From Intentions to Plans: A Contextual Planning Guidance

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

From Intentions to Plans: A Contextual Planning Guidance

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

Abstract The proposal AgLOTOS algebraic language is dedicated to the specification of agent plans in ambient systems (AmI). From its two level specification, plans can be built automatically as a system of concurrent processes. In this context, we show how to achieve a powerful mechanism for a contextual guidance based on a specific and formal construction called Contextual Planning System (CPS). The CPS structure is used to propose an optimal plan preserving the consistency of the intentions.

1 Introduction

Ambient Intelligence (AmI) is the vision of ubiquitous electronic environment that is non-intrusive and proactive, when assisting people during various activities [7, 5]. For the design of such complex systems, MAS approaches offer interesting frameworks, since their agents are considered as intelligent, proactive and autonomous [4].

Ahmed-Chawki Chaouche
LIP6 Laboratory, University Pierre and Marie Curie, 4 Place Jussieu, 75005 Paris, France, e-mail: ahmed.chaouche@lip6.fr (in cotutelle with MISC Laboratory, University Constantine 2)

Amal El Fallah Seghrouchni
LIP6 Laboratory, University Pierre and Marie Curie, 4 Place Jussieu, 75005 Paris, France, e-mail: amal.elfallah@lip6.fr

Jean-Michel Ilié
LIP6 Laboratory, University Pierre and Marie Curie, 4 Place Jussieu, 75005 Paris, France, e-mail: jean-michel.ilie@upmc.fr

Djamel Eddine Saïdouni
MISC Laboratory, University Constantine 2, Ali Mendjeli Campus, 25000 Constantine, Algeria, e-mail: saidouni@misc-umc.org

The major problem for AmI agent consists in recognizing its environmental contexts, including its locality and the discovery of other agents. In [2], it is shown how autonomous BDI agents [8] can evolve and move within an ambient environment, based on an agent centric approach and a context-awareness. The proposed HoA model takes into account the major features and functionalities of AmI, in particular dynamic requirements: AmI systems can be open; agents can enter or leave the system.

This paper introduces an efficient planning management process into the architecture of the agent. In particular, we aim at offering to each AmI agent, a powerful predictive service, that can run on the fly. Like in other recent approaches, e.g. [6, 9], which are dedicated to the planning and the validation of BDI MAS systems, 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.

We take profit from the fact that the plan of the agent can be derived from the current set of intentions, which result from the reasoning of the BDI interpreter. Our approach is based on a formal description language recently proposed for plans, namely *AgLOTOS* [3]. This allows us to introduce modularity and concurrency aspects to compose sub-plans. Unlike the formal description of [9], the *AgLOTOS* semantics overpasses the sequential execution of sub-plans. Rather, the concurrency of sub-plans is fully implemented, actually only being restrained to solve the possible inconsistencies of intentions.

In this paper, the semantics of *AgLOTOS* is enriched to automatically produce a *Contextual Planning System* (*CPS*), which allows to automatically guide the execution of plans. In contrast to [6], which restrains the execution to one possible subset of consistent intentions, the CPS is a state transition structure which captures whatever execution in respect to the predicted evolution of the context.

The paper is organized as follows: In Section 2, the *AgLOTOS* specification language is briefly described and used to associate plans with intentions (e.g. composition of plans). In Section 3, the *AgLOTOS* operational semantics is enriched to automatically produce the Contextual Planning System (*CPS*). A realistic scenario is given as an illustration of our guidance mechanism. The last section concludes and outlines our perspectives.

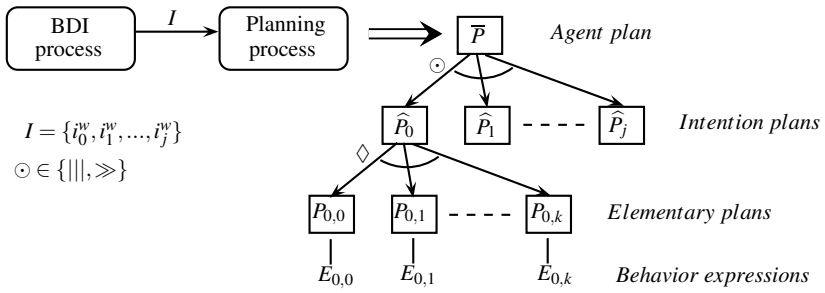

Fig. 1 Agent cycle

2 AgLOTOS Specification Language

The behavior of the BDI agent we consider in this paper is carried by two successive processes, as highlighted in Figure 1. As usual, the *BDI process* represents the reasoning mechanism, based on the beliefs (B), desires (D), and intentions (I) structures, the instances of which defines the BDI states of the agent. Triggered by the perceived events, the BDI process manages/updates the B,D and I structures. In order to organize its selected intentions, the BDI process is able to schedule them by associating each one a given weight (see Section 2.2).

