Multi-Robot LTL Planning Under Uncertainty

Appendix

A1. Problem formulation

We provide additional definitions related with Section 3.

Definition 7. *A* definitive robot model *$r = (S, init, A, \Pi, T, Meet, L)$ is a* partial robot model *where $T = T_p$, $Meet = Meet_p$ and $L(\alpha, \pi) \neq ?$ for all (α, π) such that $\alpha \in A$ and $\pi \in \Pi$.*

Informally, a definitive robot model is a partial robot model that does not contain uncertainty.

Definition 8. *A* definitive robot application *$\mathcal{H}$ is a set of definitive robot models $\{r_1, r_2, \ldots, r_N\}$ such that for every couple of definitive (partial) robot models $r_n, r_m \in \mathcal{H}$, where r_n=(S_n, $init_n$, A_n, Π_n, T_n, $T_{p,n}$, $Meet_n$, $Meet_{p,n}$, L_n) and r_m=(S_m, $init_m$, A_m, Π_m, T_m, $T_{p,m}$, $Meet_m$, $Meet_{p,m}$, L_m), the following is satisfied $\Pi_n \cap \Pi_m = \emptyset$ and $A_n \cap A_m = \emptyset$.*

Informally, a definitive robot application is a partial robot application where all the models of the robots are definitive.

Consider the notion of definitive and possible plan presented in Definition 2. We use the notation $\mathcal{B}_d(r)$ and $\mathcal{B}_p(r)$ to indicate the set of all definitive and possible plans of a robot r, respectively. Note that $\mathcal{B}_d(r) \subseteq \mathcal{B}_p(r)$, i.e., a definitive plan is also a possible plan.

Definition 9. *Given two partial robot models r and r', the partial robot model r'* refines *r, i.e., $r \preceq r'$, iff $\mathcal{B}_d(r) \subseteq \mathcal{B}_d(r')$ and $\mathcal{B}_p(r') \subseteq \mathcal{B}_p(r)$.*

A model r' refines r if it contains more definitive plans, i.e., definitive plans of $\mathcal{B}_d(r)$ are a subset (or equal) of definitive plans of r' and if the partial information about r is reduced, i.e., possible plans of $\mathcal{B}_p(r')$ are a subset (or equal) of possible plans of r.
If the obtained partial robot model r' is a definitive robot model, it is a completion of a partial robot model r.

Definition 10. *Given a definitive robot model r' and a partial robot model r, r' is a* completion *of r iff $r \preceq r'$.*

We define $\mathcal{C}(r)$ as the set of all the completions of a partial robot model r.

Definition 11. *A partial robot application $\mathcal{H}$' defined over the set of robots $\{r'_1, r'_2, \ldots, r'_N\}$ is a refinement of a partial robot application $\mathcal{H}$ defined over the set of robots $\{r_1, r_2, \ldots, r_N\}$ if and only if $r_n \preceq r'_n$ holds for every robot $r_n \in \{r_1, r_2, \ldots, r_N\}$.*

Informally, a partial robot application $\mathcal{H}'$, refines a partial robot application $\mathcal{H}$ if the robots in $\mathcal{H}'$ refine the one in $\mathcal{H}$.

Definition 12. *A definitive robot application $\mathcal{H}$' is a completion of a partial robot application $\mathcal{H}$ if and only if $\mathcal{H}$' refines $\mathcal{H}$.*

A definitive robot application is obtained by refining robots of a partial robot application.

A plan is *executed* by a robot.

Definition 13. *Given a partial robot $r = (S, init, A, \Pi, T, T_{p,}, Meet, Meet_{p,}, L)$, one of its completion $r_c = (S_c, init_c, A_c, \Pi_c, T_c, T_{p,c}, Meet_c, Meet_{p,c}, L_c)$ representing the actual model of the system and a plan $\beta = s_1, \alpha_1, s_2, \alpha_2, \ldots$. An execution of a plan is an infinite sequence $q_1, \alpha_1, q_2, \alpha_2, \ldots$ or a finite sequence $q_1, \alpha_1, q_2, \alpha_2, \ldots \alpha_{n-1}, q_n$ of states and actions.*

