I \times LibP \rightarrow 2^I$ is the Intention function which yields the intentions the agent decides to pursue, among the possible options, taking into account new opportunities.
- $options : I \times LibP \rightarrow P$ is the function which associates a plan with each intention of the agent by using the $LibP$ library.
- $plan : 2^I \rightarrow P$ is a Plan function that processes an executable plan from some (filtered) Intention, knowing that any intention can be viewed as a partial plan.

3 The Higher-Order Agent model

We are interested in modeling the evolution of the agent. Figure 2 highlights the agent architecture we consider in this paper, called *Higher-order Agent ar-*

chitecture (*HoA*). In contrast to the traditional BDI architecture, our approach enhances a clear separation in three processes:

- The *Context Process* is in charge of the context information of the agent. It is triggered by new perceptions of the environment and also by internal events informing about the executions of actions. At a low level, it is in charge of observing the realization of the executions of actions and plans of actions, in order to state whether they are successfully achieved or if a failure occurs.
- The *Mental Process* corresponds to the reasoning part of the agent. It is notified by the context process so that it can be aware of the important context changes and can provoke possible revisions of the beliefs (B), desires (D), and intentions (I) data.
- The *Planning Process* is called by the mental process. Helped by the LibP library, it mainly produces a plan of actions from the set of intentions, but also offers some services related to the management of plans.

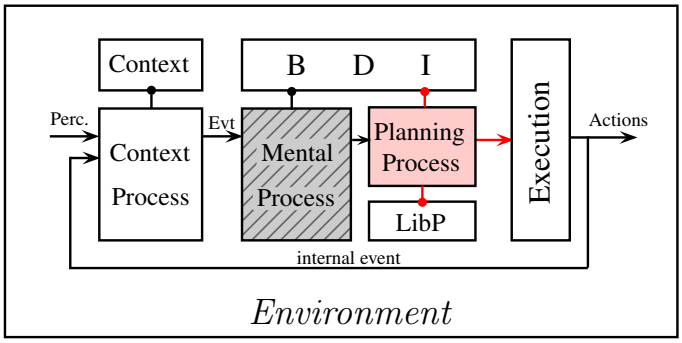

Fig. 2. Higher-order Agent architecture

The behavioral model of the agent only relates on two main aspects of the agent: *(1)* the mental reasoning of the agent and *(2)* the evolution of the selected plan. Hence, we consider that the state of the agent, namely its *HoA configuration*, is a pair composed of a *BDI state* and a *Planning state*. As illustrated by Figure 3, the occurrences of events may cause some changes of one of these parts or both.

The evolution of configurations is formally represented by the following *Higher-order Agent* (*HoA*) behavioral model. It is defined over an alphabet of events triggered by the actions being executed and by perception events, namely $Evt = Evt_{Act} \cup Evt_{Perc}$. 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*).

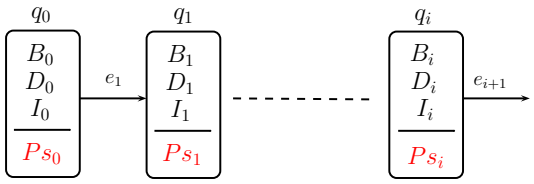

Fig. 3. The agent behavioral changes

Definition 1. *(The HoA model and HoA configuration)*
Consider any agent of the AmI system, then let $\mathcal{BDI}$ be the set of all the possible states that can be defined over the BDI structure. Moreover, let $\mathcal{P}$ be the set of all the possible corresponding plans, $\mathcal{PS}$ be the set of all the possible planning states evolving in some plan and let LibP be a subset of $\mathcal{P}$ representing the library of plans. The HoA model of the agent is a transition system, represented by a tuple $\langle Q, q_0, \rightarrow, F_M, F_P, F_{PS}\rangle$, where:

- *Q is the set of HoA configurations such that any configuration q is a tuple $q = (bdi, ps)$ where bdi and ps respectively represent the BDI and the planning states of the agent in q,*
- *$q_0 \subseteq Q$ is the initial HoA configuration, e.g. $q_0 = (bdi_0, ps_0)$,*
- *$\rightarrow \subseteq Q \times Evt \times Q$ is the set of transitions between configurations,*
- *$F_M : Q \longrightarrow \mathcal{BDI}$ associates a BDI state with each HoA configuration,*
- *$F_P : \mathcal{BDI} \times LibP \longrightarrow \mathcal{P}$ associates with each BDI state, an agent plan built from the LibP library,*
- *$F_{PS} : Q \longrightarrow \mathcal{PS}$ 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 intentions are assumed to be partially ordered by associating a weight to each intention. This allows the mental process to organize its selected intentions, in order to solve possible conflicts. The Plan is directly derived from the intentions and is written as an AgLOTOS plan expression. Actually, the possible planning states are derived by using the semantics of AgLOTOS from the plan expression (see Section 4 to get more details).

A Simple AmI Example

Let Alice and Bob be 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 locations 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 Bob in ℓ_1, and *(2)* to get her exam copies from ℓ_2. Clearly, the Alice's desires are inconsistent since Alice cannot be in two distinct locations simultaneously.

In the following sections, the *AgLOTOS* specification language will be used for formalizing this scenario.

4 Planning Formal Syntax and Semantics

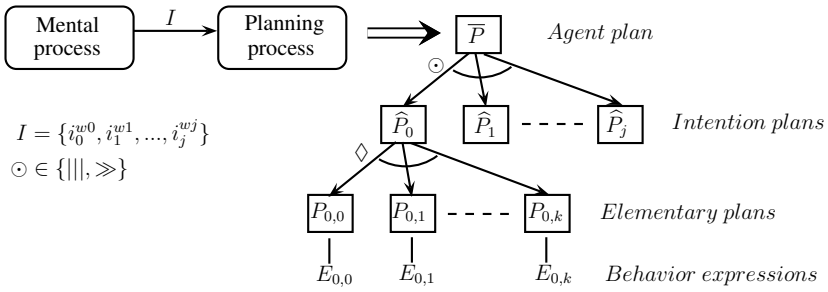

Fig. 4. Agent planning structure

Table 1. Synthetic presentation of the used notations

Notation	Description
q	HoA configuration
bdi	BDI state
ps	Planning state
E	AgLOTOS expression
P	Elementary plan
$\widehat{P}$	Intention plan
$\overline{P}$	Agent plan
(E, P)	Elementary plan configuration
$(E, \widehat{P})$	Intention plan configuration
$[\overline{P}]$	Agent plan configuration

In our approach, the planning language is well-structured as described in Figure 4 and the associated notation in Table 1. Any plan of the agent, namely the *Agent plan* accords with the two following levels: *(1)* the agent plan is made of sub-plans called *Intentions plans*. Intentions plans correspond to the agent intentions and each one is dedicated to achieve the corresponding intention; *(2)* each intention plan can be an alternate of several sub-plans, called *Elementary plans*,

each one being extracted from the LibP library. This allows one to consider different ways to achieve the corresponding intention (see Section 4.1). Further, we assume that the LibP library is indexed by the set of all the possible intentions that the agent can engage.

