Learning from situated experiences for a contextual planning guidance

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

Noname manuscript No.
(will be inserted by the editor)

Learning from Situated Experiences for a Contextual Planning Guidance

Ahmed-Chawki Chaouche · Amal El Fallah Seghrouchni · Jean-Michel Ilié · Djamel Eddine Saïdouni

Received: date / Accepted: date

Abstract This paper presents AgLOTOS as an algebraic language dedicated to the specification of agent plans in ambient systems. It offers two levels of plans: Elementary plans which are composed to produce an intention plan ; The intention plans which are, in turn, composed to build an agent plan. The composition relies on several operators such as alternative and concurrency. Consequently, the plans can be built automatically as a system of concurrent processes. At the execution level, our approach helps the agent to select an optimal plan preserving the consistency of its intentions. The selection is based on an original and formal construction called Contextual Planning System (CPS), which presents the potential paths with the associated contexts while removing the inconsistent options. Finally, our CPS is improved by using past-experiences for a better guidance of the agent.

Keywords Planning language · BDI agent · ambient system · intention consistency · past-experiences · contextual planning guidance

1 Introduction

Ambient Intelligence (AmI) is the vision of a ubiquitous electronic environment that is non-intrusive and proactive, when assisting people during various activities (**??**). For the design of such complex systems, Multi-Agent System (MAS) approaches offer interesting frameworks, since their agents are designed to be intelligent, proactive and autonomous (**?**).

Ahmed-Chawki Chaouche · Djamel Eddine Saïdouni
University Abdelhamid Mehri - Constantine 2, MISC Laboratory, Campus Ali Mendjeli, 25000 Constantine, Algeria.
Tel: +213-781-851-597 ; Fax: +33-144-277-000
E-mail: {ac.chaouche, saidouni}@misc-umc.org

Amal El Fallah Seghrouchni · Jean-Michel Ilié
LIP6, UMR 7606 UPMC - CNRS, 4 place jussieu 75005 Paris, France.
E-mail: {amal.elfallah, jean-michel.ilie}@lip6.fr

The major problem of an AmI agent is to recognize its context. This includes gathering information on its location and the ability to discover other agents. In (**?**), it is shown how autonomous Belief-Desire-Intention (BDI) agents (**?**) can evolve and move within an ambient environment, based on an agent centric approach and context-awareness. The proposed model, called Higher-order Agent (HoA), takes into account the major features and functionalities of AmI. In particular the AmI systems can be open i.e. agents can enter or leave the system. Locations may change, exposing the agent to different surroundings.

This paper introduces an efficient planning management process into the HoA architecture. In particular, we aim to endow each AmI agent with a powerful on the fly predictive service. The plan of a BDI agent can be viewed as a set of intentions produced by the BDI interpreter.

We adopt the formal description language for plans, namely *AgLOTOS*, introducing modularity and concurrency aspects for sub-plan composition. Unlike the formal description of **?**, the AgLOTOS semantics overpasses the sequential execution of sub-plans. Moreover, the fact that the agent plans are produced automatically from concurrent intentions, formally enforces the consistency of the agent, hence, extends the GraphPlan-based approaches (**?**).

The formal semantics of AgLOTOS is enriched to automatically produce a *Contextual Planning transition System* (*CPS*), which allows to automatically guide the agent over the possible execution of actions in plans. In the CPS structure, the selection of the possible actions is informed by the successes and failures concerning the running past-experiences of actions. In the literature, the benefits of a learning process have already been studied but under a huge number of iterations (**??**). Our learning approach is different and adapted to AmI and dynamic systems requiring an efficient activity of the agent even when the knowledge about the past-experiences is very limited.

The paper is organized as follows: Section 2 describes the functional architecture of the planning process, together with its guidance mechanism based on lookahead capability and past experience knowledges. In Section 3, the AgLOTOS specification language is briefly described and used to build an agent's plans from its intentions (e.g. composition of plans). In Section 4, the Contextual Planning System (CPS) is defined. Section 5 presents our guidance mechanism and a realistic scenario for illustration. In Section 6, we improve the guidance mechanism by exploiting the agent past-experiences to qualify the actions of the CPS. In Section 7, we study the complexity of the proposed approach and we show how decreasing the combinatorial effect within the CPS. Finally, Section 8 discusses our approach with regard to state of the art.

2 Contextual Planning Architecture

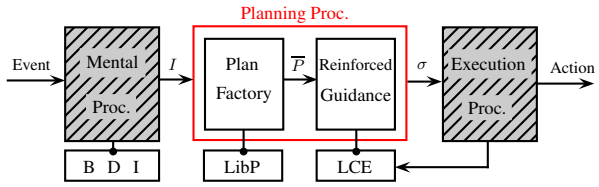

Fig. 1 View of the contextual planning architecture

The overall agent architecture, we deal with, is standardly based on several processes. Triggered by *events* expressing the context changes, the *mental process* is assumed to be highly deductive and rational, based on BDI structures (Beliefs, Desires and Intentions).

Figure 1 brings out a functional view of our *planning process*, in charge to offer some plan to execute (σ) from the current set of intentions (I). It is composed of two modules:

- The *Plan Factory* builds the so-called *agent plan* which globally corresponds to the current set of intentions. The *LibP library* is used to exploit the different alternative plans that must be specified for each intention. With respect to the current spatial context of the agent, a Contextual Planning System (CPS) is built. By its traces (paths), the CPS represents all the possible plans that can be realized.
- The *Guidance module* is able to enrich the CPS information taking profit from the past-experiences of actions, that are stored in the so-called *Learned Contextual Experiences (LCE) module*. This results in a structure called CPS with learning (CPS-L for short), which can be used to searching the optimal trace with respect to the agent context.