(1) The execution of a plan is an an infinite sequence $q_1, \alpha_1, q_2, \alpha_2, \ldots$ if and only if $q_1 = s_1$ and for all $i >= 1$, $q_{i+1} = s_{i+1}$ and $s \xrightarrow{\alpha_i} s' \in T_c$ and for every service π such that $L(\alpha_i, \pi) = ?$, $L_c(\alpha_i, \pi) = T$ and $Meet_c(q_{i+1}) = Meet_p(s_{i+1})$.
(2) The execution of a plan is a maximal finite sequence $q_1, \alpha_1, q_2, \alpha_2, \ldots \alpha_{n-1}$, q_n if and only if it is not an infinite sequence and $q_1 = s_1$ and for all $1 \leq i < n$, $q_{i+1} = s_{i+1}$ and $s \xrightarrow{\alpha_i} s' \in T_c$ and for every service π such that $L(\alpha_i, \pi) = ?$, $L_c(\alpha_i, \pi) = T$ and $Meet_c(q_{i+1}) = Meet_p(s_{i+1})$.

Informally, a plan is executed as long as there is true evidence about partial information about action execution, service provisioning and meeting capabilities in the actual model of the environment r_c, i.e., all the transitions of the plan are firable, the services are provided and synchronization is possible in r_c. If this is not the case, the robot stops execution the plan, i.e., the execution of the plan is finite and stops in the state q_n.

As plans are executed the model of the robot is updated.

Definition 14. *Given a partial robot $r = (S, init, A, \Pi, T, T_{p,}, Meet, Meet_{p,}, L)$, one of its completion $r_c = (S_c, init_c, A_c, \Pi_c, T_c, T_{p,c}, Meet_c, Meet_{p,c}, L_c)$ representing the actual model of the system and a plan $\beta = s_1, \alpha_1, s_2, \alpha_2, \ldots$.*
If the execution of the plan has the form $q_1, \alpha_1, q_2, \alpha_2, \ldots$, then $r_1 = r = (S, init, A, \Pi, T, T_{p,}, Meet, Meet_{p,}, L)$ and at each step $i > 1$ of the execution a new model $r_i = (S_i, init_i, A_i, \Pi_i, T_i, T_{p,i}, Meet_i, Meet_{p,i}, L_i)$ is generated from r_{i-1} as follows. First r_{i-1} is copied in r_i and then the following changes are applied:

- *if $s \xrightarrow{\alpha_i} s' \in T_c$ and $s \xrightarrow{\alpha_i} s' \notin T_{i-1}$ then $s \xrightarrow{\alpha_i} s' \in T_i$;*
- *for every service π if $L_{i-1}(\alpha_i, \pi) = ?$, $L_c(\alpha_i, \pi) = \top$ then $L_i(\alpha_i, \pi) = \top$;*
- *if $Meet_c(q_{i+1}) \neq \emptyset$ and $Meet_{i-1}(s_{i+1}) = \emptyset$ then $Meet_i(s_{i+1}) = Meet_c(q_{i+1})$.*

If the execution e of the plan has the form $q_1, \alpha_1, q_2, \alpha_2, \ldots \alpha_{n-1}, q_n$, then $r_1 = r = (S, init, A, \Pi, T, T_{p,}, Meet, Meet_{p,}, L)$ and at each step $1 < i \leq n$ of the *execution a new model* $r_i = (S_i, init_i, A_i, \Pi_i, T_i, T_{p,i}, Meet_i, Meet_{p,i}, L_i)$ *is generated from* r_{i-1} *as follows. First* r_{i-1} *is copied in* r_i *and then the following changes are applied:*

