DOI: 10.15439/2016F351
ACSIS, Vol. 8. ISSN 2300-5963

A General Method of the Hybrid Controller Construction for Temporal Planning with Preferences

Krystian Adam Jobczyk
University of Caen
Laboratoire de Automatique, Informatique, Instrumentation et Image
Marechal Juin 6, 14000 Caen, cedex 5
E-mail: krystian.jobczyk@unicaen.fr

Antoni Ligęza
AGH University of Science and Technology
Department of Applied Computer Science
al. Mickiewicza 30, 30-059 Krakow
E-mail: ligeza@agh.edu.pl

***Abstract*—This paper is aimed at presenting some general construction method of the hybrid plan controller for some task of temporal planning with preferences. This construction is multi-stage and it begins with a description of a chosen robot environment and its plan in some extended version of Linear Temporal Logic. This description is later transformed to the appropriate preferential Büchi automaton. In the same way, the real plan performing by the robot is encoded by the similar automaton. Finally, both automata are exploited to construct its product automaton, which is later described in** PROLOG.

***Index Terms*—the hybrid plan controller, temporal planning, preferences, the robot motion environment, PROLOG, automata, Linear Temporal Logic, Halpern-Shoham logic**

I. Introduction

A *Plan Controller* constitutes a machine (sometimes only abstract) suitable for controlling how different tasks are performed in comparison with the initially developed plans or schedules. The plan controller construction is often a multi-stage activity and it begins with a description of the robot environment and tasks in the appropriate formal language. This description should be later translated into the appropriate automata which encode the initial part of information in terms of automaton states and admitted transitions between them. Such controllers[1] satisfy many different features, and – depending on the proposed approach to the robot's motion representation – are usually rendered in terms of Linear Temporal Logic (LTL), but may be also rendered in terms of the motion description language [9] or of the control and computation language [14].

An interesting approach to the plan controller construction was presented in [4], [20], which is based, however, on the known idea of the environment triangulation. The classical approach to the controller construction leads just from a triangulation of the robot's/agent's environment and the appropriate finite transition system by a representation of this environment in terms of LTL. In the next step, the LTL-formulas are represented by the appropriate Büchi automaton[2]. Its construction is later complemented by the construction of the appropriate product automaton for LTL-specification and for the considered transition system – suitable for a grasping of the basic dynamism in the robot's environment. This product automaton ensures acceptance of certain desired transitions only, see: [4], and it forms a 'core' of the hybrid plan controller representation.

[1]They are often called "hybrid controllers" as they join different automaton types and they refer to different features and 'entities' such as plans and the motion environements.

The method of the Büchi automaton construction alone for a use of different planning situations was presented in [6] and – w.r.t. formulas of some extended LTL — earlier in [21], [22]. The applicability of the formalism of temporal logic for specification of the robot's motion environment is a commonly known and widely discussed fact; see for example: [1], [2]. Indeed, such systems as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) have a satisfying expressive power to give a relatively detailed description of components of robot's motion planning such as action sequencing or objectives, the properties of the robot's environments. This utility of LTL was suggestively demonstrated in [2]. The robot's environments specification in terms of temporal logic constitutes the first fundamental step for planner and plan controller construction.

We intend to extend these recent approaches in two ways: we introduce an additional preferential component to the considered transition system and we extend the language for the robot's environment specification by introducing Halpern-Shoham logic (HS) – introduced in [7] – with single operators $\langle D\rangle$ and $\langle L\rangle$ for relations: 'during' and 'later' (*resp.*). Such a radical restriction of HS is dictated by formal requirements of the conversion to the automaton, which can be realized in this case.

Motivation. Majority of approaches to plan controlling of the robot behavior in polygonal environment – such as the presented in [1], [2], [4], [20] – discuss this issue rather in a form of extended outlines and without (often needed) technical details. By contrast, some formal papers such as [21], [22],

[2]This useful formal tool was invented in 1962 and described in [**?**].

[6] are aimed at presenting a purely meta-logical face of the construction of automata in a language of temporal logic. In result, there is no real coincidence between these approaches. Moreover, no of them take into account preferential aspects of planning, which introduce a portion of 'rationality' to performing tasks and, its management and controlling.

These lacks constitute a main motivation factor of this paper investigations. The next motivation factor stems from an additional lack of some correlation between research on the real expressive power of temporal logic systems – such as Halpern-Shoham logic in [18], [19] – and some engineering tendency to exploit temporal logic systems in (almost) all possible contexts and without restrictions.