3 The AgLOTOS Specification Language

The behavior of the BDI agent we consider in this paper is carried out by two successive processes, as highlighted in Figure 2. As usual, the *Mental process* represents the reasoning mechanism, based on the beliefs (B), desires (D), and intentions (I), the instances of which define the BDI states of the agent. Triggered by the perceived events, the mental process manages/updates the B, D and I structures. In order to organize its selected intentions, the mental process is able to schedule them by associating each one a given weight (see Section 3.2).

From the set of intentions, the *Planning process* is called by the mental process to produce a plan of actions, helped by a library of plans (*LibP*). In our approach for each BDI state, the *Agent plan* is composed as follows: *(1)* the agent plan is made of sub-plans called *Intention plan*, each one dedicated to achieving a selected intention; *(2)* each intention plan is an alternate of several sub-plans, called *Elementary plans*, extracted from the *LibP* library. This allows one to consider different ways to achieve the associated intention (see Section 3.1). Further, we assume that the *LibP* library of elementary plans is indexed by the set of all the possible intentions for the agent.

In our approach, the associated notation of the planning language is described in Table 1.

Table 1 Synthetic presentation of the used notations

Notation	Description
E	Elementary 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

3.1 The Syntax of Elementary Plans

Elementary plans are written using the algebraic language *AgLOTOS* (**?**). This language extends the LOTOS language (**?**) in order to deal with the concurrency of actions in plans.

Let $\mathcal{O}$ be the (finite) set of observable actions which are viewed as instantiated predicates, ranged over $a, b, \ldots$ 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 entrance and leaving of other agents in the AmI system, *(2)* it

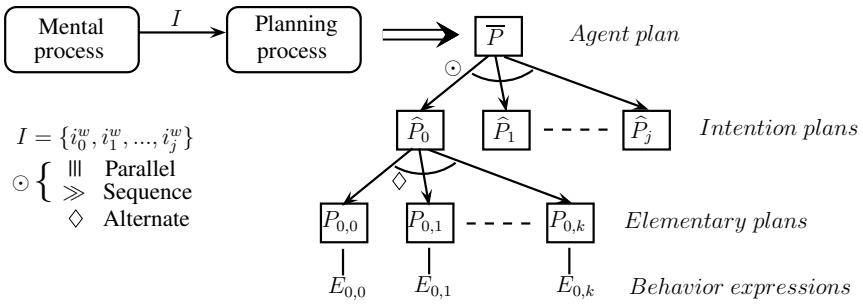

Fig. 2 Agent planning structure

can move between the AmI system locations 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 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 by the semantics of the π-calculus primitives. However, contrary to the classical use of the calculus, the communication support is completely dynamic, so there is no need to specify the predefined channels: the expression $x!(v)$ specifies the emission to the agent x ($x \in \Lambda$) of some message v ($v \in \mathcal{M}$), whereas, the expression $x?(v)$ means that v is received from some agent x.

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

The AgLOTOS language specifies for each elementary plan a name to identify it and an AgLOTOS expression to describe its behavior. Consider that the elementary plan's names are ranged over $P_1, P_2, ...$ 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 using LOTOS operators. The syntax of an AgLOTOS elementary plan P is defined inductively as follows:

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

In this syntax, $P ::= E$ represents an elementary plan identified by P, such that its behavior expression is E. The elementary expression *stop* specifies a plan behavior without possible evolution and *exit* represents the successful termination of some plan. The expression $a;E$ denotes an action a prefixing E and the set $\odot$ represents the standard LOTOS operators and some of them can refer to any subset L of $\mathcal{O}$. The expression $E\,[\,]\,E$ specifies a non-deterministic choice, $E \gg E$ a sequential composition and $E\ [> E$ the interruption. The LOTOS parallel composition, denoted $E\,|[L]|\,E$, can model *both synchronous* composition for actions defined in L, denoted $E\,||\,E$ with $L = \mathcal{O}$, and *asynchronous composition*, denoted $E\,|||\,E$ with $L = \emptyset$. The expression *hide L in E* represents an explicit hiding of actions mentioned in L, making them unobservable in E.

Every expression of a process is terminated by an *exit*. Here are now different examples of compositions between two sub expressions E_1 and E_2, with $E_1 ::= a;b;\textit{exit}$ and $E_2 ::= a;b;c;\textit{exit}$:

- $E ::= E_1 \gg E_2$ means that all the expression E_1 is realized before the expression E_2.
- $E ::= E_1\ |||\ E_2$ means that the expression E_1 and E_2 are realized in concurrence.
- $E ::= E_1\ [\,]\ E_2$ means that only one of E_1 and E_2 is realized
- $E ::= E_1\ [> E_2$ means that the expression E_1 can no more be executed when the execution of E_2 starts.
- $E ::= E_1\ |[b]|\ E_2$, means that the joint execution of b in E_1 and E_2 is possible only when b is possible in both E_1 and E_2.
- $E ::= \textit{hide } a \textit{ in } (E_1\ |[b]|\ E_2)$ means that the action a is hidden when a is offered in the expression $(E_1\ |[b]|\ E_2)$. In fact, this operator makes all the occurrences of the action a unobservable (replaced by τ) at the semantical level (see Section 4).