– *if* $s \xrightarrow{\alpha_i} s' \in T_c$ *and* $s \xrightarrow{\alpha_i} s' \notin T_{p,i-1}$ *then* $s \xrightarrow{\alpha_i} s' \notin T_{p,i}$*;*
– *if* $s \xrightarrow{\alpha_i} s' \notin T_c$ *and* $s \xrightarrow{\alpha_i} s' \notin T_{i-1}$ *then* $s \xrightarrow{\alpha_i} s' \in T_i$*;*
– *for every service* π *if* $L_{i-1}(\alpha_i, \pi) = ?$*,* $L_c(\alpha_i, \pi) = \top$ *then* $L_i(\alpha_i, \pi) = \top$*;*
– *for every service* π *if* $L_{i-1}(\alpha_i, \pi) = ?$*,* $L_c(\alpha_i, \pi) = \bot$ *then* $L_i(\alpha_i, \pi) = \bot$*;*
– *if* $Meet_c(q_{i+1}) \neq \emptyset$ *and* $Meet_{i-1}(s_{i+1}) = \emptyset$ *then* $Meet_i(s_{i+1}) = Meet_c(q_{i+1})$*;*
– *if* $Meet_p(q_{i+1}) = \emptyset$ *and* $Meet_{p,i-1}(s_{i+1}) \neq \emptyset$ *then* $Meet_{p,i}(s_{i+1}) = \{\#\}$*.*

Informally, when the execution of the plan is infinite only true evidence about the partial information is detected and the model r_i is updated with the information present in the completion r_c. When the execution of the plan is finite false evidence about partial information is detected when the plan reaches state q_n. The model r_n is updated with the detected information.

Theorem 2. *Let us consider a partial robot model* r*. As the robot executes a plan, the models* r_i *obtained by updating model* r *as specified in Definition 14 are refinements of* r*.*

Proof. When the run is infinite, true evidence about the partial information is detected, the transition is transformed into a definitive transition, the service is provided or the meeting is possible. By Definition 2 every plan that was involving the transition, the services or the meeting capability was a possible plan. Since true evidence is detected, possible plans are not removed from the partial model, i.e., $\mathcal{B}_p(r') = \mathcal{B}_p(r)$. Since partial knowledge about a transition or a service or a meeting capability is transformed into definitive knowledge, plans that were possible may now be also definitive, i.e., additional plans are added to the set of definitive plans of r', i.e., $\mathcal{B}_d(r) \subseteq \mathcal{B}_d(r')$. Thus, $\mathcal{B}_d(r) \subseteq \mathcal{B}_d(r')$ and $\mathcal{B}_p(r') \subseteq \mathcal{B}_p(r)$.
When the run is infinite, true evidence about the partial information is detected, until state q_n is reached. The execution stops in q_n since false evidence about the partial information is detected when the robot tries to perform transition s_i, α_i, s_{i+1} such that $s_i = q_n$. Thus, by Definition 14 the transition is removed, the service is not provided or the meeting is not possible. Since the plan was involving partial information, by Definition 2 it was a possible plan. Furthermore, by Definition 2, any plan that involved the removed transition, the not provided service or the meeting capability was a possible plan. Thus, rules (1), (2) and (3) do not modify the set of definitive plans of r when false evidence about partial information is detected, i.e., $\mathcal{B}_d(r) = \mathcal{B}_d(r')$. On the contrary, by removing the transition, by not providing the service or by avoid meeting the set of possible plans is reduced, i.e., $\mathcal{B}_p(r') \subseteq \mathcal{B}_p(r)$. Thus, $\mathcal{B}_d(r) \subseteq \mathcal{B}_d(r')$ and $\mathcal{B}_p(r') \subseteq \mathcal{B}_p(r)$.

Definition 15. *Consider a* partial robot application $\mathcal{H} = \{r_1, r_2 \ldots r_n\}$ *and a set of definitive (possible)* compatible *plans* $\mathcal{B}_d$ *(*$\mathcal{B}_p$*). The execution of the plans* *is either an infinite sequence* $(q_{1,1}, q_{1,2} \ldots, q_{1,n})(\alpha_{1,1}, \alpha_{1,2} \ldots, \alpha_{1,n})(q_{2,1}, q_{2,2} \ldots, q_{2,n}) \ldots$ *or a finite sequence* $(q_{1,1}, q_{1,2} \ldots, q_{1,n})(\alpha_{1,1}, \alpha_{1,2} \ldots, \alpha_{1,n})(q_{2,1}, q_{2,2} \ldots, q_{2,n}) \ldots (q_{n,1}, q_{n,2} \ldots, q_{n,n})$.

