A Framework for Interleaving Planning-while-Learning and Execution

Marcello Balduccini

Centro di Ricerca "Informatica Interattiva"
Università degli Studi dell'Insubria
via Ravasi 2, I-21100 Varese (Italy)
ph:+39-332-250212 fax:+39-332-281308
e-mail: marcy@mail.varbio.unimi.it

Abstract. Interacting with the environment in presence of incomplete information requires the ability to acquire new knowledge from the interaction with the environment and to employ it when deliberating about which actions to execute. The ability to identify a particular environmental behaviour by inspecting perceptual feedback greatly contributes to completing the knowledge available to the agent. This paper introduces a formal framework for interleaving planning-while-learning and execution in partially specified environments. Planning-while-learning combines conventional planning with the search of the environmental behaviour model that best fits the experienced behaviour of the environment. Heuristics for early termination of planning and assumptions are used in order to reduce the cost of planning. Sufficiency conditions are given that guarantee the soundness and the completeness of the agent's control system w.r.t. the environmental model and the goal.

1 Introduction

In the area of intelligent autonomous agents and of autonomous robots, many different approaches have been attempted to reconcile the need to carefully plan sequences of actions with the need to interact timely with a rapidly evolving environment.

Some researchers have proposed to substitute planning with more light-weighted reasoning techniques, like behaviours ([9], [8]), policies ([3], [4], [10], [2]) and others. The drawback of these solutions is that, while they solve the problem of a timely interaction with the environment, they pose new problems because of the general reduction in the representation and reasoning power.

On the other hand, other researchers have developed solutions which reduce the cost of planning, but do not affect the representation and reasoning ability. State of the art approaches ([11], [12], [1], [6], [7]) adopt early termination heuristics, assumptive planning, abstract planning, often combined together, to form a powerful reasoning device able to interact timely with the environment.

Even if it is a widely spread opinion that learning may increase the autonomy of both agents and robots by allowing them to refine their a-priori

environmental model according to the empirical experience, examples of well formalized frameworks for agents integrating learning and planning while acting timely with the environment are rare ([7]).
The present paper tries to fill this lack by introducing a formal framework for interleaving planning-while-learning and execution in partially specified environments. In presence of incomplete information, interleaving ([5]) is a very efficient way to perform planning, since it allows to exploit the perceptual feedback from the environment in order to increase the knowledge available to the agent before a new planning phase is started, thus reducing its computational cost. Planning-while-learning ([14]) combines conventional planning with the search, within a given initial set, of the environmental behaviour model that best fits the experienced behaviour of the environment.
Our approach is based upon Nourbakhsh's work on the interleaving of planning and execution ([11], [12]) and Safra's and Tennenholtz's paper on planning-while-learning ([14]); particular attention is paid to the sufficiency conditions that guarantee the soundness and the completeness of the agent's control system w.r.t. the environmental model and the goal. Heuristics for early termination of planning and assumptions about both the current state of the world and the actual environmental behaviour are used in order to reduce the cost of the planning process.
The paper is structured as follows. In Sect. 2 we introduce our version of Nourbakhsh's framework for the interleaving of planning and execution, which represents the state of the art in this area. It will constitute the basis for Sect. 3, where we describe our framework for interleaving planning-while-learning with execution and deal with soundness and completeness. In Sect. 4 we compare our work with relevant research in this area. In Sect. 5 we draw the conclusions from our work and highlight possible future developments.

2 Interleaving Planning and Execution

With respect to modeling a robot's interaction with the environment, Nourbakhsh ([11]) has shown that the world may be viewed as a system made up of two distinct, interacting parts: the robot and its environment, which are connected only by the percept and the action streams. In our framework, a *percept* is formally defined as a vector representing the instantaneous image of the robot's inputs. An *action* is a vector representing the instantaneous image of the robot's outputs.
The interaction between the agent and the environment is modeled as the interaction between two finite automata, the robot finite automaton and the environment finite automaton.