4.1 The Syntax of AgLOTOS Plans

Syntax of Elementary Plans. Elementary plans are written using the algebraic language *AgLOTOS* [13]. This language inherits from the LOTOS language [14] so offers different ways to express the concurrency of actions in plans. The building of an AgLOTOS expression refers to a finite set of observable actions. Further, let $\mathcal{O}$ be this set whose elements range over $a, b, ...$ and let L be any subset of $\mathcal{O}$. Let $\mathcal{H} \subset \mathcal{O}$ be the set of the so-called AmI primitives which represent the mobility and communication:

- In AgLOTOS, actions are refined to make the AmI primitives observable: *(1)* an agent can perceive the enter and leave of another agent in the AmI system, *(2)* it can move between the AmI system locations, and *(3)* it 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 locations, *(2)* Λ is a set of agents with which it is possible to communicate, and *(3)* $\mathcal{M}$ 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 agent move to some location ℓ ($\ell \in \Theta$). The syntax of the communication primitives is inspired from the semantics of π-calculus primitives, however with the consideration of 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 $Act = \mathcal{O} \cup \{\tau, \delta\}$, be now the considered set of actions, where $\tau \notin \mathcal{O}$ is the internal action and $\delta \notin \mathcal{O}$ is a particular observable action which features the successful termination of a plan.

The AgLOTOS language specifies a pair for each elementary plan composed of a name to identify it and an AgLOTOS expression to feature its behavior. Consider that elementary plan's names $\mathcal{P}$ 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 actions through LOTOS operators. The syntax of an AgLOTOS elementary plan P is defined inductively as follows:

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

The elementary expression *stop* specifies a plan behavior without possible evolution and *exit* represents the successful termination of some 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 of the left hand side part by the right one. 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.

Building of the Agent Plans from Intentions and Elementary Plans. The building of an agent plan requires the specific AgLOTOS operators:

- at the agent plan level, the parallel $|||$ and the sequential $\gg$ composition operators are used to build the agent plan, in respect to the intentions of the agent and the associated weights.
- the *alternate composition* operator, denoted $\Diamond$, allows to specify an alternate 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: $\overline{P} \in \overline{\mathcal{P}}$.

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

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

- $\textit{options} : \mathcal{I} \rightarrow \widehat{\mathcal{P}}$, yields for any $i \in \mathcal{I}$, an intention plan of the form: $\widehat{P_i} = \Diamond_{P \in libp(i)} P$.
- $\textit{plan} : 2^I \rightarrow \overline{\mathcal{P}}$, 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 $\widehat{P_i} = \textit{options}(i)$ for each $i \in I$, are composed by using the AgLOTOS composition operators $|||$ and $\gg$.

In our approach, the mental process can label the different elements of the set I of intentions by using a weight function $\textit{weight} : I \longrightarrow \mathbb{N}$. This allows the planning process to schedule 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: $\textit{plan}(I) = \widehat{P_1} \gg (\widehat{P_0}|||\widehat{P_2}) \gg \widehat{P_3}$.

4.2 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 concurrent processes. A configuration (E, P) represents a process identified by P, such that its behavior expression is E. Table 2 recalls the Basic LOTOS semantics which formalizes how a process can evolve under the execution of actions. Here, it represents the operational semantics of *elementary plans*, viewed as processes. In particular, the last rule specifies how any (E, P) configuration is changed to (E', P) under any action a. Actually, $P := E$ denotes that the behavior expression E is assigned to P and $P \xrightarrow{a} E'$ represents the evolution of P from E to E'. Observe that for sake of simplicity, the notification that an action is launched is not represented in the semantics.

Table 2. Semantic rules of elementary plans

(Termination)	$\dfrac{}{exit \xrightarrow{\delta} stop}$	
(Action prefix)	$\dfrac{a \in \mathcal{O}}{a;E \xrightarrow{a} E}$	
(Choice)	$\dfrac{E \xrightarrow{a} E'}{F \ [] \ E \xrightarrow{a} E' \qquad E \ [] \ F \xrightarrow{a} E'}$	
(Concurrency)	$\dfrac{E \xrightarrow{a} E' \qquad a \notin L \cup \{\delta\}}{E \ \|[L]\| \ F \xrightarrow{a} E' \ \|[L]\| \ F}$ $\dfrac{E \xrightarrow{a} E' \quad F \xrightarrow{a} F' \quad a \in L \cup \{\delta\}}{E \ \|[L]\| \ F \xrightarrow{a} E' \ \|[L]\| \ F'}$	$\dfrac{E \xrightarrow{a} E' \qquad a \notin L \cup \{\delta\}}{F \ \|[L]\| \ E \xrightarrow{a} F \ \|[L]\| \ E'}$
(Hiding)	$\dfrac{E \xrightarrow{a} E' \qquad a \notin L}{hide \ L \ in \ E \xrightarrow{a} hide \ L \ in \ E'}$	$\dfrac{E \xrightarrow{a} E' \qquad a \in L}{hide \ L \ in \ E \xrightarrow{\tau} hide \ L \ in \ E'}$
(Sequence)	$\dfrac{E \xrightarrow{a} E' \qquad a \neq \delta}{E \gg F \xrightarrow{a} E' \gg F}$	$\dfrac{E \xrightarrow{\delta} E'}{E \gg F \xrightarrow{\tau} F}$
(Interruption)	$\dfrac{E \xrightarrow{a} E' \qquad a \neq \delta}{E \ [> \ F \xrightarrow{a} E' \ [> \ F}$ $\dfrac{F \xrightarrow{a} F'}{E \ [> \ F \xrightarrow{a} F'}$	$\dfrac{E \xrightarrow{\delta} E'}{E \ [> \ F \xrightarrow{\delta} E'}$
(Relabeling)	$\dfrac{E \xrightarrow{a} E' \qquad a \notin \{a_1,...,a_n\}}{E[b_1/a_1,...,b_n/a_n] \xrightarrow{a} E'[b_1/a_1,...,b_n/a_n]}$ $\dfrac{E \xrightarrow{a} E' \qquad a = a_k \ (1 \leq k \leq n)}{E[b_1/a_1,...,b_n/a_n] \xrightarrow{b_k} E'[b_1/a_1,...,b_n/a_n]}$	
(Plan definition)	$\dfrac{P := E \quad E \xrightarrow{a} E'}{P \xrightarrow{a} E'}$	

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

Definition 2. *Any agent plan configuration* $[\overline{P}]$ *has a generic representation defined by the following two rules:*