(1) The execution of the plans $\mathcal{B}_d$ *is an infinite sequence* $(q_{1,1}, q_{1,2} \ldots, q_{1,n})(\alpha_{1,1}, \alpha_{1,2} \ldots, \alpha_{1,n})(q_{2,1}, q_{2,2} \ldots, q_{2,n}) \ldots$ *if and only if for every* $1 \leq i \leq n$*, the infinite sequence* $q_{i,1}, \alpha_{i,1}, q_{i,1} \ldots$ *is an execution of* r_i*;*

(2) The execution of the plans $\mathcal{B}_d$ *is a maximal finite sequence* $(q_{1,1}, q_{1,2} \ldots, q_{1,n})(\alpha_{1,1}, \alpha_{1,2} \ldots, \alpha_{1,n})(q_{2,1}, q_{2,2} \ldots, q_{2,n}) \ldots (q_{n,1}, q_{n,2} \ldots, q_{n,n})$ *if and only if it is not an infinite sequence and for every* $1 \leq i \leq n$*, the infinite sequence* $q_{i,1}, \alpha_{i,1}, q_{i,1} \ldots q_{i,n}, \alpha_{i,n} \ldots$ *is an execution of* r_i *or the finite sequence* $q_{i,1}, \alpha_{i,1}, q_{i,1} \ldots q_{i,m}$ *where* $m \geq n$ *is an execution of* r_i*.*

Informally, the execution of the plans of a partial robot application is infinite or it stops when one of the execution of one of its robots stops.

Theorem 3. *Consider a partial robot application* $\mathcal{H}$*. As the robots execute theis plans, robot application* $\mathcal{H}$*' obtained by updateing all the models of the robotic application as specified in Definition 14 is a refinement of* $\mathcal{H}$*.*

Proof. It follows from Theorem 2 and from Definition 11.

A2. Planning with partial knowledge

We provide additional definitions related with Section 4.

The three-valued LTL semantics is based on the ordering relation $>$ between values $\{\top, ?, \bot\}$ such that $\top > ? > \bot$. Based on this ordering, the following functions are defined as usual [5]: (1) complement (*comp*) such that *comp*$(\top) = \bot$, *comp*$(\bot) = \top$ and *comp*$(?) = ?$; (2) minimum (min) such that $\min(\top, \bot) = \bot$, $\min(\top, ?) = ?$ and $\min(?, \bot) = \bot$ and (3) maximum (max) such that $\max(\top, \bot) = \top$, $\max(\top, ?) = \top$ and $\max(?, \bot) = ?$.

Definition 16. *Let us consider a property* ϕ *and a word* $w_\tau = \varpi_1 \varpi_2 \ldots$*. The three valued semantics* $\overset{3}{=}$ *associates to a word* w_τ *a value in the set* $\{\top, ?, \bot\}$ *as follows:* $[\varpi_i \models \pi] \overset{3}{=} \varpi_i(\pi)$

$[\varpi_i \models \neg\phi] \overset{3}{=} \mathrm{comp}([\varpi_i \models \phi])$

$[\varpi_i \models \phi_1 \wedge \phi_2] \overset{3}{=} \min([\varpi_i \models \phi_1], [\varpi_i \models \phi_2])$

$[\varpi_i \models X\phi] \overset{3}{=} [\varpi_{i+1} \models \phi]$

$[\varpi_i \models \phi_1 U \phi_2] \overset{3}{=} \max_{j \geq 0}(\min(\{[\varpi_k \models \phi_1] | i \leq k < j\} \cup \{[\varpi_j \models \phi_2]\}))$

Words can be refined by assigning values to the unknown propositions.