The **robot finite automaton** or *rfa* is defined as a tuple $<P, B, A, int, ext, b>$, where:
- P is a set of input objects (the agent's possible percepts);
- B is a set of internal states;
- A is the set of output objects (the agent's possible actions);
- int is a function $P \times B \to B$ that updates the robot's internal state;

- ext is a function $B \to A$ that prescribes the robot's output;
- b is a member of B (the robot's initial internal state).

The **environmental finite automaton** or *efa* is defined as a tuple $<A, S, P, do, see, s>$, where:
- A is the environment's set of input objects (the robot's actions);
- S is the environment's set of states;
- P is the environment's set of output objects (the robot's percepts);
- do is a function $A \times S \to S$ that updates the environment's state;
- see is a function $S \to$ P that updates the robot's percept;
- s is a member of S (the initial state of the world).

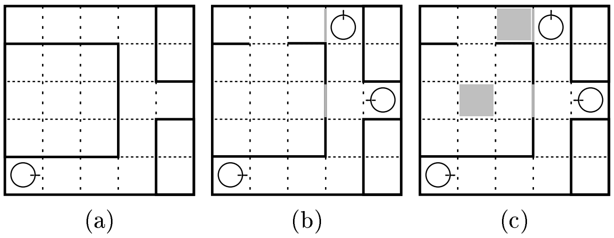

Fig. 1. (a) A simple maze. (b) An incompletely specified maze. (c) A maze with objects with incompletely specified behaviour.

Example 1. Suppose that our robot is put in the maze depicted in Fig. 1(a) at position $(1, 1)$. The robot has four binary inputs, corresponding to four wall detectors, placed one on each side of the robot. Three actions are available: *ahead*, *turn-left* and *turn-right*. The control system of the robot is programmed to go ahead whenever possible and to turn left otherwise. The $<rfa, efa>$ system for this setting may be described as follows:
- P is the set of all binary-valued vectors of length 4;
- $A = \{ahead, turn-left, turn-right\}$;
- S is the set of tuples $<maze-conf, pos>$ where $maze-conf$ is the configuration of the maze shown in Fig. 1(a) and pos is the position of the robot, ranging all over the maze, and its bearing (n,s,e,w);
- s is the tuple $<maze-conf, (1, 1, e)>$;
- $B = \{ahead\text{–}clear, ahead\text{–}blocked, start\}$ where $start$ represents the robot's initial state, when nothing is known yet about the sensors' state;
- b is set to $start$;
- do updates the position of the robot according to the action performed;
- see returns the value of the input channels of the robot according to the position of the walls around the robot;

– *int* is defined as follows:

$$int(p,b) = \begin{cases} ahead{-}clear & \text{if ahead-bit of } p \text{ is } 0 \\ ahead{-}blocked & \text{if ahead-bit of } p \text{ is } 1 \end{cases};$$

– *ext* is defined as follows:

$$ext(b) = \begin{cases} ahead & \text{if } b = ahead{-}clear \\ turn{-}left & \text{if } b = ahead{-}blocked \end{cases}.$$

In many cases, the unreliability of sensors and effectors, as well as incomplete knowledge about the environment, prevent the designer from coming up with an *efa*, which by definition doesn't permit to express *incomplete information.*
For this reason we introduce a **partially specified environmental finite automaton** or *pefa*, which is defined as a tuple $<A, S, P, \mathit{effect}, sense, I>$, where:

- A, S and P are defined like in *efa*;
- *effect* is a function $A \times \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ that updates the environment's set of possible states;
- $sense$ is a function $P \times \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ that returns the maximal subset of the second argument consistent with the robot's percept;
- I is a subset of $\mathcal{P}(S)$ (the set of possible initial states of the world).

The *pefa* makes it possible to express incomplete information by reasoning in terms of the set of the environment's *possible* states (the so-called *state-set*).