From the set of intentions, the *Planning process* is called by the BDI process to produce a plan of actions, helped by a library of plans (*LibP*). In our approach for each BDI state, the plan of the agent, namely the *Agent plan* is composed in two levels: *(1)* The agent plan is made of sub-plans called *Intentions plans*, each one dedicated to achieve the associated 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 2.1). Further, we assume that the *LibP* library is indexed by the set of all the possible intentions for the agent.

2.1 The Syntax of Elementary Plans

Elementary plans are written using the algebraic language *AgLOTOS* [2]. This language extends the LOTOS language [1] in order to deal with the concurrency of actions in plans.

Let $\mathscr{O}$ 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 $\mathscr{O}$. Let $\mathscr{H} \subset \mathscr{O}$ 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 other agents in the AmI system, *(2)* it can move between the AmI system localities and *(3)* an agent can communicate with another agent in the system.
- An *AgLOTOS* expression refers to contextual information with respect to the (current) BDI state of the agent: *(1)* Θ is a finite set of space localities, *(2)* Λ is a set of agents with which it is possible to communicate, and $\mathscr{M}$ 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 locality ℓ ($\ell \in \Theta$). The syntax of the communication primitives is inspired from the semantics of the π-calculus primitives, however considered within a totally dynamic communication support, hence without specification of predefined channels: the expression $x!(\nu)$ specifies the emission to the agent x ($x \in \Lambda$) of some message ν ($\nu \in \mathscr{M}$), whereas, the expression $x?(\nu)$ means that ν is received from some agent x.

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

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

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

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. The LOTOS parallel composition, denoted $E\,|[L]|\,E$, can model both synchronous composition, $E\,||\,E$ if $L = \mathscr{O}$, 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.

2.2 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, in respect with the intentions of the agent and the associated weights.
- the *alternate composition* operator, denoted $\Diamond$, allows to specify an alternation of elementary plans. In particular, an intention is satisfied iff at least one of the associated elementary plans is successfully terminated.

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

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

With respect to the set 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*:

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

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

A Simple AmI Example. Let us consider the *AmI Universitary scenario* presented in [2] 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 with Bob in the locality ℓ_1, and *(2)* to get her exam copies from the locality ℓ_2. Clearly, the Alice's desires are conflictual since Alice cannot be in two distinct localities simultaneously.

Alice's scenario
$I_A = \{meeting(Bob, \ell_1), asking(Bob, get\_copies(\ell_2))\}$
$\overline{P_A} = meet(Bob); exit \ ||| \ Bob!(get\_copies(\ell_2)); exit$

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

The scenarii of Alice and Bob are specified separately, assuming that Bob and Alice may coordinate in order to achieve their intentions, at their BDI process levels. The actions in plans are simply expressed by 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 BDI 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 $weight(meeting(Alice, \ell_1)) < weight(getting\_copies(\ell_2))$. In the intention set I_B, the corresponding agent plan expression of Bob is: $\overline{P_B} = get\_copies(\ell_2); exit \gg$

$move(\ell_1);meet(Alice);exit$, which is built by using the *options* and *plan* mappings. Pay attention that some actions can be processed concurrently, so is the case in the agent plan $\overline{P_B}$, for the two intention plans $get\_copies(\ell_2);exit$ and $move(\ell_1);meet(Alice);exit$.

3 Contextual Planning Management

3.1 Semantics of AgLOTOS

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. Basic LOTOS semantics is detailed in [2] which formalizes how a process can evolve under the execution of actions. In particular, the rule $\frac{P:=E \quad E \xrightarrow{a} E'}{P \xrightarrow{a} E'}$, specifies how an (E,P) pair 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.

The next definition specifies how the expression of an *agent plan* is formed compositionally from the expressions of the *intentions plans* of the agent, themselves built from an alternate of *elementary plans* and their behavior expressions. With respect to some agent plan $\overline{P}$, we introduce a notion of configuration of plans in order to specify that a part of the plan can already be executed. Further, the notation $[\overline{P}]$ represents the configuration of the agent plan $\overline{P}$, it is an *AgLOTOS* expression, which is obtained by composition of the different intention plan configurations of the agent, like $(E,\widehat{P})$.