Against this state of the art, we intend to propose such a new preferential extension of the current approaches to the plan controller construction for a robot in polygonal environment which respects last arrangements about temporal logic and its expressive power. This intention determines a choice of the appropriate subsystems of Halpern-Shoham logic: its restriction to the modal operator $\langle D\rangle$ and $\langle L\rangle$ for representation "during' and 'later'-relations (*resp.*). In fact, these operators can be effectively transformed to the appropriate automata in the light of recent observations from [19], [17]. Some conceptual background of the automaton construction for combined formulas of some multi-valued extension of HS-logic has been presented in [13], [10].

Moreover, the authors of this paper give venture of their enthusiasm with respect to some utility of the proposed construction in different areas of utility of temporal logic systems: in engineering or – for example – in business processes and their management. Some applicability of temporal logic systems for engineering has been recently discussed in [12], [11], for a use of business management – in [15], [16]. Last, but not least, expected future implementation of the automaton construction in languages of a declarative paradigm such as PROLOG or ASP forms some additional motivating factor of this paper's analysis.

Objectives of the paper. According to these motivation factors, rendered above, objectives of this paper – in a chronological order – are the following :

1) proposing a new preferential extension of the concept of the hybrid plan controller– based on product automata,
2) construction of hybrid plan controller for a robot performing task in a polygonal environment in the block world,
3) an outline of the PROLOG-representation for some fragments of the constructed plan controller.

We also associate to these main objective some additional goal to extend the used specification language from LTL to LTL extended by Halpern-Shoham logic with $\langle D\rangle$ and $\langle L\rangle$-operators, symb. HS^D.

Organization of the paper. This paper is organized as follows. In section 2 we present a terminological background of the analysis. In section 3 we present the main problem of the paper analysis and we give a general algorithm of our hybrid controller specification. Section 4 forms the main conceptual part of the paper, where we describe in details the steps of the controller construction *via* the Büchi automaton for LTL and for the considered transition system. In section 5 we briefly describe the implementation area of this construction and HS logic with operators: $AB\bar{A}\bar{B}$. In section 6 we formulate conclusions and some remarks on the future research direction.

II. Preliminaries

Before moving to main paper body, we present a terminological framework of this paper analysis, introducing a new concept of *preferential automata* and *preferential transition system*. The recalled definitions of a (finite) transition system and a Büchi automaton are incorporated from [7].

Definition 1. . *A (finite) transition system FTS is a n-tuple $FTS = (W, W_0, Act, Tran, \Pi, Obs)$, where:*

1) *W is the finite set of states (worlds),*
2) *$W_0 \subseteq W$ is the distinguished set of initial states,*
3) *Act denotes the set of possible actions,*
4) *$Tran : W \times Act \mapsto W$ is a transition function, i.e such a total function that returns the next stage for a given state an an action,*
5) *Π denotes the set of possible observations,*
6) *$Obs : W \mapsto \Pi$ is the observability function, which returns the observable part of the current state.*

We define an *execution* on FTS as an infinite sequence of states $w_0, w_1, \ldots$, such that $w_0 \in W_0$ and $w_{k+1} = Tran(w_k, a)$ for some action $a \in Act$. The observable part of the execution will be called a *trace*.

Definition 2. *A Büchi automaton is a tuple $A = (\Sigma, S, S_0, \rightarrow, \rho, \mathcal{F})$, where:*

1) *Σ is the alphabet of the automaton,*
2) *S is the set of states of the automaton,*
3) *$S_0 \subseteq S$ is the set of initial states of the automaton,*
4) *$\rho : S \times \Sigma \mapsto 2^S$ is the transition function of the automaton and*
5) *$\mathcal{F}$ is the set of accepting words.*

We expand this definition to a definition of preferential Büchi automaton by specification of the set of accepting words by introducing some degrees/parameters α's from an interval $[0, 1]$. The role of them is to measure a *degree of a preference* of the accepting words from F, indexed by such an α.