Definition 17. *Given two possible words* $w_\tau = \varpi_1 \varpi_2 \ldots$ *and* $w'_\tau = \varpi'_1 \varpi'_2 \ldots$ *such that for all* $\pi \in \Pi$ *and* $i \in \{1, 2 \ldots\}$*,* $\varpi_i(\pi) \in \{\top, \bot, ?\}$ *and* $\varpi'_i(\pi) \in \{\top, \bot, ?\}$*, the (possible) word* w'_τ *is a refinement of a possible word* w_τ*, indicated as* $w_\tau \preceq w'_\tau$ *if and only if the following conditions hold*

(1) for all $i \geq 0, \pi \in \Pi$, if $w_i(\pi) = \top \rightarrow w'_i(\pi) = \top$;
(2) for all $i \geq 0, \pi \in \Pi$, if $w_i(\pi) = \bot \rightarrow w'_i(\pi) = \bot$.

Word w'_τ is obtained from w_τ by assigning to some $w_i(\pi) = ?$ a $\top$ or a $\bot$ value.

Definition 18. *A definitive word $w'_\tau = \varpi'_1 \varpi'_2 \ldots$ is a completion of a possible word w_τ if and only if w'_τ refines w_τ.*

A definitive word w'_τ that refines a possible word w_τ is a completion of w_τ.

Lemma 1. *Given two words w_τ and w'_τ, such that $w_\tau \preceq w'_\tau$, and an LTL formula ϕ the following are satisfied: (1) $[w_\tau \models \phi] \overset{3}{=} \top \Rightarrow [w'_\tau \models \phi] \overset{T}{=} \top$; (2) $[w_\tau \models \phi] \overset{3}{=} \bot \Rightarrow [w'_\tau \models \phi] \overset{T}{=} \bot$.*

Proof. Lemma 1 has been proved to hold for partial models, i.e., it has been proved to hold for partial Kripke structures (PKSs) [6]. A word $w_\tau = \varpi_1 \varpi_2 \ldots$ can be considered as a PKS obtained as follows:

(1) a new state s_i of the PKS is created for every $\varpi_i \in w_\tau$;
(2) the initial state of the PKS is state s_0;
(3) each service π is associated with a proposition a of the PKS;
(4) for each state s_i of the PKS the function L associates to each proposition a in the set of proposition of the PKS the value $\varpi_i(\pi)$;
(5) for each s_i, s_{i+1} a transition is added from s_i to s_{i+1}.

Thus, since words w_τ and w'_τ can be considered as PKS, it follows that Lemma 1 also holds for words.

Since word w_τ can be considered as a PKS M obtained using the procedure previously specified, the model checking procedure can encode the word in a PKS which is then checked against property ϕ. Checking whether a PKS satisfies a property ϕ w.r.t. the thorough LTL semantics can be done in polynomial time in the length of w_τ and double exponential time in the size of ϕ [15].

A3. Algorithm

We prove correctness of Algorithm 1 and provide additional definitions on how Algorithm 1 is called.

Theorem 1. *Consider a partial robot application $\mathcal{H}$, a set of missions Φ (one for each robot) and the three-valued LTL semantics ($\overset{3}{=}$). A set of plans $\mathfrak{B}$ that (1) are compatible and (2) $\mathfrak{B}$* locally definitively (resp. possibly) satisfies *each ϕ_n w.r.t. the three-valued semantics. is returned from Algorithm 1 if and only if they exist in $\mathcal{H}$. If formulae are self-minimizing this theorem also applies to the thorough LTL semantics ($\overset{T}{=}$).*

Proof. If a plan is found in Step 5, it is a definitive plan. The plan is obtained by only executing definitive transitions. Furthermore, the considered model is obtained putting the formula in negation normal form and setting the values of the services in the complement closed structure to $\bot$, meaning that the services are not provided. If a mission is found it definitely satisfies the property of interest w.r.t. the three valued semantics. Intuitively, if a plan is found in the pessimistic approximation it is a definitive plan, since the pessimistic approximation the system "does it best" to not allow the computation of plans and thus do not involve any partial information. See [5] and [2] for further details. If the decentralized planner used by MAPmAKER returns compatible plans, the plans found in Step 5 are also compatible.