Definition 1. Any 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. $\dfrac{\overline{P}::=\overline{P_1} \odot \overline{P_2} \qquad \odot\in\{|||,\gg\}}{[\overline{P}]::=[\overline{P_1}] \odot [\overline{P_2}]}$

The planning state of the agent is now defined contextually, taking into account the agent locality and a termination information about the different intention plans defined for the agent.

Definition 2. A (contextual) planning state is a tuple $(\mathscr{C},\ell,T)$, where $\mathscr{C}$ is any plan configuration $[\overline{P}]$, ℓ corresponds to an expected locality for the agent, and T is the subset of intention plans which are terminated.

Table 1 shows the operational semantic rules defining the possible planning state changes for the agent. These rules are applied to produce a Contextual Planning

Table 1 Semantic rules of intention and agent configurations

Intention plan level		
(Action)	$\dfrac{E \xrightarrow{a} E' \quad a \in \mathscr{O} \cup \{\tau\}}{(E,\widehat{P}) \xrightarrow{a} (E',\widehat{P})}$	$\dfrac{E \xrightarrow{\delta} E'}{(E,\widehat{P}) \xrightarrow[\widehat{P}]{\tau} (E',\widehat{P})}$

Agent plan level		
(Action)	$\dfrac{\mathscr{C} \xrightarrow{a} \mathscr{C}' \quad a \in \mathscr{O} \cup \{\tau\}}{(\mathscr{C},\ell,T) \xrightarrow{a} (\mathscr{C}',\ell,T)}$	$\dfrac{\mathscr{C} \xrightarrow[\widehat{P}]{\tau} \mathscr{C}'}{(\mathscr{C},\ell,T) \xrightarrow{\tau} (\mathscr{C}',\ell,T \cup \{\widehat{P}\})}$
(Communication)	$\dfrac{\mathscr{C} \xrightarrow{x!(v)} \mathscr{C}' \quad x \in \Lambda}{(\mathscr{C},\ell,T) \xrightarrow{x!(v)} (\mathscr{C}',\ell,T)}$	$\dfrac{\mathscr{C} \xrightarrow{x?(v)} \mathscr{C}' \quad x \in \Lambda}{(\mathscr{C},\ell,T) \xrightarrow{x?(v)} (\mathscr{C}',\ell,T)}$
(Mobility)	$\dfrac{\mathscr{C} \xrightarrow{move(\ell')} \mathscr{C}' \quad \ell \neq \ell'}{(\mathscr{C},\ell,T) \xrightarrow{move(\ell')} (\mathscr{C}',\ell',T)}$	$\dfrac{\mathscr{C} \xrightarrow{move(\ell)} \mathscr{C}'}{(\mathscr{C},\ell,T) \xrightarrow{\tau} (\mathscr{C}',\ell,T)}$
(Sequence)	$\dfrac{\mathscr{C}_1 \xrightarrow{a} \mathscr{C}_1' \quad a \in \mathscr{O} \cup \{\tau\}}{\mathscr{C}_1 \gg \mathscr{C}_2 \xrightarrow{a} \mathscr{C}_1' \gg \mathscr{C}_2}$	$\dfrac{\mathscr{C}_1 \xrightarrow[\widehat{P}]{\tau} \mathscr{C}_1'}{\mathscr{C}_1 \gg \mathscr{C}_2 \xrightarrow[\widehat{P}]{\tau} \mathscr{C}_1' \gg \mathscr{C}_2}$
(Parallel)	$\dfrac{\mathscr{C}_1 \xrightarrow{a} \mathscr{C}_1' \quad a \in \mathscr{O} \cup \{\tau\}}{\mathscr{C}_1 \mid\mid\mid \mathscr{C}_2 \xrightarrow{a} \mathscr{C}_1' \mid\mid\mid \mathscr{C}_2}$	$\dfrac{\mathscr{C}_1 \xrightarrow[\widehat{P}]{\tau} \mathscr{C}_1'}{\mathscr{C}_1 \mid\mid\mid \mathscr{C}_2 \xrightarrow[\widehat{P}]{\tau} \mathscr{C}_1' \mid\mid\mid \mathscr{C}_2}$
	$\dfrac{\mathscr{C}_1 \xrightarrow{a} \mathscr{C}_1' \quad a \in \mathscr{O} \cup \{\tau\}}{\mathscr{C}_2 \mid\mid\mid \mathscr{C}_1 \xrightarrow{a} \mathscr{C}_2 \mid\mid\mid \mathscr{C}_1'}$	$\dfrac{\mathscr{C}_1 \xrightarrow[\widehat{P}]{\tau} \mathscr{C}_1'}{\mathscr{C}_2 \mid\mid\mid \mathscr{C}_1 \xrightarrow[\widehat{P}]{\tau} \mathscr{C}_2 \mid\mid\mid \mathscr{C}_1'}$

transition System, called *CPS*, from an initial planning state, e.g. $([\overline{P}],\ell,\emptyset)$, meaning that the agent is initially at locality ℓ, and its plan configuration is $[\overline{P}]$. There are two kinds of transition rules:

Intention plan level: When an intention plan is assumed to be treated, the left hand side transition $(\mathscr{C}_1,a,\widehat{P},\mathscr{C}_2)$, denoted $\mathscr{C}_1 \xrightarrow[\widehat{P}]{a} \mathscr{C}_2$, expresses a change of intention configuration, from $\mathscr{C}_1$ to $\mathscr{C}_2$, and assumes the execution of the action a from $E \xrightarrow{a} E'$ and $P := E$. The right hand side transition highlights the termination case, keeping trace of the intention plan $\widehat{P}$ that is going to be terminated. By calling $\mathscr{CN}$ the set of all the possible intention plan configurations for the agent, the transition relation is a subset of $\mathscr{CN} \times Act \times \widehat{\mathscr{P}} \times \mathscr{CN}$. For sake of clarity, the transition $(\mathscr{C}_1,a,nil,\mathscr{C}_2)$ is simply denoted $\mathscr{C}_1 \xrightarrow{a} \mathscr{C}_2$. Observe that due to the fact we consider a predictive guidance in this paper, 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 take into account every elementary plan in order to achieve the corresponding intention.

Agent plan level: the possible changes of the planning states, like $(\mathscr{C}, \ell, T)$, are expressed at this level. In the Communication rules, the action send $x!(v)$ (resp. receive $x?(v)$) is constrained by the visibility of the agent x in its neighborhood. In the Mobility rule, the effect of the $move(\ell')$ action yields the agent to be placed in ℓ'. The Action rules refer to the ones of the intention plan level. The left hand side one exhibits the case of a regular action, whereas the right hand side one specifies the termination case of some intention plan, which is added to T.

3.2 Planning Guidance

From any set of intentions in the agent, denoted I, a Contextual Planning System is built, by using the rules of Table 1 and taking into account contextual information of three kinds: *(1)* the reached locality in a planning state, *(2)* the set 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 Contextual Planning System, denoted *CPS*, is a labeled kripke structure $\langle S, s_0, Tr, \mathscr{L}, \mathscr{T} \rangle$ where:

- S is the set of planning states,
- $s_0 = ([\overline{P}], \ell, \emptyset) \in S$ is the initial planning state of the agent, such that $[\overline{P}]$=$plan(I)$ and ℓ represents the current locality 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 $a \in \mathscr{O} \cup \{\tau\}$,
- $\mathscr{L} : S \to \Theta$ is the locality labeling function
- $\mathscr{T} : S \to 2^{\widehat{\mathscr{P}}}$ is the termination labeling function which captures the terminated intention plans.

In a *CPS*, the transitions from any state s only represent actions that are realizable. Like in STRIPS description language [6], actions to be executed are modeled by instantiated predicates submitted to preconditions and effects. In this paper, the preconditions only concern the contextual information known in that state. Let $pre(a)$ be the precondition of any action a, then $pre(x!(v)) = pre(x?(v)) = (x \in \Lambda)$ and for any other action a, $pre(a(\ell)) = \ell \in \mathscr{L}(s)$.

In order to guide the agent, the planning process can select an execution trace through the *CPS* which maximizes the number of intention terminations, with respect to the mapping $\mathscr{T}$ in *CPS* states. This can be captured with the notion of Maximum trace, based on a trace mapping $end : \Sigma \longrightarrow 2^{\widehat{\mathscr{P}}}$ used to specify the set $end(\sigma)$ of the termination actions that occur in a trace $\sigma \in \Sigma$. From an algorithmical 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 satisfies the (LTL) temporal logic property `AF(MAX)`.

The consistency of a set I of agent intentions can also be checked, in particular in two extreme cases:

- if $|end(\sigma)| = |I|$, means 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 unappropriated with respect to the set of agent intentions.

Application to the scenario. We reconsider the scenario of Section 2 to achieve the intentions of Bob in a parallel way: $[\overline{P_B}] = ((E_g, \widehat{P_g})|||(E_m, \widehat{P_m}))$ 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 expression 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 $E_g = get\_copies(\ell_2); exit$.