Example 2. Suppose that our robot is put in the maze depicted in Fig. 1(b). There is incomplete information about the initial position, which may be either $(1,1)$ or $(5,3)$ or $(4,5)$, and about its initial bearing, like shown in figure. There is incomplete knowledge, as well, about the walls drawn in grey: only one of them exists, but it is not known which one. The other features of the environment are like in Example 1.
The *pefa* for this setting may be described as follows:

- P and A are defined like in Example 1;
- S is the set of tuples $<maze{-}conf, pos>$ where $maze{-}conf$ is one of the two possible configurations of the maze shown in Fig. 1(b) and pos is defined like in Example 1;
- I is the set of tuples $<maze{-}conf, pos>$, where $maze{-}conf$ is one of the two possible configurations of the world and pos is either $(1,1,e)$ or $(5,3,w)$ or $(4,5,n)$;
- $\mathit{effect}(a, S')$ and $sense(p, S')$ are defined following the environment's specifications.

Intuitively, many *efa*'s are "compatible" with a given *pefa*. We formally define the notion of *consistency* as follows:

Definition 1. *A pefa* $<A, S, P, \mathit{effect}, sense, I>$ *is consistent with an efa* $<A, S, P, do, see, s>$ *iff:*

1. $\forall S' \subseteq S \;\; \forall p \in P \;\; sense(p, S') \supseteq \{\bar{s} \mid \bar{s} \in S' \wedge see(\bar{s}) = p\}$;
2. $\forall S' \subseteq S \;\; \forall a \in A \;\; \mathit{effect}(a, S') \supseteq \{\bar{s} \mid \exists s' \; (s' \in S' \wedge \bar{s} = do(a, s'))\}$;
3. $s \in I$.

Consistency of a *pefa* with an *efa* guarantees that a state-set tracker (see Definition 3), applied to an environment modeled by the *efa*, tracks the environment using the information provided by the *pefa*, despite its incompleteness.
Now we can define the planning problem in terms of a *pefa* and a goal. A *problem instance* is a tuple <*pefa*, *G*> where *pefa* is defined as usual and G is a *set of state sets*, representing the disjunction of mutually exclusive goals, each expressed by a state set. The concept of solution of a planning problem is formalized by the following definitions.

Definition 2. *Satisfaction of a goal G by a state-set I is expressed by the relation $satisfies(I, G)$ defined upon $\mathcal{P}(S) \times \mathcal{P}(\mathcal{P}(S))$ as follows: $satisfies(I, G) \iff \exists g \; (g \in G \wedge I \subseteq g)$.*

Definition 3. *A* state set tracker (sst) *for a pefa $<A, S, P, effect, sense, I>$ and an action-percept history $\alpha = \{a_1, p_1, a_2, p_2, \ldots, a_n, p_n\}$ is a robot finite automaton $<P, B, A, int, ext, b_1>$ where: (1) $B = \mathcal{P}(S)$; (2) $int(p_i, b_i) = sense(p_i, effect(a_i, b_i))$; (3) $ext(b_i) = a_i$; (4) $b_1 = I$.*

Definition 4. *An rfa is a* solution *to a problem instance <pefa, G> iff, for every efa consistent with pefa, given the action-percept history $\alpha = \{a_1, p_1, a_2, p_2, \ldots, a_n, p_n\}$ of the system <rfa, efa>, the state set tracker $sst(pefa, \alpha)$ in system $< sst, efa >$ computes b_n s.t. $satisfies(b_n, G)$ is true.*

In our approach we employ conditional planning to build plans leading to the goal[1]; search for solution plans is performed in the space of state-sets.
The interleaving of planning and execution is achieved by splitting the planning process in several planning episodes which are alternated to execution (see [11], [5], [13], [6], [1], [7])).
We divide the monolithic planning process into smaller planning episodes by acting along two dimensions:

- *early termination heuristics (ETH)* cause interruption of a planning episode *before* a solution plan is found (see [6], [7]);
- *assumptions* about the current state-set reduce the size of the search space, simplifying the planning process as well.