If a plan is found in Step 8, it is a possible plan. The plan is obtained by executing only definitive and possible transitions. Furthermore, the plans are obtained putting the formula in negational normal form and setting the values of the services in the complement closed structure to $\top$. If a mission is found it possibly satisfies the property of interest w.r.t. the three valued semantics. Intuitively, if a plan is found in the optimistic approximation it is a possible plan, since the optimistic approximation the system "does it best" to allow the computation of plans. If the decentralized planner used by MAPmAKER returns compatible plans, the plans found in Step 5 are also compatible.

Thus, plans returned by Algorithm 1 are compatible and locally definitively (resp. possibly) satisfies each ϕ_n w.r.t. the three-valued semantics.

The prove that if a set of plans that are compatible and locally definitively (resp. possibly) satisfies each ϕ_n w.r.t. the three-valued semantics exists, they are returned by Algorithm 1 is performed by contradiction. Assume a set of plans that are compatible and locally definitively (resp. possibly) satisfies each ϕ_n exist and Algorithm 1 does not return any plan. Then, no plans are found in the optimistic and pessimistic approximations. Since in the optimistic approximation the MAPmAKER "does it best" to allow the computation of plans and no plans are returned, then, no possible plans are present in the robotic application contradicting the hypothesis.

When the thorough LTL semantics is considered Algorithm 1 may return possible plans that are not expected. Specifically, the algorithm may return spourious possible plans, i.e., plans that can not executed or plans that definitively satisfy the property of interest. As showed in [14], there exists a subset of LTL formulae, known in literature as *self-minimizing* formulae, such that the three valued and the thorough semantics coincide [14]. In this case, Theorem 1 is also correct w.r.t. thorough LTL semantics.

Remark 1. This work does not discuss how to choose between possible and definitive plans. Possible plans can be computed in cases in which definitive plans are not present or they can always be computed and appropriate policies can be used to select between definitive and possible plans. A policy may choose for example the plan with the shortest length, or it may consider non-functional aspects of the plans e.g., time to perform certain actions, or likelihood of detecting true or false evidence about partial information. In the following we assume that

Algorithm 2 The MAPmAKER main loop.

```
1: Input a partial robot application $\mathcal{H}$ and a set of missions $\Phi$
2: Output a set of definitive plans (if they exist)
3: $\{\beta_1, \beta_2, \ldots, \beta_N\} = CREATE\_PLANS(\mathcal{H})$;
4: $\{dp_1, dp_2, \ldots, dp_M, \Psi_1, \Psi_2, \ldots \Psi_M\} = DEP\_CLASSES(\mathcal{H}, \Phi)$;
5: for $dp_n \in \{dp_1, dp_2, \ldots, dp_M\}$ do
6:   $disc = 1$
7:   while $disc == 1$ do
8:     $disc = 0$;
9:     $[pp, \{pd_1, pd_2, \ldots, pd_M\}]{=}PARTIAL\_PLAN(dp_n, \Psi_n)$;
10:    if $pp = false$ then return $NO\_PLAN$; end if
11:    $i = 1$;
12:    while $i \leq LENGTH(\{dp_1, dp_2, \ldots, dp_M\})$ & $disc == 1$ do
13:      $(disc, tr) = TR(\{dp_1, dp_2, \ldots, dp_M\}, i, \{\beta_1, \beta_2, \ldots, \beta_N\}, tr)$;
14:      $i = i + 1$;
15:    end while
16:  end while
17: end for
18: return $(\{\beta_1, \beta_2, \ldots, \beta_N, \}, tr)$
```

the planner always chooses the shortests between the possible and the definitive plan. Note that, in this case, it is still necessary to execute Step 1 of Algorithm 2 since Step 2 may return a possible plan while a definitive plan of the same length is present. This policy may for example reduce energy consumption, since every action performed by the robots may consume energy. Thus, shorter plans require less energy.

Algorithm 1 is performed at the beginning and every time false evidence about partial information is detected. More precisely, Algorithm 2 describes the main loop of MAPmAKER that is performed when Algorithm 1 is called.