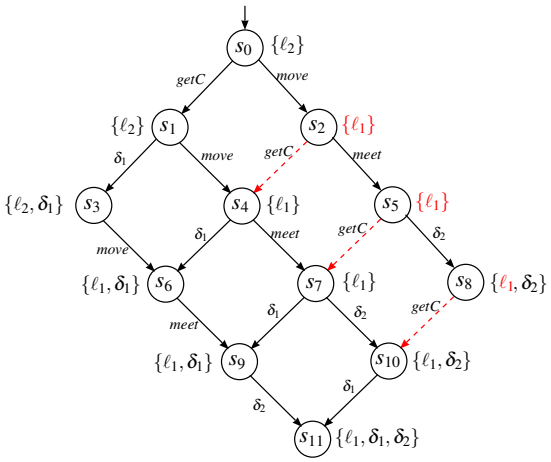

Fig. 2 The CPS_B corresponding to the plan $\overline{P_B}$

The Contextual Planning System of Bob, denoted CPS_B, is illustrated in Figure 2. It is built from the initial CPS state, $s_0 = ([\overline{P_B}], \ell_2, \emptyset)$, taking into account the current locality 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 $pre(getC) = \ell_2 \notin \mathscr{L}(s)$.

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:

$$((E_g, \widehat{P_g})|||(E_m, \widehat{P_m})) \xrightarrow{getc} ((E'_g, \widehat{P_g})|||(E_m, \widehat{P_m}), \ell_2, \emptyset) \xrightarrow[\widehat{P_g}]{\tau} ((E_m, \widehat{P_m}), \ell_2, \{P_g\}) \xrightarrow{move(\ell_1)}$$

$$((E'_m, \widehat{P_m}), \ell_1, \{P_g\}) \xrightarrow{meet} ((E''_m, \widehat{P_m}), \ell_1, \{P_g\}) \xrightarrow[\widehat{P_m}]{\tau} ((stop, \widehat{P_m}), \ell_1, \{P_g, P_m\})$$

4 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. The main contribution of this paper is an enriched semantics of AgLOTOS: it allows to build a Contextual Planning System (CPS), for any BDI state of the agent. From the current set of intentions of the agent, all the possible plan evolutions can be evaluated through the CPS. This allows one to define an original predictive mechanism to contextually guide the agent in its future executions. In particular, we demonstrate how to realize the concurrent executions of the agent plans, while optimizing the number of satisfied intentions. Observe that the proposed techniques are also suitable for some class of partially ordered set of instances. Among the possible perspectives, we aim at combining the CPS approach with learning techniques like in [10], since this also can be viewed as a guidance mechanism but based on past-experiences.

References

1. Brinksma, E. (ed.): ISO 8807, LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour (1988)
2. 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**, 156 – 163 (2013)
3. Chaouche, A.C., El Fallah Seghrouchni, A., Ilié, J.M., Saïdouni, D.E.: A dynamical plan revisiong for ambient systems. Procedia Computer Science **32**, 37 – 44 (2014)
4. Doyle, J.: Rationality and its roles in reasoning. Computational Intelligence **8**, 376–40 (1992)
5. Guivarch, V., Camps, V., Pninou, A.: Context awareness in ambient systems by an adaptive multi-agent approach. In: F. Patern, B. Ruyter, P. Markopoulos, C. Santoro, E. Loenen, K. Luyten (eds.) Ambient Intelligence, *Lecture Notes in Computer Science*, vol. 7683, pp. 129–144. Springer Berlin Heidelberg (2012)
6. Meneguzzi, F., Zorzo, A.F., da Costa Móra, M., Luck, M.: Incorporating planning into BDI agents. Scalable Computing: Practice and Experience **8**, 15–28 (2007)
7. Olaru, A., Florea, A.M., El Fallah Seghrouchni, A.: A context-aware multi-agent system as a middleware for ambient intelligence. MONET **18**(3), 429–443 (2013)
8. Rao, A.S., Georgeff, M.P.: An abstract architecture for rational agents. In: B. Nebel, C. Rich, W.R. Swartout (eds.) KR, pp. 439–449. Morgan Kaufmann (1992)
9. Sardina, S., de Silva, L., Padgham, L.: Hierarchical planning in bdi agent programming languages: a formal approach. In: AAMAS '06 conference on Autonomous agents and multiagent systems, 1001–1008. ACM, New York, NY, USA (2006)
10. Singh, D., Sardina, S., Padgham, L., James, G.: Integrating learning into a BDI agent for environments with changing dynamics. In: C.K. Toby Walsh, C. Sierra (eds.) Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 2525–2530. AAAI Press, Barcelona, Spain (2011)