The AgLOTOS language is expressive due to its various operators, so that each execution plan is mainly based on the concurrent composition of elementary plans (one for each intention). Unlike LOTOS specification, it is worth noticing that AgLOTOS does not proposed any recursion mechanism based on the two following points:

– The mental attitudes of the agent can simulate a recursion by maintaining some intention, hence the possible associated elementary plans. Further in this paper, this AgLOTOS restriction guarantees that the semantic construction is finite, hence calculable.
– The maintainability of intentions is very efficient since an agent can maintain some intentions, even if the other intentions are revised (**??**),

3.2 Building 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 line with the intentions of the agent and associated weights.
– the *alternate composition* operator, denoted $\Diamond$, allows to specify an alternation of elementary plans. In particular, an intention is satisfied if and only if 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: $\overline{P} \in \overline{\mathcal{P}}$.

$$\widehat{P} ::= P \mid \widehat{P} \Diamond \widehat{P} \qquad \textit{Intention plan}$$
$$\overline{P} ::= \widehat{P} \mid \overline{P} ||| \overline{P} \mid \overline{P} \gg \overline{P} \qquad \textit{Agent plan}$$

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*:

– $libp : \mathcal{I} \rightarrow 2^{\mathcal{P}}$, features the library of elementary plans. It yields, for each intention $i \in \mathcal{I}$, a set of instantiated elementary plans dedicated for achieving i.
– $options : \mathcal{I} \rightarrow \widehat{\mathcal{P}}$, yields for any $i \in \mathcal{I}$, an intention plan of the form: $\widehat{P_i} = \Diamond_{P \in libp(i)} P$.
– $plan : 2^I \rightarrow \overline{\mathcal{P}}$, 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 $\widehat{P_i} = options(i)$ s.t. $i \in I$ are composed by using the AgLOTOS composition operators $|||$ and $\gg$.

In order to account for 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 $weight : I \rightarrow \mathbb{N}$. 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 $\gg$.

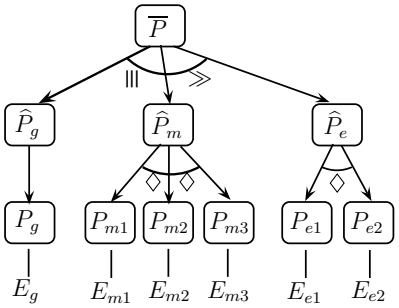

Fig. 3 An example of an agent plan structure

For instance, let $I = \{i_g^2, i_e^1, i_m^2\}$ be the considered set of intentions, such that the superscript information denotes a weight value. As illustrated in Figure 3, let $\widehat{P_g}, \widehat{P_e}, \widehat{P_m}$ be their corresponding intention plans, the constructed agent plan could be viewed as: $\overline{P} ::= (\widehat{P_g} ||| \widehat{P_m}) \gg \widehat{P_e}$, where $\widehat{P_g} ::= P_g$, $\widehat{P_m} ::= P_{m1} \Diamond P_{m2} \Diamond P_{m3}$ and $\widehat{P_e} ::= P_{e1} \Diamond P_{e2}$. Further, each elementary plan P_k is associated with an expression E_k describing its behavior.

3.3 A Simple AmI Example

Let us consider the *AmI University scenario* presented in (**?**) where Alice and Bob are two agents. The proposed problem of Alice is that she cannot make the two following tasks in the same time: *(1)* to meet Bob in the location ℓ_1, and *(2)* to get her exam copies from the location ℓ_2. Clearly, Alice's desires are conflictual since Alice cannot be in two distinct locations simultaneously.

Table 2 Scenarios for Alice and Bob

Alice's scenario
$I_A = \{meeting(Bob, \ell_1), receiving(Bob, confirm\_getc)\}$
$\overline{P_A} = meet(Bob); exit \mid\mid\mid Bob?(confirm\_getc); exit$

Bob's scenario
$I_B = \{meeting(Alice, \ell_1), getting\_copies(\ell_2)\}$
$\overline{P_B} = get\_copies(\ell_2); Alice!(confirm\_getc); exit \gg$ $move(\ell_1); meet(Alice); exit$

As illustrated in Table 2, the scenarios of Alice and Bob are specified separately. It is assumed that Bob and Alice may coordinate in order to achieve their intentions, at their mental process levels. The actions in plans are simply expressed using instantiated predicates, like $get\_copies(\ell_2)$. Intention plans are composed from elementary plans which are viewed as concurrent processes, terminated by *exit*, *a la LOTOS*.

The mental process can order the set of intentions to be considered. For instance, the intention set of Bob $I_B = \{meeting(Alice, \ell_1), getting\_copies(\ell_2)\}$ is ordered such that

$weight(meeting(Alice,\ell_1)) < weight(getting\_copies(\ell_2))$. Regarding the intention set I_B, the corresponding agent plan expression of Bob is: $\overline{P_B} = get\_copies(\ell_2);Alice!(confirm\_getc);exit \gg move(\ell_1);meet(Alice);exit$, which is built by using the *options* and *plan* mappings. The resulting agent plan expression for Bob is thus the following: act two successive processes. First, get the copies locally in ℓ_2 and confirm this to Alice ($Alice!(confirm\_getc)$). Second, move to ℓ_1 and meet Alice there. Pay attention that some actions can be processed concurrently, so is the case in the Alice' agent plan $\overline{P_A}$, for the two intention plans $meet(Bob);exit$ and $Bob?(confirm\_getc);exit$.

4 Contextual Planning Management

The AgLOTOS operational semantics is basically derived from the one of LOTOS. A pair (E,P) represents a process identified by P, such that its behavior expression is E. The semantic rules defining the possible elementary plan changes are detailed in Table 3 which formalize how a process can evolve under the execution of actions. In particular, the (Plan definition) rule specifies how a pair (E,P) is changed to (E',P) under any action a. Actually, $P := E$ means to consider any (E,P) source pair and $P \xrightarrow{a} E'$ means changing E to E' for P under the execution of a. As far as AgLOTOS is concerned, these rules also represent the operational semantics of elementary plans, viewed as processes.