MAPmAKER gets as input a partial robot application $\mathcal{H}$ and a set of missions Φ, i.e., one for each robot in the partial robot application. First (Line 3), MAPmAKER creates a set of empty plans, one for each robot of the partial robot application. Then (Line 4), the set of robots in the partial robot application is partitioned into dependency classes based on the missions the robots have to achieve. Each dependency class $dp_n \in \{dp_1, dp_2, \ldots, dp_M\}$ is a partial robot application, each $\Psi_n \in \{\Psi_1, \Psi_2, \ldots, \Psi_M\}$ contains a set of missions for the robots in the partial robot application dp_n. Then, each dependency class dp_n is analyzed (Lines 5-17). The variable *disc* (Line 6) is used to track whether new information about the partial robot models in dp_n has been discovered while plans are executed. Then the planner for partial robot application is executed on dp_n (Line 9). If the variable *pp* is equal to the value *false*, no plan is available and the value NO_PLAN is returned (Line 10). Otherwise (Lines 11-15), the plans returned by the planner are executed. The algorithm executes one action for each robot per time, by calling function TR until the maximum length of the plans is reached. Variable *tr* keeps track of the trace generated by performing executions. It contains a sequence $(q_{1,1}, q_{1,2} \ldots, q_{1,n})(\alpha_{1,1}, \alpha_{1,2} \ldots,$

$\alpha_{1,n})(q_{2,1}, q_{2,2} \ldots, q_{2,n}) \ldots$ which is updated by TR at each iteration. When the procedure ends tr is such that for all $j \geq 1$, $(q_{j,1}, q_{j,2} \ldots, q_{j,n})$ it contains the states of the robots in the robotic applications at step j and $\alpha_{j,1}, \alpha_{j,2} \ldots, \alpha_{j,n}$ are the performed actions. Variable i keeps the index of the actions to be executed next (Line 13). Note that, as specified in Section 3 when actions are executed by robots the models of the robots are updated. If one of the robots in the partial robot application detects a *false evidence* about partial information, value 0 is returned and assigned to variable *disc*. Otherwise, value 1 is assigned to variable *disc*. When *false evidence* about partial information is detected, the loop of Line 12 is left and the planner for partial robot applications (Line 9) is re-executed.

Consider the trace generated using Algorithm 2 and contained in the variable tr has the form $(q_{1,1}, q_{1,2} \ldots, q_{1,n})(\alpha_{1,1}, \alpha_{1,2} \ldots, \alpha_{1,n})(q_{2,1}, q_{2,2} \ldots, q_{2,n}) \ldots$ The *word* associated with the execution β is $w_\tau = \varpi_1 \varpi_2 \ldots$, such that for each $i > 1$ and $\pi \in \Pi_i$, $\varpi_i(\pi) = L_n(\alpha_i, \pi)$.

Correctness results of Algorithm 2 depend on the employed planning algorithm and on the model of the robotic application.

- since the planner is re-executed any time false evidence about partial information is detected, if the planning algorithm is not able to manage history (services detected before a re-planning activity is performed), the plan is correct only considering the portion of the trace that starts from the last instant the planner has been called. Viceversa, if the planner is able to manage history, correctness is ensured w.r.t., the all trace. The planning algorithm presented in [38] and considered in this work does not perform the planning from scratch every time false evidence is detected, but is able to consider the already performed trace.
- if the model of the robotic application is such that when movement is possible from cell c_i to c_j it is also possible from c_j to c_i, then, if a definitive plan is present in the actual model of the robotic application it is found by Algorithm 2. In this case, as Algorithm 2 is performed, new knowledge about the real robot application is added to the partial robot application, until (in the worst case) the model of the partial robot application corresponds to the real robot application. If a set of plans that are compatible and and $\mathfrak{B}$ locally definitively satisfies each ϕ_n exist, in the worst case, it must be returned when Algorithm 1 is performed on the model of the real robot application. Assume now that a movement is possible from cell c_i to c_j, but not from c_j to c_i and that the set of plans $\mathfrak{B}$ that are compatible and such that $\mathfrak{B}$ locally definitively satisfies each ϕ_n, do not require one of the robots to move from c_i to c_j. The robot may move from c_i to c_j while it is following a possible plan. However, after reaching c_j and performing the rest of the possible plan it may detect false evidence about the partial information. At this point, it may be impossible to came back to c_i and to compute the set of plans $\mathfrak{B}$.