Definition 3. *A A Preferential Büchi automaton is a tuple $A = (\Sigma, S, S_0, \rightarrow, \rho, \mathcal{F}, \alpha_1, \alpha_2 \ldots)$, where:*

1) *Σ is the alphabet of automaton,*
2) *S is the set of states of automaton,*
3) *$S_0 \subseteq S$ is the set of initial states of automaton,*
4) *$\rho : S \times \Sigma \mapsto 2^S$ is the transition function of automaton and*
5) *$\mathcal{F}^\alpha$ is the set of accepting words with associated α-degree preferences.*

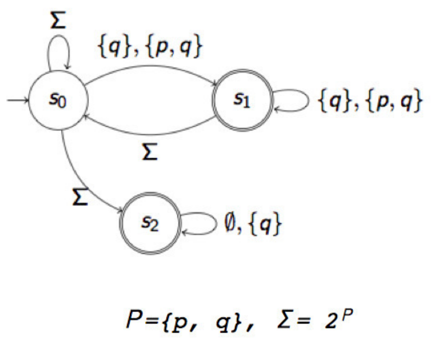

Fig. 1. Fragment of the Büchi automaton with states s_0, s_1 and s_2 over an alphabet Σ.

6) $\alpha_1, \alpha_2, \ldots$ *are values of functions* $f :\mapsto \mathcal{F}^{\alpha} \to [0,1]$ *called preferences.*

We assume that a set of α degrees is finite as they index accepting words from set F^{α}. Naturally, F^{α} should be finite as a a set of accepting words of a finite automaton.

For such a defined automaton we define a *run* of $\mathcal{A}$ (on an infinite words $a_0, a_1, a_2, \ldots$) is an infinite sequence of such states $s_0, s_1, \ldots \in S^{\omega}$ that $s_0 \in S_0$ and $s_{i+1} \in \rho(s_i, a)$. We say that a run r is *accepting* iff a set $\{s | s$ occurs in r infinitely often $\} \cap F \neq \emptyset$. If F is finite, this general condition means that there exists at least one state s that occurs in a run r infinitely often.

Linear Temporal Logic (LTL).

Syntax. Bi-modal language of LTL is obtained from standard propositional language (with the Boolean constant $\top$) by adding temporal-modal operators such as: *always in a past* (H), *always in a future* (G), *eventually in the past* (P), *eventually in the future* (F), *next and until* ($\mathcal{U}$) and *since* ($\mathcal{S}$) – co-definable with "until". The set FOR of LTL-formulas is given as follows:

$$\phi := \phi | \neg\phi | \phi \vee \psi | \phi\mathcal{U}\psi | \phi\mathcal{S}\psi | H\phi | P\phi | F\phi | Next(\phi) \qquad (1)$$

Some of the above operators of temporal-modal types are together co-definable as follows: $F\phi = \top\mathcal{U}$, $P\phi = \top\mathcal{S}\phi$ and classically: $F\phi = \neg G\phi$ and $P\phi = \neg H\phi$.

Semantics. LTL is traditionally interpreted in models based on the point-wise time-flow frames $\mathcal{F} = \langle T, < \rangle$ and dependently on a set of states S. In result, we consider pairs (t, s) (for $t \in T$ representing a time point and $s \in S$) as states of LTL-models. Anyhow, we often consider a function $f : T \mapsto S$ that associates a time-point $t \in T$ with some state $s \in S$ and we deal with pairs (t, f) instead of (t, s). Hence the satisfaction relation $\models$ is defined as follows:

1) $(t, f) \models G\phi \iff (\forall t^{'} > t)t^{'} \models \phi$, $(t, f) \models H\phi \iff (\forall t < t^{'})t^{'} \models \phi$.
2) $(t, f) \models F\phi \iff (\exists t^{'} > t)t^{'} \models \phi$, $(t, f) \models P\phi \iff (\exists t < t^{'})t^{'} \models \phi$.
3) $(t_1, f) \models \phi\mathcal{S}\psi \iff$ there is $t_2 < t_1$ such that $t_2, f \models \psi$ and $t, f \models \phi$ for all $t \in (t_1, t_2)$

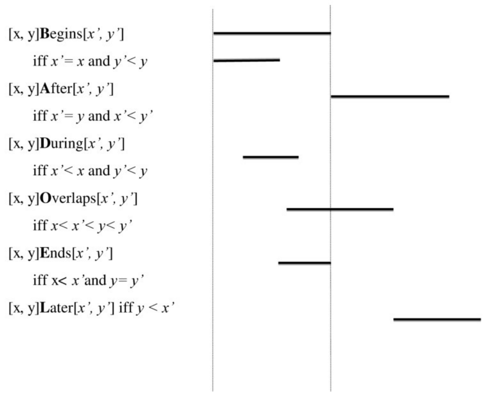

Fig. 2. Visual presentation of temporal interval relations of Allen

4) $(t_1, f) \models \phi\mathcal{U}\psi \iff$ there is $t_2 > t_1$ such that $t_2, f \models \psi$ and $t, f \models \phi$ for all $t \in (t_1, t_2)$
5) $(t_k, f) \models Next(\phi) \iff (t_{k+1}, f) \models \phi, k \in \mathcal{N}$.