A great deal of early termination heuristics have been investigated by the researchers. We distinguish between early termination *rules*, which preserve the soundness and the completeness of the control system of the robot[2] (e.g. the viable plan termination rule and the forced plan termination rule in [11]), and early termination *heuristics*, which don't preserve them (e.g. the resource-based termination heuristics in [6]).
Assumptive planning is realized by means of the *state selection function* which is applied to the current state-set at the beginning of each planning episode.

[1] Conditional plans prescribe the action to be performed according to every history of percepts that can be encountered at execution time ([11]). For example, "*reach-elevator*; if *elevator-present* then *use-elevator* else *use-stairs*" is a fragment of a conditional plan for reaching the ground floor.

[2] w.r.t. the goal specified and the environmental model.

Definition 5. *The* state selection function *is a function* $sel: \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ *used to select, within the given state-set, those states which are deemed relevant with respect to the current situation.*
A function sel is a state selection function *iff* $\forall I\,(I \subseteq S \Rightarrow sel(I) \subseteq I)$.

A general scheme of an *rfa* implementing interleaved planning and execution by means of both early termination heuristics and assumptive planning can be given in algorithmic form as follows:

Algorithm 1 (et-CSEL). *Given:*

- *a pefa* $<A, S, P, effect, sense, I>$;
- *a state of state-sets G (the goal of the agent);*
- *a state selection function sel;*
- *a function* et–plan *that, given the current state-set J, the current goal G and the effect and sense functions, returns a conditional plan which may be either a solution plan or a partial plan (if an early termination heuristic was applied);*
- *the relation* $satisfies$;
- *a function* first *which returns the first action of a conditional plan;*

compute:

1. *Let* $p =$ *current percept*
2. $I = sense(p, I)$
3. *if* $satisfies(I, G)$ *then terminate*
4. $J = sel(I)$
5. $CPlan = \text{et–plan}(J, G, effect, sense)$
6. *Let* $a = \text{first}(CPlan)$
7. $I = effect(a, I)$
8. $output = a$
9. *goto 1*

The algorithm performs continuous selection ([11]): assumptions and plans are recomputed from scratch after the execution of every action.
The drawback of this approach is that the total conditional planning cost may be much higher than in an algorithm in which replanning occurs only when strictly needed. However, in domains with incomplete information, in which the knowledge carried by the percepts plays a fundamental role, the tight interleaving of planning and execution imposed by Algorithm 1 can have significant benefits.
The planning algorithm implementing et–plan is specified as follows:

Algorithm 2 (et–plan). *Given:*

- *a state-set I representing the current state-set;*
- *a state of state-sets G representing the goal of the agent;*
- *the effect and sense functions;*
- *a relation* check–ETH *that checks whether early termination heuristics can be triggered;*
- *a function* nondet *that performs non-deterministic choice;*
- *the set A of possible actions and the set P of possible percepts;*

compute:

1. *if* $satisfies(I, G)$ *then return*
2. *if* $\text{check–ETH}(I)$ *then return*
3. $a = \text{nondet}(A)$
4. $I' = effect(a, I)$
5. *for every* $p \in P$ *do:*
6. $I'' = sense(p, I')$
7. $\text{et–plan}(I'', G, effect, sense)$

3 Interleaving Planning-while-Learning and Execution

The basic framework (the *pefa* framework) that we have presented so far makes it possible to cope with environments about which only incomplete information is available.
Three forms of incomplete information can be handled:
- partial knowledge about the initial state of the world (represented by the set I);
- partial knowledge about the behaviour of the environment (represented by the *effect* function);
- unreliability of sensory information (represented by the *sense* function).