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

Table 3 shows the operational semantic rules defining the possible planning state changes for the agent. The rules apply from any HoA configuration $q = (bdi, ps)$, where the planning state ps is directly specified as an agent plan configuration, like $[\overline{P}]$. In each row of the table, there are three kinds of derivations: *(1)* the first rule shows the nominal case considering the execution of any action a ($a \in \mathcal{O} \cup \{\tau\}$), *(2)* and *(3)* the other two rules focus on the termination action of some intention plan, $\widehat{P}$. In the second one, the considered intention plan is successfully terminated whereas in the third 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 $\mathcal{PS}$ is the set of all the possible planning states for the agent, then the transition relation between the planning states is a subset of $\mathcal{PS} \times Act \times (\widehat{\mathcal{P}} \cup \neg\widehat{\mathcal{P}}) \times \mathcal{PS}$. The transitions $(ps_1, a, \widehat{P}, ps_2)$ and $(ps_1, a, \neg\widehat{P}, ps_2)$ such that $\widehat{P} \in \widehat{\mathcal{P}}$, provoke an internal event informing the mental process of the termination of the intention plan $\widehat{P}$. For sake of clarity, the transition (ps_1, a, nil, ps_2) is simply denoted $ps_1 \xrightarrow{a} ps_2$, 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 second rule captures the successful termination of $\widehat{P}$, under the execution of the action δ, while the third 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 context of the agent. In the first *Alternate rule*, the behavior expression $E = \Diamond^{k=1..n} E_k$ of an intention plan $\widehat{P}$, is refined by using the mapping *select*, the role of which is to select 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 used to express the compositions of intention plan configurations, in a sequential or parallel way. Here again, the rules are refined to take possible failure cases into account.

Table 3. Semantic rules of agent plan configurations

(Action)	$\dfrac{E \xrightarrow{a} E'}{(E,\widehat{P}) \xrightarrow{a} (E',\widehat{P})}$ $\dfrac{E \equiv fail}{(E,\widehat{P}) \xrightarrow[\neg\widehat{P}]{\tau} (stop,\widehat{P})}$	$\dfrac{E \xrightarrow{\delta} stop}{(E,\widehat{P}) \xrightarrow[\widehat{P}]{\tau} (stop,\widehat{P})}$
(Alternate)	$\dfrac{E_j \xrightarrow{a} E'_j \quad E_j = select(\widehat{P})}{(\Diamond^{k=1..n} E_k,\widehat{P}) \xrightarrow{a} (E'_j \triangleright (\Diamond^{k=1..n}_{k\neq j} E_k),\widehat{P})}$ $\dfrac{E \equiv fail \quad F \xrightarrow{a} F'}{(E \triangleright F,\widehat{P}) \xrightarrow{a} (F',\widehat{P})}$	$\dfrac{E \xrightarrow{\delta} stop}{(E \triangleright F,\widehat{P}) \xrightarrow[\widehat{P}]{\tau} (stop,\widehat{P})}$
(Sequence)	$\dfrac{ps_1 \xrightarrow{a} ps'_1}{ps_1 \gg ps_2 \xrightarrow{a} ps'_1 \gg ps_2}$ $\dfrac{ps_1 \xrightarrow[\neg\widehat{P}]{\tau} ps'_1}{ps_1 \gg ps_2 \xrightarrow[\neg\widehat{P}]{\tau} ps'_1 \gg ps_2}$	$\dfrac{ps_1 \xrightarrow[\widehat{P}]{\tau} ps'_1}{ps_1 \gg ps_2 \xrightarrow[\widehat{P}]{\tau} ps'_1 \gg ps_2}$
(Parallel)	$\dfrac{ps_1 \xrightarrow{a} ps'_1}{ps_1 \mid\mid\mid ps_2 \xrightarrow{a} ps'_1 \mid\mid\mid ps_2 \quad ps_2 \mid\mid\mid ps_1 \xrightarrow{a} ps_2 \mid\mid\mid ps'_1}$ $\dfrac{ps_1 \xrightarrow[\widehat{P}]{\tau} ps'_1}{ps_1 \mid\mid\mid ps_2 \xrightarrow[\widehat{P}]{\tau} ps'_1 \mid\mid\mid ps_2 \quad ps_2 \mid\mid\mid ps_1 \xrightarrow[\widehat{P}]{\tau} ps_2 \mid\mid\mid ps'_1}$ $\dfrac{ps_1 \xrightarrow[\neg\widehat{P}]{\tau} ps'_1}{ps_1 \mid\mid\mid ps_2 \xrightarrow[\neg\widehat{P}]{\tau} ps'_1 \mid\mid\mid ps_2 \quad ps_2 \mid\mid\mid ps_1 \xrightarrow[\neg\widehat{P}]{\tau} ps_2 \mid\mid\mid ps'_1}$	

Application to the Scenario. Let us take the scenario of section 3 again. Table 4 separately represents a possible evolution of the HoA configurations for the agent Alice and Bob, to solve the Alice's problem. In order to express the agent plan configurations, BDI propositions and plan of actions are simply expressed by using instantiated predicates, e.g. $get\_copies(\ell_2)$. Intention plans are composed from elementary plans which are viewed as concurrent processes, terminated by $exit$.

The initial configurations of Alice and Bob are respectively q_0^A and q_0^B, such that Alice is in ℓ_1 and has the mentioned two inconsistent desires, whereas Bob is in ℓ_2 and has expressed the desire to work with Alice. The current intention of Alice is only to meet with Bob. Here, BDI information is simply expressed by using intuitive predicate assertions.

The mental process can order the set of intentions to be considered. For instance, the intentions of Bob in q_1^B: $I_1 = \{meeting(Alice, \ell_1), getting\_copies(\ell_2)\}$ are ordered such that $weight(meeting(Alice, \ell_1)) < weight(getting\_copies(\ell_2))$. In the intention set I_1 of Bob, the corresponding agent plan configuration is: $[\widehat{P_1}] = ((E_g, \widehat{P_g}) \gg (E_m, \widehat{P_m}))$, where $(E_g, \widehat{P_g})$ and $(E_m, \widehat{P_m})$ are the two intention plan configurations of Bob. The first one corresponds to the inten-

Table 4. A state evolution for Alice and Bob

Alice's scenario	
q_0^A	$B_0 = \{in(me, \ell_1), in(copies, \ell_2)\}$ $D_0 = \{meeting(Bob, \ell_1), getting\_copies(\ell_2)\}$ $I_0 = \{meeting(Bob, \ell_1)\}$ $[\overline{P_0}] = (meet(Bob); exit, \widehat{P_m})$
q_1^A	$B_1 = \{in(me, \ell_1), in(copies, \ell_2), in(Bob, \ell_2)\}$ $D_1 = \{meeting(Bob, \ell_1), asking(Bob, get\_copies(\ell_2))\}$ $I_1 = \{meeting(Bob, \ell_1), asking(Bob, get\_copies(\ell_2))\}$ $[\overline{P_1}] = (meet(Bob); exit, \widehat{P_m}) \ ||| \ (Bob!(get\_copies(\ell_2)); exit, \widehat{P_a})$