Halpern-Shoham logic. HS forms a modal representation of the following temporal relation between intervals, defined by [7]: **a**fter" (or **m**eets"), (**l**ater"), **b**egins" (or **s**tart"), **d**uring", **e**nd" and **o**verlap". These relations are intuitive and their visualization can be easily found in many papers, so we omit their visual presentation, they correspond to the modal operators: $\langle A \rangle$ for **a**fter", $\langle B \rangle$ for **b**egins", $\langle D \rangle$ for **d**uring", etc. The syntax of HS entities ϕ is defined by:

$$\phi := p | \neg\phi | \phi \wedge \phi | \langle X \rangle | \langle \bar{X} \rangle, \qquad (2)$$

where p is a propositional variable and $\langle \bar{X} \rangle$ denotes a modal operator for the inverse relation with respect to $X \in \{A, B, D, E, O, L\}$ If $\phi \in \mathcal{L}$(HS), M is a model, and I is an interval in the M-domain, then the satisfaction for the HS-operators looks as follows:

$$M, I \models \langle X \rangle \phi \iff \exists I^{'} that\ IXI^{\prime}\ and\ M, I^{'} \models \phi. \qquad (3)$$

Example 1. *Some simple spatial-temporal requirements imposed on the robot's environment E can be expressed by formulas:*

• *Always if you take a block A, take B, as well:* $G(take(A) \to take(B))$,

• *For any intervals: if you take A, than you also put B:* $[D](HOLDS(take(A)) \to HOLDS(put(B)))$.

III. Problem formulation and a general algorithm of the controller construction

Assume that E is a polygonal environment of robot motion operations. All possible admitted holes of E have to be enclosed by a single polygonal chain. The motion of robot is expressed as follows:

$$x(t) \in E \subseteq R^2, u(t) \in U \subseteq R^2, u(t) \cap x(t) \neq \emptyset \qquad (4)$$

where $x(t)$ is a trajectory of robot's motion (position of a robot in a time t) in E and $u(t)$ is a control input. Non-emptiness of the above intersection $u(t)$ and $x(t)$ ensures that a controller detects the robot's trajectory. In such a framework, the goal of the paper is to give an outline of a construction of a hybrid controller that generates controllers inputs $u(t)$ for a trajectory $x(t)$ and environment with a specification given by formulas of LTL and – partially – by HS restricted to D-operator.

A general path of our controller construction looks a follows. We begin with the environment E and its triangulation. Secondly, we consider some transition system FTS to describe a basic dynamism of E. The next, we specify E in terms of LTL (ϕ-formula) and of some subsystem of HS logic. In the next construction step, we transform FTS to the appropriate Büchi automaton $\mathcal{A}_{FTS}$ for it. The similar automaton $\mathcal{A}_{LTL,HS}$ is constructed for representation of a specification of E (with a chosen point x_0) in terms of the considered temporal logic. Having these automaton, we construct some product automaton $\mathcal{A}$ to 'reconcile' the activity of both automata. Assume that some environment E of a robot and a formula $\phi \in \mathcal{L}(LTL \cup HS^{D,L})$ – describing this environment or the robot motion are given. Thus, the algorithm of the hybrid controller construction could be given as follow – as a specified version of algorithm from [4]:

Algorithm: The Hybrid Controller Construction

```
Procedure: CONTROLLER($E$, $\phi$)
    1) $\triangle \leftarrow Triangulate(E)$
    2) $FTS \leftarrow TriangulationToFTS(\triangle)$
    3) $\mathcal{A}_{FTS} \leftarrow FTS\ to\ Bchi\ Automaton$
    4) $\mathcal{A}_{LTL,HS^D} \leftarrow LTL \cup HS^D to\ Bchi\ Automaton$
    5) $\mathcal{A} \leftarrow Product \mathcal{A}_{FTS}, \mathcal{A}_{LTL,HS^D}$
    6) return: $Controller(\mathcal{A}, \triangle, \phi)$
End procedure
```

IV. Controller construction for path temporal planning with observability

A. From triangulation to the FTS system

In order to propose an exact construction of our path temporal planning controller we will represent a polygonal environment E as a finite set of partitions. One can use many methods of the initial polygonal environment's decomposition, presented in [4], [5].