In many cases, the performance of the agent might be improved by tuning the description of the environment's behaviour at *run-time*. This contrasts with the a-priori specification of the *effect* function given in the *pefa*.
The idea behind *planning-while-learning* is that the agent is given a set of possible behaviours of the environment before interaction starts; while interaction with the world progresses, the agent uses the information available through sensors to select, from the initial set, the behaviour which best fits the experienced behaviour of the environment.
In order to specify a set of possible environment's behaviours we introduce a **planning-while-learning environmental finite state automaton** or *plefa*, defined as a tuple $<A, S, P, E, update, sense, F, I>$ where:
- A, S, P, $sense$, I are defined like in *pefa*;
- $E \subseteq \mathcal{P}(S)^{A\times\mathcal{P}(S)}$ is the set of the possible behaviours of the environment;
- *update* is a function $\mathcal{P}(E) \times \mathcal{P}(S) \times A \times \mathcal{P}(S) \to \mathcal{P}(E)$ that updates the set of possible behaviours of the environment;
- $F \subseteq E$ is the set of the possible initial behaviours of the world;

Example 3. Suppose that our robot is put in the maze depicted in Fig. 1(c). Besides the specifications of Example 2 the cells with the grid can either prevent robot's crossing or make the robot advance of two cells instead of just one or have no special effect. The behaviour is unknown, but it is fixed over time and is the same for all the cells. The *plefa* for this setting may be described as follows:
- P, A, S, I and $sense$ are defined like in Example 2;
- $E = F = \{b_1, b_2, b_3\}$, where b_i are the effect functions corresponding to the behaviours described above.

The definitions that follow describe the concept of solution to a problem instance and the relation among *plefa*, *pefa* and *efa*.

Definition 6. *The effect function* w.r.t. a set of behaviours F *is a function* $effect_F\colon A \times \mathcal{P}(S) \to \mathcal{P}(S)$ *which describes the consequences of an action from a state-set according to* every *behaviour belonging to the given behaviour-set:* $\forall I \subseteq S\ \forall a \in A\ \ effect_F(a, I) = \bigcup \{I' \mid \exists b\ (b \in F \wedge b(a, I) = I')\}$.

Definition 7. *A* state-behaviour set tracker (sbt) *for a plefa* $<A, S, P, E, update, sense, F, I>$ *and an action-percept history* $\alpha = \{a_1, p_1, a_2, p_2, \ldots, p_n, a_n\}$ *is a robot finite automaton* $<P, B, A, int, ext, b_1>$ *where:*
- $B = \{<s, e> \mid s \in \mathcal{P}(S) \wedge e \in \mathcal{P}(E)\}$;
- $int(p_i, b_i) = <s_{i+1}, update(e_i, s_i, a_i, s_{i+1})>$, *where* $s_{i+1} = sense(p_i, effect_{e_i}(a_i, s_i))$ *and* $b_i = <s_i, e_i>$;
- $ext(b_i) = a_i$;
- $b_1 = I$.

Definition 8. *A plefa* $< A, S, P, E, update, sense, F, I >$ *is consistent with a pefa* $<A, S, P, effect, sense, I>$ *iff, for every efa consistent with pefa, the action-percept history* $\alpha = \{a_1, p_1, a_2, p_2, \ldots, a_n, p_n\}$ *of the system* $< rfa, efa >$ *makes the state-behaviour set tracker* $sbt(plefa, \alpha)$ *compute* $e_k = \{effect\}$ *for an index* $k \leq n$ *when applied to the system* $<sbt, efa>$.