We now describe how the expression of an *agent plan* is formed compositionally from the expressions of the *intention plans* of the agent, themselves built from an alternate of *elementary plans* and their behavior expressions. The state of an agent plan $\overline{P}$, also called an *agent plan configuration*, is denoted $[\overline{P}]$. From the plan structure of any agent plan, such configuration is described by associating a behavior expression to each elementary plan, and by composing them according to the used algebraic operators.

The *canonical rules* of Definition 1, specify how $[\overline{P}]$ is formed compositionally from some *intention plan configurations*, like $(E,\widehat{P})$ (*rule 1*), themselves built from an alternate of *elementary plan configurations*, like (E_k,P_k) (*rule 2*) which represents an elementary plan identified by P_k, and its behavior expression is E_k. Considering the example of Figure 3:
$[\overline{P}] = \Big((E_g,\widehat{P_g})|||(E_{m1}\Diamond E_{m2}\Diamond E_{m3},\widehat{P_m})\Big) \gg (E_{e1}\Diamond E_{e2},\widehat{P_e})$

Definition 1 Any plan configuration $[\overline{P}]$ has a generic representation defined by the following two rules:

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

We will now define the planning state of the agent contextually, taking into account the agent location and a termination information about the different intention plans defined for the agent.

Definition 2 A contextual planning state is a tuple (ps,ℓ,T), where ps is any planning state, ℓ corresponds to an expected location for the agent, and T is the subset of intention plans which are terminated.

Table 4 shows the operational semantic rules defining the possible contextual planning state changes for the agent. These rules are applied to produce a Contextual Planning transition System, called *CPS*, from an initial contextual planning state, e.g. $([\overline{P}],\ell,\emptyset)$, meaning that the agent is initially at location ℓ and the plan configuration $[\overline{P}]$ represents its planning state. There are two kinds of transition rules:

Intention plan level: When an intention plan is assumed to be treated, a transition $(ps_1,a,\widehat{P},ps_2)$, denoted $ps_1 \xrightarrow[\widehat{P}]{a} ps_2$, expresses a change of intention configuration, from ps_1 to ps_2. In table 4 (Intention plan level), the fist rule assumes the execution of the action a from $E \xrightarrow{a} E'$ and $P := E$, whereas the second one highlights the termination case, keeping trace of the intention plan $\widehat{P}$ that is going to be terminated. By calling *PS* the set of all the possible intention plan configurations for the agent, the transition relation is a subset of $PS \times Act \times \widehat{\mathcal{P}} \times PS$. For sake of clarity, the transition (ps_1,a,nil,ps_2) is simply denoted $ps_1 \xrightarrow{a} ps_2$.
Observe that due to the fact we consider a predictive guidance in the CPS, only successful executions are taken into account, thus abstracting that a plan may fail. Moreover, the semantics of the alternate operator is reduced to the simple non-deterministic choice of LOTOS: $\Diamond^{k=1..n}E_k \equiv [\,]^{k=1..n}E_k$ in order to take into account every elementary plan for achieving the corresponding intention.

Agent plan level: The possible changes of the contextual planning states, like (ps,ℓ,T), are expressed in Table 4. In (Action) rules, the first one exhibits the case of a regular action, whereas the second one specifies the termination case of some intention plan, which is added to T. In (Communication) rules, the action send $x!(v)$ (resp. receive $x?(v)$) is constrained by the visibility of the agent x in its neighborhood. In (Mobility) rules, the effect of the action $move(\ell')$ yields the agent to be placed in ℓ'.

5 Guidance Mechanism

From any set of intentions in the agent, denoted I, a Contextual Planning System is built by using the rules of Table 4 and taking into account contextual information of three kinds: *(1)* the reached location in a planning state, *(2)* the set

Table 3 Semantic rules of elementary plan configurations

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

of intention plans that are terminated when reaching a planning state, and *(3)*, more globally, the set Λ of neighbors currently known by the agent.

Definition 3 The CPS (Contextual Planning System) is a labeled Kripke structure $\langle S, s_0, Tr, \mathcal{L}, \mathcal{T} \rangle$ where:

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

In a CPS, any transition $s \xrightarrow{a} s'$ represents an action to be performed. Like in the STRIPS description language (**?**), the actions are associated with preconditions and effects. In our approach, the preconditions only concern the contextual information attached to the source state. Let $pre(a)$ be the precondition of any action a, e.g. $pre(a(\ell)) = \ell = \mathcal{L}(s)$. In Figure 4, the three actions that are not realizable are represented by dashed transitions from the states s_2, s_5 and s_9. From these states, $pre(getc(\ell_2)) = \ell_2 \neq \mathcal{L}(s)$.

In order to guide the agent efficiently, the planning process can select an execution trace which maximizes the number of intentions that can be achieved. This can be captured over the set $\Sigma \subseteq 2^{Tr}$ of all the possible traces of the CPS. We introduce the notion of *maximum trace* based on the mapping $end : \Sigma \rightarrow 2^{\widehat{\mathcal{P}}}$, used to specify the set $end(\sigma)$ of the different terminated intention plans that occur in a trace $\sigma \in \Sigma$. Let Σ_{MAX} represent the set of maximum traces of the CPS. For instance, the trace carried out by $s_0 \rightarrow s_2 \rightarrow s_5 \rightarrow s_9$ in Figure 4, will not be represented because it is not a maximum trace. Hence, there are only 10 maximum traces in this CPS.