Bob's scenario	
q_0^B	$B_0 = \{in(me, \ell_2)\}$ $D_0 = \{waiting(\nu), meeting(Alice, \ell_1)\}$ $I_0 = \{waiting(\nu), meeting(Alice, \ell_1)\}$ $[\overline{P_0}] = (Alice?(\nu); exit, \widehat{P_w}) \ ||| \ (move(\ell_1); meet(Alice); exit, \widehat{P_m})$
q_1^B	$B_1 = \{in(me, \ell_2), in(copies, \ell_2)\}$ $D_1 = \{meeting(Alice, \ell_1), getting\_copies(\ell_2)\}$ $I_1 = \{meeting(Alice, \ell_1), getting\_copies(\ell_2)\}$ $[\overline{P_1}] = (get\_copies(\ell_2); exit, \widehat{P_g}) \ \gg \ (move(\ell_1); meet(Alice); exit, \widehat{P_m})$

tion $getting\_copies(\ell_2)$ and the second to $meeting(Alice, \ell_1)$, such that $E_g = get\_copies(\ell_2); exit$ and $E_m = move(\ell_1); meet(Alice); exit$.

An example of execution derived from the initial planning state of Bob in q_1^B 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:

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

$\xrightarrow[\widehat{P_m}]{\tau} (stop, \widehat{P_m})$.

The reader may notice that the Alice and Bob agent plan configurations in q_1^A and q_1^B can change according to their revised intentions. Section 4.3 enriches the semantics with the plan revising service. This scenario will be taken again as an illustration.

4.3 Dynamical Plan Revising

In our model, the mental process drives the planning process such that adding or removing intentions possibly provoke the change of the agent plan. Of course, the mental process cannot ask for such a change on some plan is in progress, since this could imply that the agent could fall down in an incoherent state. In fact, the mental process must be informed by the planning process about the terminations of sub-plans, in particular 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 organization of the weighted intentions, required by the mental process.

A rough approach would consist in waiting the whole plan termination, meaning that the planning process reaches the final planning state of the current agent plan, before taking the change of intentions into account. In this paper, we propose an improved method which consists in updating the agent plan as the revisings of intentions are required by the mental process. Such updates consist 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 specifies different intention plan configurations corresponding to intentions to be achieved.

In some HoA configuration $q = (bdi, ps)$, $I(bdi)$ represents the intention set in this configuration. Let us consider some planning state $ps = [\overline{P}]$, 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 updatings of the intention set and of the corresponding agent plan rely on the following principles:

- according to the semantics, every termination of intention plan produces an internal event which changes the BDI state, in particular the updating of the intention set.
- 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)* $remain : \overline{\mathcal{P}} \times I \rightarrow \mathcal{E} \times \widehat{\mathcal{P}}$ maps each intention $i \in I$ to the corresponding pair $(E, \widehat{P})$ of $[\overline{P}]$; *(2)* $index : \widehat{\mathcal{P}} \rightarrow I$ 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 of updating operations are formalized by the following two mappings: $add, remove : 2^{\mathcal{E} \times \widehat{\mathcal{P}}} \times \mathcal{I} \rightarrow [\overline{\mathcal{P}}]$, be the mapping which builds an agent plan configuration $[\overline{P}]$ from a set of intention plan configurations.

The *Adding* of a new intention k, assuming its intention plan configuration is $(E_k, \widehat{P_k})$, means:

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

Formally, let ps be the current planning state of the agent and k be the intention to be added, the resulting planning state after the revising is defined by: $(add(\cup_{i \in I}\, remain(ps, i), k))$.

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 after removing the intention k is: $(remove(\cup_{i \in I}\, remain(ps, i), k))$.

Application to the scenario. Consider the example of Table 4 again, the changes of 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 ($e_1 = perc(in(Bob, \ell_2))$), meaning in the same location as the exam copies, Alice enriches her beliefs, desires and intentions, aiming communicating with Bob and asking for his help to bring her the copies. Consequently, she evolves to the new 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 $Alice?(\nu)$ in q_0^B. The reception of the message sent by Bob triggers an event at the Bobs mental process level. Since here Bob is accepting bringing the copies to Alice, he expands his beliefs ($in(copies, \ell_2)$) and also considers a new intention $getting\_copies(\ell_2)$ to take into account, in fact consistent with the previous ones. Consequently, the HoA configuration of Bob is changing to q_1^B i.e. $I_1 = I_0 \cup \{getting\_copies(\ell_2)\}$, and the plan $[\overline{P_0}]$ 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.

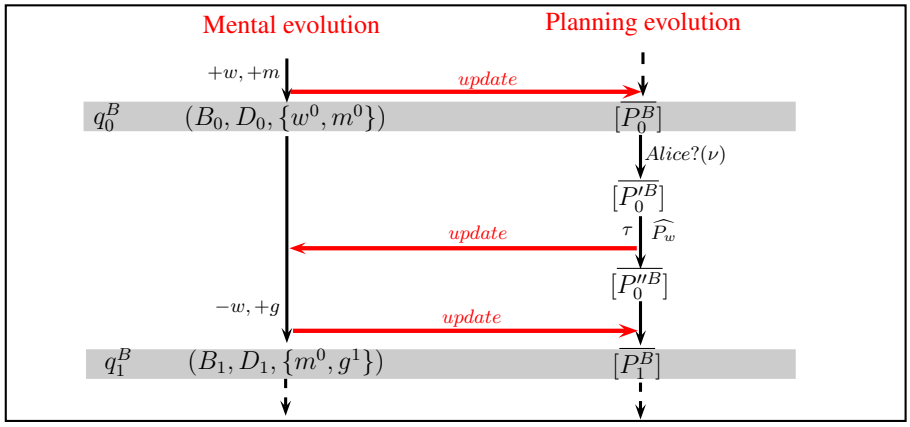

Fig. 5. HoA evolution of the agent Bob

The sequence diagram of Figure 5 focuses on the behavior of Bob, in order to highlight the mutual updates required by the mental process and the planning process, to synchronize the content of the intention set and the specification of the agent plan. For sake of simplicity, notice that:

- w and m respectively represent the two initial intentions of Bob in I_0, that are $waiting(\nu)$ and $meeting(Alice, \ell_1)$,
- g represents the intention $getting\_copies(\ell_2)$

Moreover, the signs $+$ and $-$ respectively mean adding the intention and removing it.