The main idea of such a triangulation consists in a mapping of each point $x \in E$ to a one of the disjoint equivalence classes determined by an equivalence relation $\sim$. The natural way is to define $\sim$ as follows: $\forall x, y \in E : x \sim y \iff x, y \in \triangle$, i.e each of such an equivalence class forms a triangle, what allows us to represent a quotient set $E| \sim$ as a sum of triangles. Assume now that $T : E \rightarrow Q$ for $Q = \{\triangle_1, \dots \triangle_k\}$. Than each $T^{-1}(\triangle_i)$, for $i \in \{1, 2, \dots, n\}$ contains some states $x \in E$ and a set $\{T^{-1}(\triangle_i) | \triangle_i \in Q$, for $i \in \{1, 2, \dots, n\}\}$ of all such triangle anti-images is a desired partition of the initial motion environment E. In order to preserve a consideration generality, we do not impose any special requirements on E (concerning the temporal or spatial coverage etc.)

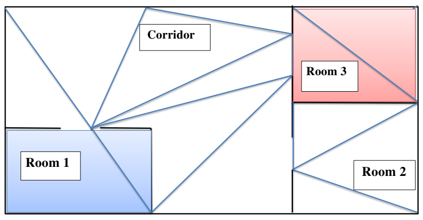

Fig. 3. An example of a triangulation of some polygonal robot environment

In this framework we can introduce a *finite transition system with preferences*– FTS, modifying a definition of of FST with observability from [4], [20] as follows:

Definition 4. *We define the finite transition system FTS with preferences as a tuple* $FTS = \langle Q, Q_0, Act, tran_{FTS}, Pref, P \rangle$*, where:*

1) *Q is the finite set of states (triangles),*
2) *$Q_0 \subseteq Q$ is the set of initial robot states,*
3) *Act is the set of actions,*
4) *$tran_{FTS} : Q \times Act \mapsto Q$ is defined as a 'move' from $\triangle_i \rightarrow \triangle_j$ iff the cells $T^{-1}(\triangle_i)$ and $T^{-1}(\triangle_j)$ share a common edge,*
5) *P is the set of objects called preferences, $Pref : Q \rightarrow P$ is a preferential map which associates to a $\triangle_1 \in Q$ some preference $pref$ such that $pref(\triangle_1) \in P$ if $P \subset [0, 1]$ and $Pref$ is a function.*

Example 2. *One can consider such a transition system with preferences as a system –depicted on a Fig. 2 with a function PREF, which satisfies a condition: $PREF(\triangle(room3)) = \frac{1}{2}$. (All triangles from a room 3 are preferable (in a sense of a robot task as places to visit) with a degree $\frac{1}{2}$).*

B. From FTS system to the Büchi automaton

It easy to observe that the finite transition system FTS naturally models a basic structure of the motion environment. It can play this role independently of a way of its presentation – in the standard form: $FTS = (W, W_0, Act, Tran, \Pi, Obs)$ or in the "triangle" tuple $FTS = \langle Q, Q_0, Act, tran_{FTS}, Pref, P \rangle$. Moreover, their structure – as it has been said – is similar to a structure of automata, so they seems to be a natural base of a construction of the required automata. In essence, (finite) automata may be considered as (finite) transition systems with labeling functions, connected with the appropriate language which delivers an alphabet as a required component of the automaton structure. Due to some practice in this area – expressed in [4], [20] – we will consider the standard representation of FTS as more suitable for an expected automaton construction.

Authors of the approach from [4], [20] recommended to an immediate use a structure of such an FTS for this construction. We only partially make use of this advise. Of course, we will base the automata construction on the FTS-structure with its states as transitions as states and transitions of the newly constructed automaton. Nevertheless, we decide for a more"descriptive" solution: we firstly describe a robot environment (FTS) in terms of LTL and we construct states of desired automaton from sets of the used formulas. For that reason, one need now a mechanism of such a description. Its presentation will be given below.