From an algorithmic point of view, the configurations having the maximum number of terminated intention plans could be straightforwardly detected by parsing the CPS structure, with regards to the set of terminated intention plans of each built configuration. By labeling these configurations with a specific proposition `MAX`, the search of maximum traces is reduced to the traces which satisfy the CTL (Computational Tree Logic) property `AF(MAX)`.

The consistency of a set of intentions I can also be checked over some trace σ of the CPS, in particular in two extreme cases:

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

Table 4 Semantic rules of intention and agent configurations

Intention plan level		
(Action)	$\frac{E \xrightarrow{a} E' \quad a \in Act}{(E,\widehat{P}) \xrightarrow{a} (E',\widehat{P})}$	$\frac{E \xrightarrow{\delta} E' \quad E = exit}{(E,\widehat{P}) \xrightarrow[\widehat{P}]{\tau} (E',\widehat{P})}$

Agent plan level		
(Action)	$\frac{ps \xrightarrow{a} ps' \quad a \in Act}{(ps,\ell,T) \xrightarrow{a} (ps',\ell,T)}$	$\frac{ps \xrightarrow[\widehat{P}]{\tau} ps'}{(ps,\ell,T) \xrightarrow{\tau} (ps',\ell,T \cup \{\widehat{P}\})}$
(Communication)	$\frac{ps \xrightarrow{x!(v)} ps' \quad x \in \Lambda}{(ps,\ell,T) \xrightarrow{x!(v)} (ps',\ell,T)}$	$\frac{ps \xrightarrow{x?(v)} ps' \quad x \in \Lambda}{(ps,\ell,T) \xrightarrow{x?(v)} (ps',\ell,T)}$
(Mobility)	$\frac{ps \xrightarrow{move(\ell')} ps' \quad \ell \neq \ell'}{(ps,\ell,T) \xrightarrow{move(\ell')} (ps',\ell',T)}$	$\frac{ps \xrightarrow{move(\ell)} ps'}{(ps,\ell,T) \xrightarrow{\tau} (ps',\ell,T)}$
(Sequence)	$\frac{ps_1 \xrightarrow{a} ps'_1 \quad a \in Act}{ps_1 \gg ps_2 \xrightarrow{a} ps'_1 \gg ps_2}$	$\frac{ps_1 \xrightarrow[\widehat{P}]{\tau} ps'_1}{ps_1 \gg ps_2 \xrightarrow[\widehat{P}]{\tau} ps'_1 \gg ps_2}$
(Parallel)	$\frac{ps_1 \xrightarrow{a} ps'_1 \quad a \in Act}{ps_1 \mid\mid\mid ps_2 \xrightarrow{a} ps'_1 \mid\mid\mid ps_2}$ $\frac{ps_1 \xrightarrow{a} ps'_1 \quad a \in Act}{ps_2 \mid\mid\mid ps_1 \xrightarrow{a} ps_2 \mid\mid\mid ps'_1}$	$\frac{ps_1 \xrightarrow[\widehat{P}]{\tau} ps'_1}{ps_1 \mid\mid\mid ps_2 \xrightarrow[\widehat{P}]{\tau} ps'_1 \mid\mid\mid ps_2}$ $\frac{ps_1 \xrightarrow[\widehat{P}]{\tau} ps'_1}{ps_2 \mid\mid\mid ps_1 \xrightarrow[\widehat{P}]{\tau} ps_2 \mid\mid\mid ps'_1}$

Application to the scenario

We reconsider the scenario of Section 3 to achieve the intentions of Bob in a concurrent way: $\overline{[P_B]} = ((E_g,\widehat{P_g}) \mid\mid\mid (E_m,\widehat{P_m}))$ is the agent plan configuration considered for Bob.
The pairs $(E_m,\widehat{P_m})$ and $(E_g,\widehat{P_g})$ are two intention plan configurations of Bob ; The first one corresponds to the intention $meeting(Alice,\ell_1)$ and the second to $getting\_copies(\ell_2)$, such that
$E_m = move(\ell_1);meet(Alice);exit$ and
$E_g = get\_copies(\ell_2);Alice!(confirm\_getc);exit$.

The Contextual Planning System of Bob, denoted CPS_B, is illustrated in Figure 4. It is built from the initial CPS state, $s_0 = (\overline{[P_B]},\ell_2,\emptyset)$, taking into account the current location ℓ_2 of Bob. In the figure, the dashed edges represent the unrealized transitions from the states $s \in \{s_2,s_5,s_8\}$, because $pre(getc) = \ell_2 \notin \mathcal{L}(s)$.

An example of maximum trace derived from s_0 is the following, expressing that Bob got the copies before moving to the meeting with Alice:
$((E_g,\widehat{P_g}) \mid\mid\mid (E_m,\widehat{P_m})) \xrightarrow{getc} ((E'_g,\widehat{P_g}) \mid\mid\mid (E_m,\widehat{P_m}),\ell_2,\emptyset) \xrightarrow{confirm}$
$((E''_g,\widehat{P_g}) \mid\mid\mid (E_m,\widehat{P_m}),\ell_2,\emptyset) \xrightarrow[\widehat{P_g}]{\tau} ((E_m,\widehat{P_m}),\ell_2,\{\widehat{P_g}\}) \xrightarrow{move(\ell_1)}$
$((E'_m,\widehat{P_m}),\ell_1,\{\widehat{P_g}\}) \xrightarrow{meet} ((E''_m,\widehat{P_m}),\ell_1,\{\widehat{P_g}\}) \xrightarrow[\widehat{P_m}]{\tau}$
$((stop,\widehat{P_m}),\ell_1,\{\widehat{P_g},\widehat{P_m}\})$

6 Planning Guidance from Past-Experiences

Based on the CPS structure, we augment the planning process with a rich learning mechanism dedicated to improve selection of elementary plans in the LibP library and to guide the agent to achieve as many concurrent intentions as possible.