The bold horizontal arrows show the updates of this part of the scenario. First, the agent plan is built from the fact that w and m are added to the intentions set (with the same weight). This results in two concurrent intention plans, $[\overline{P_0^B}] = (E_w, \widehat{P_w})|||(E_m, \widehat{P_m})$. Due to the reception of the Alice's message, this yields to the execution of the reception action, followed by the exit action (τ) mentioning the termination of the intention plan $\widehat{P_w}$. This yields the agent plan $[\overline{P_0''^B}] = (E_m, \widehat{P_m})$. Since the mental process is informed of both actions (by internal events), it finally decides to update its set of intentions, with $-w, +g$ (m has a lower weight than g). As a consequence, the planning process is triggered to revise its agent plan in accordance, so $[\overline{P_1^B}] = (E_g, \widehat{P_g}) \gg (E_m, \widehat{P_m})$.

5 Contextual Planning Services

We now augment the planning process by two new services, exploiting the contextual information of the agent. Our aim is to provide a guidance service for the mental process and a model checking approach to analyze some temporal properties over the agent plan. Both services are based on the building of a specific transition system called *Contextual Planning Systems* (*CPS*), that can represent the different execution traces the agent could perform, from a given HoA configuration, in the best case, assuming the information context of the HoA configuration hold.

5.1 Building of a Contextual Planning System

Let consider any HoA configuration, the *CPS state* of the agent is now defined contextually to this configuration, taking into account the agent location and a termination information about the different intention plans.

Let q be any HoA configuration of the agent, and $I(q)$ its intention set in q, the rules used to build the CPS, take into account contextual information of three kinds which are: *(1)* the reached location in a CPS state, *(2)* the set of intention plans that are terminated when reaching a CPS state, and *(3)* more globally, the set $\Lambda(q)$ of neighbors currently known by the agent.

Definition 3. *A* CPS state *is a tuple* (ps, ℓ, T)*, where* ps *is the current planning state of the agent which represents its agent plan configuration* $[\overline{P}]$*,* ℓ *corresponds to the location within which the agent is placed, and* T *is the subset of intention plans which are terminated.*

Table 5 brings out the operational semantic rules, defining so the possible ways of CPS state changes. These rules are applied from an initial CPS state and $([\overline{P}], \ell, \emptyset)$ means that the agent is initially considered at location ℓ, and its plan configuration is $[\overline{P}]$.

The *Action rules* are used to derive the execution of an action with respect to an intention plan. The left hand side rule exhibits the case of a regular action, whereas the right hand side one shows the termination case of an intention plan,

Table 5. Semantic rules of CPS state changes

(Action)	$\dfrac{ps \xrightarrow{a} ps' \qquad a \in \mathcal{O} \cup \{\tau\}}{(ps,\ell,T) \xrightarrow{a} (ps',\ell,T)}$	$\dfrac{ps \xrightarrow[\widehat{P}]{\tau} ps'}{(ps,\ell,T) \xrightarrow{\tau} (ps',\ell,T \cup \{\widehat{P}\})}$
(Mobility)	$\dfrac{ps \xrightarrow{move(\ell')} ps' \qquad \ell \neq \ell'}{(ps,\ell,T) \xrightarrow{move(\ell')} (ps',\ell',T)}$	$\dfrac{ps \xrightarrow{move(\ell)} ps'}{(ps,\ell,T) \xrightarrow{\tau} (ps',\ell,T)}$
(Communication)	$\dfrac{ps \xrightarrow{x!(\nu)} ps' \qquad x \in \Lambda}{(ps,\ell,T) \xrightarrow{x!(\nu)} (ps,\ell,T)}$	$\dfrac{ps \xrightarrow{x?(\nu)} ps' \qquad x \in \Lambda}{(ps,\ell,T) \xrightarrow{x?(\nu)} (ps',\ell,T)}$

wherein the terminated intention plan is added to T. Also, the *Mobility rules* capture the change of location from ℓ to ℓ' and in the *Communication rules*, the action send $x!(\nu)$ (resp. receive $x?(\nu)$) is constrained by the visibility of the agent x in its neighborhood.

Observe that due to the fact we consider a predictive approach in this section, only successful executions are taken into account, thus abstracting that a plan may fail. Moreover, the semantics of the alternate operator is reduced to a simple non-deterministic choice of LOTOS: $\Diamond^{k=1..n} E_k \equiv [\,]^{k=1..n} E_k$ in order to possibly try every elementary plan to achieve the corresponding intention.

Definition 4. *Let $q = (bdi, ps)$ be any HoA configuration of the agent, the Contextual Planning System of q, denoted $CPS(q)$, is a labeled kripke structure $\langle S, s_0, Tr, \mathcal{L}, \mathcal{T} \rangle$ where:*

- *S is the set of CPS states,*
- *$s_0 = (ps, \ell, \emptyset) \in S$ is the initial CPS state, such that $ps = [\overline{P}]$ represents the current planning state of the agent in q and ℓ its current location,*
- *$Tr \subseteq S \times Act \times S$ is the set of transitions. The transitions are denoted $s \xrightarrow{a} s'$ such that $s, s' \in S$ and $a \in \mathcal{O} \cup \{\tau\}$,*
- *$\mathcal{L} : S \to \Theta$ is the location labeling function,*
- *$\mathcal{T} : S \to 2^{\widehat{\mathcal{P}}}$ is the termination labeling function which captures the terminated intention plans in some CPS state.*

Moreover, in a CPS, the transitions from a CPS state only represent actions that are realizable. In this paper, actions are modeled by instantiated predicates and their execution in CPS state is submitted to pre-conditions to be satisfied with respect to the contextual information known in that state, e.g. $pre(x!(\nu)) = pre(x?(\nu)) = x \in \Lambda$.

Remark 1. (Realizable actions) Let $pre(a)$ be the pre-condition of the action a. A transition $(s \xrightarrow{a} s') \in Tr$ is realizable in the state s of $CPS(q)$ iff $pre(a) \subseteq \mathcal{L}(s) \cup \Lambda(q)$.

Consider Table 4 where $[\overline{P_1}]$ is the agent plan configuration considered for Bob in the HoA configuration q_1^B. The initial *CPS* state s_0 in q_1^B is written:

$s_0 = ([\overline{P_1}], \ell_2, \emptyset)$ where $[\overline{P_1}] = ((E_g, \widehat{P_g}) \gg (E_m, \widehat{P_m}))$. An example of trace of $CPS(q_1^B)$ derived from s_0 is: $((E_g, \widehat{P_g}) \gg (E_m, \widehat{P_m}), \ell_2, \emptyset) \xrightarrow{getc} ((E'_g, \widehat{P_g}) \gg (E_m, \widehat{P_m}), \ell_2, \emptyset) \xrightarrow{\tau} ((E_m, \widehat{P_m}), \ell_2, \{\widehat{P_g}\}) \xrightarrow{move(\ell_1)} ((E'_m, \widehat{P_m}), \ell_1, \{\widehat{P_g}\}) \xrightarrow{meet} ((E''_m, \widehat{P_m}), \ell_1, \{\widehat{P_g}\}) \xrightarrow{\tau} (stop, \ell_1, \{\widehat{P_g}, \widehat{P_m}\})$.