Definition 9. *A plefa* $< A, S, P, E, update, sense, F, I >$ *is consistent with an efa* $<A, S, P, do, see, s>$ *iff there exists a pefa such that:*
1. *the plefa is consistent with the pefa, and*
2. *the pefa is consistent with the efa.*

Definition 10. *An rfa is a* solution *to a problem instance* $<plefa, G>$ *iff, for every efa consistent with plefa, given the action-percept history* $\alpha = \{a_1, p_1, a_2, p_2, \ldots, a_n, p_n\}$ *of the system* $<rfa, efa>$*, the state-behaviour set tracker* $sbt(plefa, \alpha)$ *in system* $<sbt, efa>$ *computes* s_n *s.t.* $satisfies(s_n, G)$ *is true.*

The conditional planning approach that we use in our extended framework (the *plefa* framework) performs a search in the space of the couples $<$*state-set*, *behaviour-set*$>$.
The fact that the behaviour-set is part of the search space results in being it possible for the agent to build plans which take into account also the information gained about the environment's behaviour: early identification of the actual behaviour is essential for a successful interaction with the world, since sequences of actions may exist which lead to the goal for some possible behaviours but make the goal unreachable for other possible behaviours.
We represent the evolution of the world during interaction by means of a **state-behaviour set graph**.

Definition 11. *A* state-behaviour *set graph is a directed bipartite graph with effectory nodes and perceptory nodes.*
An effectory node is a tuple $<$ state-set, behaviour-set $>$ *with action-labeled arcs leading away from it. Each action arc labeled with action* a *connects an effectory node* $<S_1, B_1>$ *to a perceptory node* $<S_2, B_1>$ *iff* $S_2 = effect_{B_1}(a, S_1)$ *where* $effect_{B_1}$ *is the effect function w.r.t.* B_1.
A perceptory node is a tuple $<$ state-set, behaviour-set $>$ *with percept-labeled arcs leading away from it. Each percept arc labeled with percept* p *connects a perceptory node* $<S_1, B_1>$ *to an effectory node* $<S_2, B_2>$ *iff* $sense(p, S_1) = S_2$.
An effectory node $<S_1, B_1>$ *is connected to another effectory node* $<S_3, B_2>$ *through a perceptory node* $<S_2, B_1>$ *iff* $B_2 = update(B_1, S_1, a, S_3)$ *where* a *is the action labeling the arc from* $<S_1, B_1>$ *to* $<S_2, B_1>$.

In the *plefa* framework the interleaving of planning-while-learning and execution is obtained by means of early termination heuristics and assumptions.
Assumptions can be made either about the current state-set, like in the *pefa* framework, or about the current behaviour-set. In fact, the current behaviour-set can be quite large and it can contain elements which are not likely to represent the actual behaviour of the environment.
Planning using a large behaviour-set can be very time consuming, therefore making appropriate assumptions about which behaviours are more likely to represent the actual environmental behaviour can result in a relevant performance improvement.

Definition 12. *The* behaviour selection function *is a function* $sel_b \colon \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ *used to select, among a set of possible behaviours, those which are considered relevant with respect to the current situation.*
A function sel_b *is a* behaviour selection function *iff* $\forall F (F \subseteq E \Rightarrow sel_b(F) \subseteq F)$.

We give now a simple algorithm for implementing interleaved planning-while-learning and execution by means of both early termination heuristics and assumptive planning:

Algorithm 3 (pl-CSEL). *Given:*
- *a plefa* $<A, S, P, E, update, sense, F, I>$;
- *a state of state-sets* G *(the goal of the agent);*
- *state and behaviour selection functions* sel *and* sel_b;
- *a function* et–plan_{pl} *that, given the current state-set* J*, the current behaviour-set* C*, the current goal* G *and the update and sense functions, returns a conditional plan;*
- *the relation* $satisfies$ *and the function* first *(see Algorithm 1);*

compute:
1. *Let* $p =$ *current percept*
2. $I = sense(p, I)$
3. *if* $satisfies(I, G)$ *then terminate*
4. $J = sel(I)$
5. $C = sel_b(F)$
6. $CPlan = \text{et–plan}_{pl}(J, C, G, update, sense)$
7. *Let* $a = \text{first}(CPlan)$
8. $I' = effect_F(a, I)$
9. $output = a$
10. *Let* $p =$ *current percept*
11. $I' = sense(p, I')$
12. $F = update(F, I, a, I')$
13. $I = I'$
14. *goto 3*