In the CPS structure, the different traces capture both the concurrency of intentions and the choice of an alternative for each of these intentions. These are implicitly expressed through the interleaving of actions that composed the selected elementary plans. In this section, we focus on actions and their performances in order to reinforce the quality of maximum traces that are offered. The agent can keep track of past-experiences obtained when running actions, and can learn the gain of re-using some action in a given context. Instead of the simplest way which consists of considering the actions of each CPS state separately, we propose to evaluate an overall gain for each maximum traces of the CPS, so as to order them.

Figure 5 shows different FIFO queues recording the successes '1' and failures '-1', with respect to some action a. Such structure is generalized to all possible actions and is called the *Learned Contextual Experiences structure* of the planning process (*LCE* for short). More precisely, if a cer-

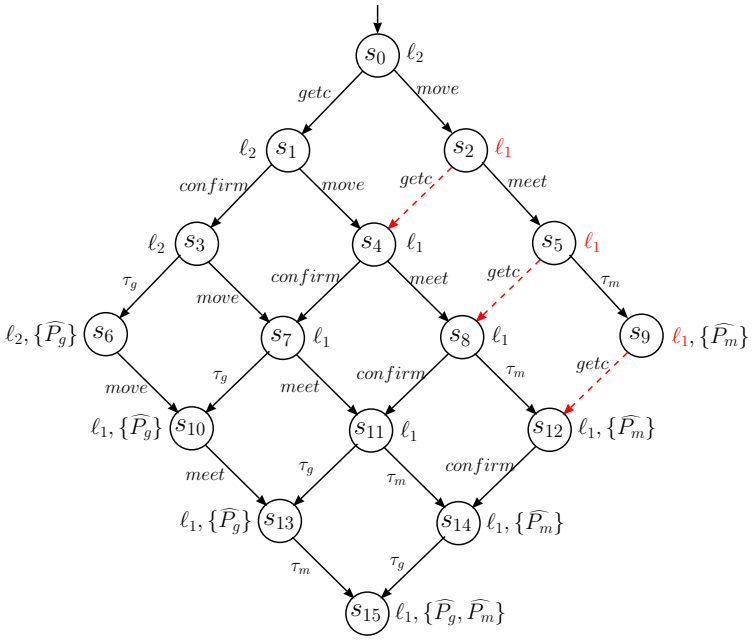

Fig. 4 The CPS_B corresponding to the plan $\overline{P_B}$

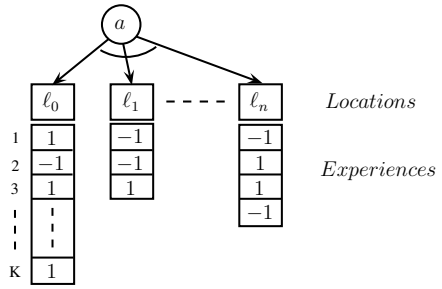

Fig. 5 Learned Contextual Experiences of an action a

tain outcome $o \in \{1,-1\}$ is obtained for the performance of an action a in some location ℓ, it is pushed in the queue $LCE_a(\ell)$.

Due to the high dynamicity of AmI systems, we accord with a common heuristics such that the more recent an execution outcome is the more pertinent it is. Each queue for an action is parametrized by a pair (f,K) where f is a function called *pertinence function* and K is the maximum size of this queue.

- In order to tackle the data explosion problem when analyzing its past-experiences, the queue is bounded by K experiences and assumed to be finite, hence the pertinence function domain. This implies that an outcome value beyond a maximum position are not pertinent, with respect to the values in the queue.
- The pertinence function $f : 1..K \rightarrow \mathbb{R}$ associates a weight for each outcome value stored in a queue. As an interesting case, for any index x of the queue, $f(x) = \frac{1}{x}$ yields much more importance for any outcome value corresponding to x.

The *contextual gain value* $G_a(\ell) \in [-1,1]$ of an action a in some location ℓ directly relates to the successes and failures during the past-experiences of a in ℓ. From a non empty queue $LCE_a(\ell)$, $G_a(\ell)$ is computed by applying the pertinence function on each outcome value, knowing that the size of the queue is $0 < k \leq K$:

$$G_a(\ell) = \frac{\sum_{j=1}^{k}(o_j * f(j))}{\sum_{j=1}^{k} f(j)} \quad (1)$$

In the case where $LCE_a(\ell) = \emptyset$, $G_a(\ell)$ is set to '0' corresponding to a middle gain value. This allows characterizing, at the agent level, that the action a in ℓ is not already explored. Hence, the non-explored actions can be privileged against the exploited ones having a gain less 0.

By extension, the different $G_a(\ell)$ values allow us to assign a gain $G(tr)$ to each CPS transition $tr = (ps,\ell,T) \xrightarrow{a} (ps',\ell',T') \in Tr$. The quality of each (maximum) trace is computed as follows, in order to guide the agent with the trace of the best quality.

$$Q(\sigma) = \frac{\sum_{tr}^{\sigma} G(tr)}{|\sigma|} \quad (2)$$

Application to the scenario

The Learned CPS_B of Bob is highlighted in Figure 6. The dotted transitions do not support any maximum trace ($s_0 \rightarrow$