5.2 Planning Guidance

In a CPS, the mapping $\mathcal{T}(s)$ captures the termination of intention plans in the CPS state s, hence the satisfaction of the corresponding intentions. In order to guide the agent, the planning process can select an execution trace through the plan, which maximizes the number of intention terminations.

Definition 5. *(Maximum trace) Let end* $: \Sigma \longrightarrow 2^{\widehat{\mathcal{P}}}$ *specify the set* $end(\sigma)$ *of termination actions that occur in a trace* σ*, then, the trace* σ *is said to be maximum iff there is no trace* σ' *in* $CPS(q)$ *such that* $|end(\sigma')| > |end(\sigma)|$.

As a corollary, a similar technique can be used to check the consistency of the agent intentions. With respect to any HoA configuration q, the comparison between one *maximum trace* σ in $CPS(q)$ and $|end(\sigma)|$ allows one to exhibit two extreme cases:

- if $|end(\sigma)| = |I(q)|$, we conclude that all the intentions of $I(q)$ are consistent,
- if $|end(\sigma)| = 0$, there is no satisfied intention, so the agent plan $\overline{P}$ of q is unappropriated with respect to the set of agent intentions.

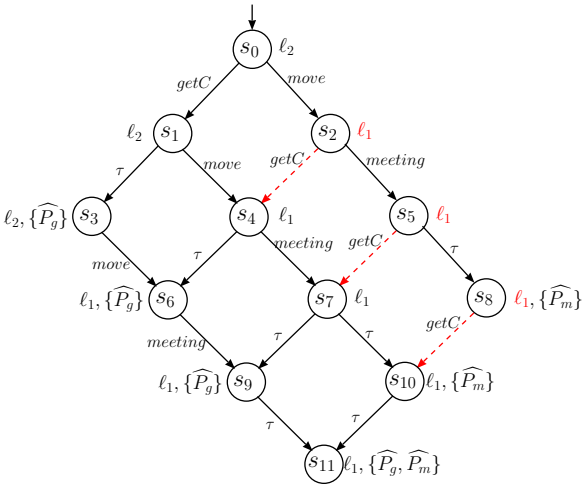

Fig. 6. $CPS(q_1^B)$ corresponding to the plan $\overline{P_1}$ of Bob

We reconsider the scenario of Table 4 to achieve the intentions of Bob in a parallel way: $[\overline{P_1}] = ((E_g, \widehat{P_g})|||(E_m, \widehat{P_m}))$ is the agent plan configuration considered for Bob. The Contextual Planning System $CPS(q_1^B)$ is highlighted in Figure 6. It is built from the initial CPS state $([\overline{P_1}], \ell_2, \emptyset)$ in q_1^B. In the figure, the dashed edges represent the unrealized transitions from the states $s \in \{s_2, s_5, s_8\}$, because $pre(getC) = \{\ell_2\} \not\subseteq \mathcal{L}(s) \cup \Lambda(q)$.

Model Checking of a Plan Properties

The fact that $CPS(q)$ is a kripke structure, allows one to process temporal logic verification, such as LTL or CTL. Due to the contextual labeling of states in $CPS(q)$, the properties to be checked concern the mobility, the communication and the terminations of intention plans. For instance, with respect to the $CPS(q_1^B)$ of Figure 6, the CTL-formula `AF(`ℓ_1`)` is checked `true`. This property means that the agent eventually reaches or crosses the location ℓ_1 on all the possible traces featured in the CPS. The use of model checking techniques is rather large since the possible properties cover both safety and liveness considerations. In fact, we view the CPS-based model checking service as a way to reinforce the planning guidance.

6 Experimentation: The Smart-Campus Project

We experiment our agent-based approach in a distributed system project called *Smart-Campus*. The aim is to assist the users of a complex Universitary campus in their activities. Concretely, we equip a float of (Android) smart-devices by the smart-campus application. In this application, the software architecture is composed of an HoA agent and a specific graphical user interface (GUI) to interact with the user to be assisted. Basic services are currently implemented over the HoA architecture, based on physical localization and (a)synchronous communication mechanisms. They are supported by the smart-device API facilities, in particular the Wireless Local Area Network (WiFi) API. As an example, the navigation service takes profit from the underlyied localization service to determine on the fly, the position and the move of the assisted user. Moreover, the contextual guidance service allows the agent to assist the user in realizing his desires in proposing different alternatives.

At the level of the Smart-Device (SD), the campus is concretized by the starting service which automatically connects the SD to the `"CAMPUS"` network, through one of the possible WiFi Access Points (AP). As illustrated in Figure 7, the SD can automatically access to the server *"SC Directory"*, dedicated to the smart-campus. This server is viewed as a middleware which maintains the persistence of contextual information like the discovery and the locations of other SD and objects concerning the campus. The starting service is also dedicated to declare the public information of the user to the server, in particular its location. One of the specificity of this project is that the agent embedded in the SD remains autonomous when the SC directory cannot be reached or when

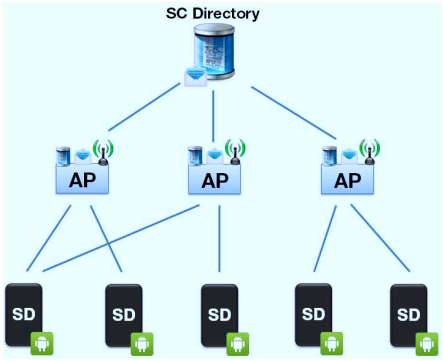

Fig. 7. Smart-campus architecture

the user is exiting the campus. It can continue assisting the user, due to the context information and persistent data previously stored in the SD, that can be pervasively updated with the help of other neighbor agents.

Observe that the localization service must work over the campus ground as well as the different stairs of the buildings. The best localization indoor technique is currently a research in progress e.g. [15]. Currently, we use different WiFi access points within the campus to compute the geographical locations, since this works in both indoor and outdoor locations. Anyway, the localization process requires a tune calibration phase to store specific information in the SC directory, concerning a set of physical reference points that must be selected over the campus, as mentioned in the finger printing approach, e.g. in [16].

In our case, information includes the physical location of the reference point (GPS), its symbolic name (place/room/corridor) and above all the perceived signal attenuation (RSSI[3]) from that location, in respect to the different WiFi access points. The localization service on the SD can then compare its proper perceptions of the WiFi attenuation in respect to the same references stored in the server, so that to deduce an approximation of its position through statistical computations and trilateration concepts.