The planning algorithm implementing et–plan_{pl} is specified as follows:

Algorithm 4 ($\text{et–plan}_{\mathbf{pl}}$). *Given:*
- *a state-set* I *representing the current state-set;*
- *a state-set* F *representing the current behaviour-set;*
- *a state of state-sets* G *representing the goal of the agent;*

– *the update and sense functions;*
– *a relation* check–ETH *and a function* nondet *(see Algorithm 2);*
– *the set A of possible actions and the set P of possible percepts;*

compute:

1. *if* $satisfies(I, G)$ *then return*
2. *if* check–ETH(I) *then return*
3. $a = \mathrm{nondet}(A)$
4. $I' = \mathit{effect}_F(a, I)$
5. *for every* $p \in P$ *do*
6. $I'' = sense(p, I')$
7. $F' = update(F, I, a, I'')$
8. $\mathrm{et\text{–}plan}_{pl}(I'', F, G, update, sense)$

3.1 Theoretical Properties of the *plefa* framework

We briefly introduce the theoretical properties of the algorithms of the *plefa* framework, namely Algorithms 3 and 4. Sufficiency conditions for the soundness and the completeness of our *rfa* w.r.t. a given problem instance <*plefa*, *G*> will be presented. Similar results regarding the *pefa* framework and the control system built upon Algorithms 1 and 2, together with proofs, can be found in [11].
Note that we restrict our attention to *semistatic environments*, where the world does not change sensibly during deliberation.

Definition 13. *An rfa is* sound *w.r.t. a problem instance <plefa, G> iff for each efa consistent with the plefa,* if *system <rfa, efa> terminates at time t_n,* then *s_n makes true $satisfies(s_n, G)$, where s_n is intended in the sense of Definition 7.*

Definition 14. *An rfa is* sound w.r.t. the *plefa* framework *iff, for any problem instance Π, the rfa is sound w.r.t. Π.*

Definition 15. *Given a conditional plan P and a plefa, we define the* overlay *of P onto the plefa as the state-behaviour set graph corresponding to the possible traces of P's execution according to the environmental model specified by the plefa.*

Definition 16. *Given a conditional plan P and a problem instance < plefa, G>, we say that* P goes from I to G *iff, in the overlay of P onto the plefa, each fringe node T makes true $satisfies(T, G)$.*

Definition 17. *An rfa is* complete *w.r.t. a problem instance <plefa, G> iff* if *there exists a conditional plan P going from I to G* then *the rfa is a solution.*

Definition 18. *An rfa is* complete w.r.t. the *plefa* framework *iff, for any problem instance Π, the rfa is complete w.r.t. Π.*

Definition 19. *Given a problem instance $<plefa, G>$, a state selection function sel is* weak *iff* $\forall f \in E\ \forall s \in I\ (\exists r \in S(\mathrm{path}_f(s, r) \wedge \neg\mathrm{path}_f(r, s))) \Rightarrow s \in sel(I)$, *where* path_f *denotes that there exists an action-labeled path of length 1 in the partial state-behaviour set graph, using $\{f\}$ as the initial behaviour set, from state s to state r.*

Definition 20. *Given a problem instance* $<plefa, G>$*, a behaviour selection function* sel_b *is* weak *iff* $\forall s, r \in I \ (\exists f \in E(\text{path}_f(s,r) \wedge \neg \text{path}_f(r,s))) \Rightarrow f \in sel_b(F)$.

Definition 21. *Given a problem instance* $<plefa, G>$*, there is* effectory expansion *iff* $\exists f \in E \ \ \exists a \in A \ \ \exists I \subseteq S \ \ |f(a,I)| > |I|$.

Definition 22. *In a sense similar to [14] we say that the update function employs* monotonic learning *iff* $\forall F \subseteq E \ \ \forall I, I' \subseteq S \ \ \forall a \in A$ $update(F, I, a, I') \subseteq F$.