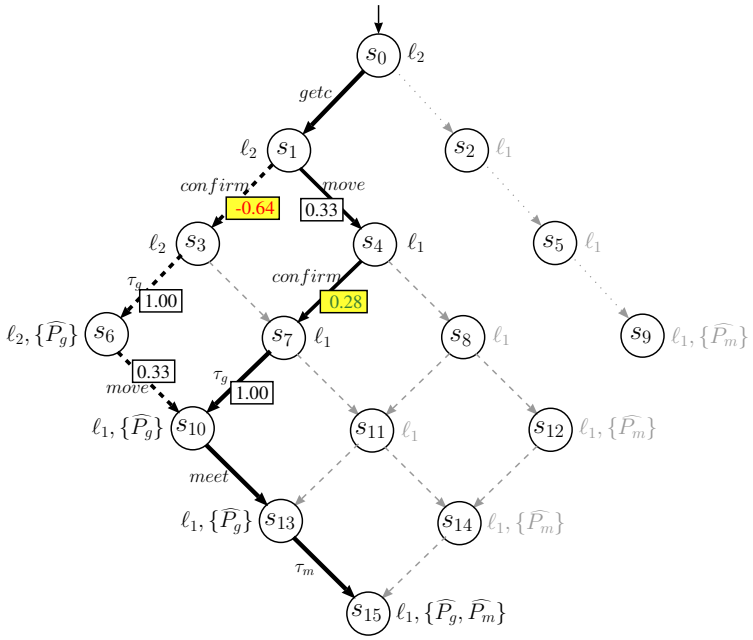

Fig. 6 The Learned CPS_B corresponding to the plan $\overline{P_B}$

$s_2 \rightarrow s_5 \rightarrow s_9$), then are neglected during the guidance phase. For the other transitions, their gains are computed. For instance, the values $G(s_1 \xrightarrow{confirm} s_3) = -0.64$ and $G(s_4 \xrightarrow{confirm} s_7) = 0.28$ are obtained by considering, the LCE of Bob sampled in Table 5, knowing that the action of both transitions is $Alice!(confirm\_getc)$ ($confirm$ for short) and the fact that the respective source locations of these transitions are $\mathcal{L}(s_1) = \ell_2$ and $\mathcal{L}(s_4) = \ell_1$. In Figure 6, two maximum traces are highlighted by a bold line, such that the intermediary parts distinguish them. By computing the respective quality of these traces, the bold one appears to be more relevant that the mixed bold/dashed one, and can be offered to the execution process as the optimal trace (σ_{opt}).

Table 5 LCE samples for Bob

confirm	1	2	3	4	...	*K*
ℓ_1	1	−1	1	−1		
ℓ_2	−1	−1	1			

move(ℓ_1)						
ℓ_2	1	−1				

$exit_g$						
ℓ_1	1	1	1			
ℓ_2	1	1	1			

The given guidance is driven by Algorithm 1, yielding an *optimal maximum trace*, σ_{opt}, from a set I of intentions. As the planning process acts as a service provider, it must also notify the case where there is no possible maximum trace. Consequently, this is useful to drive the mental process to revise the agent intentions according to the current context.

Algorithm 1 Contextual guidance process

Require:
I: set of weighted intentions;
LCE: Learned Contextual Experiences module;

Build $\overline{P}$ from I;
Construct the *CPS* from $\overline{P}$;
Extract Σ_{MAX} from the *CPS*;
if $\Sigma_{MAX} \neq \emptyset$ **then**
    Enrich the *CPS-L* from the *CPS* and *LCE*;
    Order Σ_{MAX} from the *CPS-L*;
    Return σ_{opt} among the ones of Σ_{MAX};
else
    Return $\emptyset$;
end if

7 Complexity of the CPS approach

The hardness of the CPS-L computation mainly comes from the building of the CPS, since the computation of performance information only require to parse the LCE structure linearly. To evaluate a worst case complexity of the CPS-L, our approach consists in studying the concurrent complexity coming from the agent plan structure. To this end, we propose to reduce the operators of an agent plan configuration $[\overline{P}]$ in order to only focus on the parallel and the sequential compositions, as follows:

- The parallel composition ($E_1 \,|||\, E_2$) is more complex than the interruption one ($E_1 \,[> E_2$) and the concurrent ones ($E_1 \,||\, E_2$ and $E_1 \,|[L]|\, E_2$),
- The sequential composition ($E_1 \gg E_2$) is more complex than the non-deterministic choice one ($E_1 \,[\,]\, E_2$).

We deduce that the worst case complexity of $[\overline{P}]$ can be bounded by the complexity of an expression formed of a parallel composition of sequential components. As far as complexity is concerned, this is analogous to a composition of words, hence a number of traces in the CPS of order:

$$O\left(\frac{(n_1+n_2+...+n_k)!}{n_1!*n_2!*...*n_k!}\right) \quad (3)$$

taking into account that $[\overline{P}]$ can be transformed in a parallel composition of k sequential components (sub-traces) such that n_i is the length of the i^{th} sub-trace.

From this result, it is then not surprising that a certain form of combinatorial explosion could appear in the CPS-L, as the number and the length of the (assimilated) parallel components increase. This is in fact a standard result of concurrent systems, however, here limited by the fact that only one agent is concerned and the fact that the real size of the CPS-L decreases by the number of actions whose preconditions cannot be contextually satisfied. Moreover, we consider that the number of intentions to be dealt with at a moment is reasonably small and that the lengths of the intention plans are small. These reasonable criteria tend to constrain the hardness of an agent plan configuration and the corresponding CPS-L.

8 Discussion and Related Work

We have presented in this paper, a contextual model for ambient agents which provides a mechanism to guide them contextually. In this section, we discuss the different points of view of the proposed approach, about both planning and learning techniques taking into account the guidance mechanism.

Planning

Considering MAS as a distributed systems composed of 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 a well defined semantics. Many BDI agent-centric approaches have been proposed to cope with the dynamic of the agent's environment. In (**?**), 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, **?** showed how to specify and test learning approaches in some particular cases.