GUI is also an important issue in our application since the user is definitively a mental process and also have physical capabilities expressed by all its senses. In particular, the move action finally results in some notification/proposal to the user which can walk in the good direction. Actually, all the actions that the HoA agent cannot perform can be delegated either to other agents or to some connected user, accordingly to the required capacities. As a specific GUI, graphical maps are modern and useful interfaces for the users. Our application is able to manage the maps of the campus, over which additional layers are used to render maps interactive and to show different locations and paths.

[3] RSSI: Received Signal Strength Indication

7 Related work

Since the last decade, several MAS projects were proposed embedding AmI systems. In conjunction with them, an important issue was to understand how AmI agents can interact and reason within a dynamic environment. One important topic was thus to model context-awareness within agent. Among the different existing works, different management of contexts were proposed: In [17], a hierarchical space system is considered which allows to directly specify location and some context elements for each agent moving in the space. Moreover, a list of interesting AmI system projects is brought out in the paper. In [17], the system view is abstracted but the agent context is modeled as a dynamic structure over which coordination activities are recognized by a pattern matching technique.

Among the different existing works, different management of contexts were proposed, and were dedicated to the description of AmI applications and scenarios, e.g. in [1, 2, 3]. These works concentrate on engineering issues of AmI systems while taking into account sophisticated contexts. In our formal approach, the context we deal with only refers to location and neighboring information, however, it could be extended with additional managements and techniques.

The fact to consider MAS as distributed systems composed of different communicating entities is not new, however the first proposals either do not include any built-in capacity for "lookahead" type of planning or they do it only at the implementation level without any precise defined semantics. Since 2006, BDI agent-centric approaches emerge to cope with the dynamicity of the agent environment. In [6], a hierarchical model (HTN) is proposed to better control the scheduling of plans in BDI agents, following an alternating goal-plan oriented strategy. Later on the same bases, Sardina et al. [18] showed how to specify and test learning appoaches in some particular cases. In [8], they proposed a formal procedure to manage and update the intention sets according to the agent BDI attitudes and the occurrences of new events.

Our proposed model is also agent-centric and accords with the principle of tightly controlling plan from the BDI mental attitudes. In contrast to the former works, the realizations of different intentions can be simultaneously considered and concurrently executed, in a higher level planning model. Conflicts are assumed to be solved by the mental process of the agent, however with the help of intention priorities. It is worth noticing that the conflicts which are caused by the contextual information are taken into account when building a CPS from the intention set. Dealing with action dependencies to restrain the agent activity have already largely been studied in the literature. In particular a GraphPlan planner can efficiently produce a global plan as a flow of actions, that corresponds to the subset of the desires that could be executed concurrently [10]. However, GraphPlan only deals with some of the possible schedulings between actions, since it follows a global time step approach. In contrast, our approach takes all the possible cases into account.

The on the fly revision of plans throughout the evolution of the agent is in fact a recent research topic in MAS approaches, e.g. [19]. In [8], the authors propose to revise the agent plans from the modifications of the agents goals, in particular

from the fact that some plans corresponding to intentions can be conflictual. However, the proposed solution comes from the fact that actions are processed atomically, so without concurrency, and in a tight alternate of goals and basic actions, expressed in a hierarchical structure. The well-structured semantics of AgLOTOS, especially with the intermediary notions of intention plans allows us to smoothly develop the planning update service from some changes in the set of intentions. It is not related to the concurrency of intention plans and has the capacity of continuing the execution of intention plans which are not related to the change of some intention.

In the literature there are a number of (BDI) agent programming languages [20], highlighting different aspects or modules developed in agent softwares like goal, planning, organization, reinforcement learning, ..., e.g. S-CLAIM [21], Jason [22], 2APL [23] and JIAC [24]. Most of them lack from formal description, but, the work of [6] has extended some existing formal algebraic specification models dedicated to distributed systems, in order to specify the concurrency of actions in plans. The proposed language integrates BDI ingredients within plans in a unified and interactive way. Nevertheless, our formal algebraic language based on LOTOS, appears to be more expressive in its capacity to express plans as concurrent processes as well as concurrent actions. It is also operational by the way to handle action and plan failures in the AgLOTOS language and the HoA architecture. Lastly, pay attention that in our approach, a clear separation exists between the mental and planning levels. Actually, our planning process behaves like a service that could be embedded in existing BDI architectures.

The verification task standardly applied to MAS is mainly driven by a global vision of the system, e.g. in [25]. In [26], the reuse of some program checking techniques are proposed, based on a BDI representation of the system state space. Moreover, in order to cope with the well-known combinatorial explosion of states, abstraction/reduction techniques are applied over the BDI states. One could think to introduce similar techniques within AmI agents, however, the high-level dynamics that usually features an AmI environment could introduce too much states to consider, even after reduction. Moreover, it remains open to apply these techniques according to the changes of contexts. In our paper, the proposed techniques, guidance and plan model checking, capture the AmI context but remains agent-centric. This avoids considering the system combinatorial explosion problem while allowing to take into account the AmI system dynamicity.

Petri net-based models were proposed to represent some MAS paradigms. Basically, such models can easily capture resource notions and concurrency execution of actions and dynamic processes, e.g. [27, 28]. With some typing mechanisms (coloration), MAS distributed entities can be designed and their way to communicate between them, e.g. [29]. Thus, agent interactions can be handled at different operational levels, like agent plans [30]. As interesting works, the Petri Net modeling were useful to study the global coherencies of plans or agent interactions to recognize some FIPA protocols [31]. Nevertheless, many problems remain open in order to capture a whole AmI system requirements such as the

system openness or the dynamics inherent to the change of agent contexts. In fact, Petri Nets remain low level representations with respect to the concepts used to build the mental attitudes of the agent. In contrast, AgLOTOS brings out abstraction capacity and high level composition means whereas the proposed HoA model allows to correlate both mental and planning levels.

8 Conclusion

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 a BDI agent can change its BDI state adequately to the perceptions of new events. The proposed AgLOTOS process-based algebra appears to be a powerful and intuitive way to express an agent plan as a set of concurrent processes. The presented scenario shows how the AgLOTOS language is rich due to modularity concepts and concurrency operators.

We demonstrate that the proposed model is operational. Actually, plans are automatically built from the set of intentions of the agent and a library of elementary plans already expressed in AgLOTOS. In contrast to existing works, the planning process can execute the different intention sub-plans concurrently. Moreover, the planning structure appearing in the AgLOTOS expressions allows to automatize the plan revisions on the fly, accordingly to the updates of intentions. Hence, an AmI agent can be viewed as having only one plan, updated all along the evolution of the agent behavior.

On another point, the CPS structure appears to be an interesting state-transition structure to select optimal execution from the contextual situation of the agent. In this paper, the proposed guidance service based on the CPS is used to optimize the satisfaction of the agent intentions. Moreover, intention consistency can be checked over the CPS structure. One of our next perspectives will consist to reinforce the proposed guidance service and temporal model checking techniques could be used in that sense.