Theorem 1. *Algorithm 3 (*pl-CSEL*) is sound and complete w.r.t. the plefa framework under the following conditions:*
1. et–plan$_{pl}$ *is implemented with breadth-first search*[3]*;*
2. *the initial state set is finite;*
3. *the state selection function sel is* weak*;*
4. *the behaviour selection function* sel_b *is* weak*;*
5. *there is no effectory expansion;*
6. *learning is monotonic.*

Proof. The theorem was proved for the pefa by Nourbahsh (see Theorems 5.2, 6.2 and 6.3 in [11]). The result can be extended to the plefa by noting that, under condition 6, the set of possible environmental behaviours eventually converges to a singleton because of the definitions of solution to a problem instance and of consistent efa. □

4 Related Work

Nourbakhsh [11], [12]) deals with the interleaving of planning and execution in much the same way as we did in Sect. 2. He introduces also abstract planning, in which a partial order of *pefa*'s is given, each corresponding to a different abstraction level. No attempt is made to learn the behaviour of the environment.
Safra and Tennenholtz ([14]) introduce a framework for planning while learning that they employ for a computational study on the subject. Their framework allows to specify a set of possible behaviours, of which only one element at a time can be selected as the possible actual behaviour of the environment. It is easy to show that their framework is a special case of our *plefa* framework.

5 Conclusions

We have presented a framework for interleaving planning-while-learning and execution and have introduced sufficiency conditions for the soundness and the completeness of the control system of an agent built upon our framework. Though supported by the encouraging results of Nourbakhsh's work, the *plefa* framework needs empirical testing in order to assess its advantages over the other approaches. Future work will be focused on the introduction of abstract planning and inductive learning in our framework.

[3] and using the Viable Plan Termination Rule ([11]).

6 Acknowledgments

Alessandro Provetti provided invaluable constant support throughout the whole writing of this paper. Illah Nourbakhsh gave me precious suggestions on a number of studies related to the present work and was very patient in replying to many questions about his framework. This research was partially supported by *Progetto cofinanziato (ex 40%) MURST "Agenti Intelligenti: interazione ed acquisizione di conoscenza"*.

References

[1] Marcello Balduccini and Gaetano A. Lanzarone. Autonomous semi-reactive agent design based on incremental inductive learning in logic programming. In Wiebe Van der Hoek, Yves Lesperance, and Rich Scherl, editors, *Logical Approaches to Agent Modeling and Design, Proceedings of the ESSLLI'97 Symposium*, pages 1–12, 1997.

[2] Chitta Baral and Son Cao Tran. Relating theories of actions and reactive control. *Transactions on Artificial Intelligence*, 2:211–271, 1998.

[3] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[4] Marco Dorigo. Editorial introduction to the special issue on learning autonomous robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(3):361–364, 1996. Part B.

[5] R. Fikes. Monitored execution of robot plans produces by strips. Technical Report 55, Stanford Research Institute, 1971.

[6] R. A. Kowalski. Using meta-logic to reconcile reactive with rational agents. *Meta-logic and Logic Programming*, pages 227–242, Jan 1995.

[7] John E. Laird and Paul S. Rosenbloom. Integrating execution, planning, and learning in soar for external environments. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 1022–1029, 1990.

[8] Pattie Maes. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*. MIT Press, 1990.

[9] Pattie Maes and Rodney A. Brooks. Learning to coordinate behaviours. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 796–802, 1990.

[10] David E. Moriarty, Alan C. Schultz, and John J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276, 1999.

[11] Illah Nourbakhsh. *Interleaving Planning and Execution for Autonomous Robots*. Kluwer Academic Publishers, 1997.

[12] Illah Nourbakhsh. Using abstraction to interleave planning and execution. In *Proceedings of the Third Biannual World Automaton Congress*, 1998.

[13] E. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[14] S. Safra and M. Tennenholtz. On planning while learning. *Journal of Artificial Intelligence Research*, 2:111–129, 1994.