Our model is also agent-centric and accords with the principle of tightly controlling plans from the BDI mental attitudes. In contrast with the former works, the achievement 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 has already been studied in the literature in order to restrain the agent activity. In particular a GraphPlan planner can efficiently produce a global plan as a flow of actions that corresponds to the subset of the desires (i.e. goals) that could be executed concurrently (**?**). However, GraphPlan only deals with some of the possible scheduling between actions, since it follows a global time step approach. In contrast, our approach takes all the possible cases into account.

BDI languages

In the literature there are a number of BDI agent programming languages (**?**), highlighting different aspects or modules developed in agent software like goals, planning and organization, e.g. Jadex (**?**), Jason (**?**), JIAC (**?**) and 2APL (**?**). Since 2006, BDI agent-centric approaches emerge to cope with the dynamicity of the agent environment. In particular, the work of **?** has extended some existing formal specification models dedicated to distributed systems, in order to specify actions in plans while integrating BDI ingredients within plans in a unified way. Nevertheless, our formal algebraic language based on LOTOS, appears to be more expressive in its capacity to represent plans as concurrent processes as well as concurrent actions. It is also possible to handle action and plan failures in the AgLOTOS language and the HoA architecture. Moreover 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.

Validation

The verification task standardly applied to MAS is mainly driven by a global vision of the system, e.g. in (**?**). In (**?**), the reuse of some program checking techniques is 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 consider introducing similar techniques within AmI agents, however, the high-level dynamics usually effective in an AmI environment could introduce too much states to consider, even after reduction. In our paper, the proposed guidance technique, is agent-centric so can be seen as a reduction of the system combinatorial explosion problem to the contexts handled by the agent.

Learning

Learning is the general approach to improve the behavior of intelligent agents from experimental-based information. It has been involved at different levels of the agents. Learning was first investigated at the mental level of BDI agents either to improve the BDI deliberation from some learned knowledge e.g. (**?**) or to produce a new plan with respect to some objective e.g. (**?**). Learning was also used to reinforce the selection of plans among different possible alternatives. In particular, a decision tree was introduced by **?** to represent the different contexts in which the agent behaves. Indeed, the behavior of the agent is learned from the successes and failures of the executions of the agent's plans. The idea to take profit from the past execution experiences was adapted in (**?**) bringing out an on-line technique, based on a hierarchical goal-plan structure, in order to make the selection of some alternative according to the failures of the previous ones.

In this paper, we follow a similar idea, but unlike the former proposals, our approach is not only online but also adaptive: it is driven by the maximization of intention satisfactions, in order to guide the agent through the different paths implied by the concurrency of actions.

The learned CPS can be viewed as a reinforcement learning for the selection of elementary plans but based on the successes and failures of the executed actions. Actually, the selection of a path implies the selection of some alternative for each intention plan. Moreover, as we aim to tackle ambient systems with limited experiences, our relevance function favors a detailed representation of the action past-experiences, instead of an abstract overall view counting successes and failures, generally based on huge number of repetitive actions. The queues recording the past-experiences are bounded in order to tackle the combinatorial complexity introduced by their management. This approach is compatible with the idea of forgetting useless history. Regarding the pertinence function, our technique is in line with earlier propositions like the Q-learning algorithm. The last one learns a quality value for each action taking into account that the known values become deprecated as the time progresses, under a function like $\varepsilon = 1/t$. In our context, the applied forgetting function $(1/x)$ allows us to privilege the most recent past-experiences according to a logical recording time.

Recently, the work of (**?**) introduces a reinforcement learning technique dedicated to a domotic ambient system. The proposed platform is able to learn the acts of a human by recording a set of perceptions for each act. It can react instead of the human to the change of perceptions. The fact that this works under approximation, yields some similarities with our work which takes failures into account, however, without considering any plan construction or mental references.

As a learning alternative, other works proposed probabilistic systems in the frame of repetitive scenarios. A Markov Decision Process approach is proposed in (**?**) to reinforce learning in BDI agents, whereas a probabilistic function is proposed in (**?**) to introduce a certain capacity to explore new plans, against the use of plans experimentally known as successful. Both approaches are based on a huge number of repetitive experiences, so they could be ineffective in the case of ambient systems where agents must behave in real time, within a dynamic environment. Moreover, as we focus on user assistance, we can claim that each plan is certainly executed only a few times. Our alternative which consists of a history structure for each action seems relevant in the case of ambient systems, since the forgetting function applied to the known experiments can capture the fact that the changes in the agent environment can deprecate some past experiences on some action or plan.

9 Conclusion

The algebraic language AgLOTOS appears to be a powerful way to express an AmI agent plan as a set of concurrent processes, helped by an adapted plan library describing elementary plans. Based on any current set of intentions, the semantics of AgLOTOS allows to build a Contextual Planning System (CPS), from which all the maximum traces of situated actions can be evaluated, each one representing an execution plan maximizing the satisfaction of the set of intentions.

The main contribution of this paper was to improve the guidance mechanism based on the CPS in order to minimize the risk of a plan failure. To do that, the CPS structure is enriched with learning information extracted from the past-experiences of the executions of actions. This is used to qualify each action contextually, hence to qualify each maximum trace.

We demonstrate in this paper that the complexity of our planning approach mainly comes from the interleaving of the situated actions in the CPS. As discussed in the paper, the combinatorial effect should be reasonably handled, however, we now aim at improving the CPS guidance by introducing partial search techniques. Moreover, observing that the proposed CPS only focuses on spatial information, we aims at

enriching the learning techniques with temporal aspects, to propose a spatio-temporal guidance.

References