References

[1] Guivarch, V., Camps, V., Pninou, A.: Context awareness in ambient systems by an adaptive multi-agent approach. In Patern, F., Ruyter, B., Markopoulos, P., Santoro, C., Loenen, E., Luyten, K., eds.: Ambient Intelligence. Volume 7683 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 129–144

[2] Olaru, A., Florea, A.M., El Fallah Seghrouchni, A.: A context-aware multi-agent system as a middleware for ambient intelligence. MONET **18**(3) (2013) 429–443

[3] Pi, N.S., Griol, D., Carbó, J., López, J.M.M.: Evaluation of agents interactions in a context-aware system. T. Computational Collective Intelligence **9** (2013) 79–97

[4] Georgeff, M.P., Pell, B., Pollack, M.E., Tambe, M., Wooldridge, M.: The belief-desire-intention model of agency. In Mller, J.P., Singh, M.P., Rao, A.S., eds.: ATAL. Volume 1555 of Lecture Notes in Computer Science., Springer (1998) 1–10

[5] Chaouche, A.C., El Fallah Seghrouchni, A., Ilié, J.M., Saïdouni, D.E.: A Higher-order Agent model for ambient systems. Procedia Computer Science **21** (2013) 156 – 163
[6] Sardina, S., de Silva, L., Padgham, L.: Hierarchical planning in BDI agent programming languages: a formal approach. In: AAMAS '06. (2006) 1001–1008
[7] Icard III, T.F., Pacuit, E., Shoham, Y.: Joint revision of beliefs and intention. In: Principles of Knowledge Representation and Reasoning. (2010) 572–574
[8] Shapiro, S., Sardina, S., Thangarajah, J., Cavedon, L., Padgham, L.: Revising conflicting intention sets in BDI agents. In: AAMAS '12. (2012) 1081–1088
[9] Hoek, W., Jamroga, W., Wooldridge, M.: Towards a theory of intention revision. Synthese **155**(2) (2007) 265–290
[10] Meneguzzi, F., Zorzo, A.F., da Costa Móra, M., Luck, M.: Incorporating planning into BDI agents. Scalable Computing: Practice and Experience **8** (2007) 15–28
[11] Cohen, P.R., Levesque, H.J.: Intention is choice with commitment. Artif. Intell. **42**(2-3) (1990) 213–261
[12] d'Inverno, M., Kinny, D., Luck, M., Wooldridge, M.: A formal specification of dmars. In Singh, M.P., Rao, A.S., Wooldridge, M., eds.: ATAL. Volume 1365 of Lecture Notes in Computer Science., Springer (1997) 155–176
[13] Chaouche, A.C., El Fallah Seghrouchni, A., Ilié, J.M., Saïdouni, D.E.: A dynamical plan revising for ambient systems. Procedia Computer Science **32** (2014) 37 – 44
[14] Brinksma, E., ed.: ISO 8807, LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. (1988)
[15] Galvn-Tejada, C.E., Garca-Vzquez, J.P., Garca-Ceja, E., Carrasco-Jimnez, J.C., Brena, R.F.: Evaluation of four classifiers as cost function for indoor location systems. Procedia Computer Science **32**(0) (2014) 453 – 460
[16] Gansemer, S., Grossmann, U., Hakobyan, S.: Rssi-based euclidean distance algorithm for indoor positioning adapted for the use in dynamically changing wlan environments and multi-level buildings. In: Indoor Positioning and Indoor Navigation (IPIN). (2010) 1–6
[17] Olaru, A., Florea, A.M., El Fallah Seghrouchni, A.: Graphs and patterns for context-awareness. In: Ambient Intelligence - Software and Applications. Volume 92. (2011) 165–172
[18] Singh, D., Sardina, S., Padgham, L., James, G.: Integrating learning into a BDI agent for environments with changing dynamics. In Toby Walsh, C.K., Sierra, C., eds.: Proceedings of IJCAI'11. (2011) 2525–2530
[19] Grant, J., Kraus, S., Perlis, D., Wooldridge, M.: Postulates for revising bdi structures. Synthese **175**(1) (2010) 39–62
[20] Hindriks, K.: Twenty years of engineering multiagent systems. In: Keynote of the EMAS workshop, part of AAMAS'14. (2014)
[21] Baljak, V., Benea, M.T., El Fallah Seghrouchni, A., Herpson, C., Honiden, S., Nguyen, T.T.N., Olaru, A., Shimizu, R., Tei, K., Toriumi, S.: S-claim: An agent-based programming language for AmI, a smart-room case study. Procedia Computer Science **10**(0) (2012) 30 – 37
[22] 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., eds.: Multi-agent programming : languages, platforms and applications. Number 15. Springer, New York (July 2005) 3–37
[23] Dastani, M.: 2apl: a practical agent programming language. Autonomous Agents and Multi-Agent Systems **16**(3) (2008) 214–248

[24] Lützenberger, M., Küster, T., Konnerth, T., Thiele, A., Masuch, N., Heßler, A., Keiser, J., Burkhardt, M., Kaiser, S., Albayrak, S.: Jiac v: A MAS framework for industrial applications. In: Proceedings of AAMAS '13. (2013) 1189–1190
[25] Sudeikat, J., Braubach, L., Pokahr, A., Lamersdorf, W., Renz, W.: Validation of BDI agents. In: PROMAS. (2006) 185–200
[26] Dennis, L.A., Fisher, M., Webster, M.P., Bordini, R.H.: Model checking agent programming languages. Automated Software Engg. **19**(1) (March 2012) 5–63
[27] Dahmani, D., Ilié, J.M., Boukala, M.: Time recursive petri nets. In Jensen, K., Aalst, W., Billington, J., eds.: T. Petri Nets and Other Models of Concurrency I. Volume 5100 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2008) 104–118
[28] Saïdouni, D.E., Bouneb, M., Ilié, J.M.: Maximality semantic for recursive petri nets. In: ECMS. (2013) 544–550
[29] Kouah, S., Saïdouni, D.E., Ilié, J.M.: Synchronized petri net: A formal specification model for multi agent systems. Journal of Software **8**(3) (March 2013) 587–602
[30] Ziparo, V.A., Iocchi, L., Nardi, D., Palamara, P.F., Costelha, H.: Petri net plans: a formal model for representation and execution of multi-robot plans. In Padgham, L., Parkes, D.C., Mller, J.P., Parsons, S., eds.: AAMAS (1), IFAAMAS (2008) 79–86
[31] Mazouzi, H., El Fallah-Seghrouchni, A., Haddad, S.: Open protocol design for complex interactions in multi-agent systems. In: AAMAS'02. (2002) 517–526