C. From LTL and $HS^{L,D}$ to its Büchi automaton

The next step of our construction will consist in a constructing of the appropriate Büchi automaton for LTL- and HS-formulas expressing the robot's motion environment E and the action sequencing. This situation is, however, not so comfortable as in the FTS-case for three purposes. First, both LTL and HS form formal languages, so their translation for the Büchi automaton should be more sophisticated. Moreover, we deal with two languages: LTL and HS of a different temporal and a modal-temporal nature. Last, but not least, HS logic (in generality) cannot be expressed by the Büchi automaton because of the enormous expressive power of this logic. Fortunately – as it was already mentioned – it was proven in [17] that the subsystem of HS with an operator D can be represented by a finite state automaton. Unfortunately, it is not clear whether there exists any extension of this subsystem with the same property. Nevertheless, it is known that a subsystem $A\bar{A}B\bar{B}$ is too strong, because ω-regular languages can be embedded in this system, but not in the inverse direction [19]. According to these remarks on the Büchi automaton construction in section 2, we begin with defining of a closure of formulas of LTL and HS^D.

Definition 5. *Let assume that $\phi \in \mathcal{L}$(LTL). We define its closure $cl(\phi)$ as follows:*

1) $\phi \in cl(\phi), \neg(\phi_1) \in cl(\phi)$, *than* $\phi_1 \in cl(\phi)$,
2) $\phi_1 \wedge \phi_2 \in cl(\phi)$ *than* $\phi_1, \phi_2 \in cl(\phi)$,
3) $A(\phi_1, \ldots, \phi_k) \in cl(\phi) \to A_{sub}(\phi_1, \ldots, \phi_k) \in cl(\phi)$ *for* A_{sub} *denoting a sub-formula of A.*

In the similar way we will define a closure for $\phi \in \mathcal{L}(\text{HS}^D)$. For a distinction of these languages and their formulas we will denote them as: ϕ_{LTL} and ϕ^D_{HS}.

Definition 6. *In accordance with the earlier statements we define an automaton $\mathcal{A}_{LTL,HS^D}$ as n-tuple:*

$$\mathcal{A}_{LTL,HS^D} = (2^{cl(\phi_{LTL} \cup \phi^D_{HS})}, \rho, S_0^{\phi_{LTL}}, S_0^{\phi^D_{HS}}, \mathcal{F}) \qquad (5)$$

where $(2^{cl(\phi_{LTL})}$ is the set of states of the automaton as the collection of all sums of (disjoint) sets of formulas in $cl(\phi_{LTL})$ and $cl(\phi^D_{HS})$, ρ is the transition and $S_0^{\phi_{LTL}}, S_0^{\phi^D_{HS}}$ are sets of initial states of automaton containing for ϕ_{LTL} and $\phi^{D,L}_{HS}$ and $F \subseteq 2^{cl(\phi_{LTL})} \cup 2^{cl(\phi^D_{HS})}$ is a set of accepting words of automaton[3]. *(resp.).*

One can expand this definition – based on [21], [22] to the definition of a preferential automaton as follows:

Definition 7. *A preferential automaton (for words of $\mathcal{L}(LTL \cup HS^{D,L})$) is defined as n-tuple:*

$$\mathcal{A}_{LTL,HS^D} = (2^{cl(\phi_{LTL} \cup \phi^{D,L}_{HS})}, \rho, S_0^{\phi_{LTL} \cup \phi^{D,L}_{HS}}, \mathcal{F}^\alpha, \alpha_1\alpha_2, \ldots) \qquad (6)$$

where $(2^{cl(\phi_{LTL} \cup \phi^{D,L}_{HS})}$ is the set of states of the automaton as the collection of all sums of (disjoint) sets of formulas in $cl(\phi_{LTL})$ and $cl(\phi^D_{HS})$, ρ is the transition,

$S_0^{\phi_{LTL} \cup \phi^{D,L}_{HS}}$ are sets of initial states of automaton containing for ϕ_{LTL} and $\phi^{D,L}_{HS}$ and $\mathcal{F}^\alpha \subseteq 2^{cl(\phi_{LTL}) \cup \phi^D_{HS})}$ is a set of accepting words of automaton, and each of $\alpha_1, \alpha_2 \ldots$ is a function $f : \mathcal{F}^\alpha \mapsto [0, 1]$.

Example 3. *Let us consider $\phi = \neg\phi_1$ for some ϕ_1 as a LTL-formula and $\psi = \langle D \rangle \psi_1$ (for some ψ_1) as our HS^D-formula.We show how to construct a Büchi automaton in a case of these formulas.*

Due to definition of a closure of the formula we obtain: $cl(\phi) = \{\phi_1, \neg\phi_1\}$ and $cl(\psi) = \{\psi, \neg\psi, \langle D \rangle\psi, \neg\langle D \rangle\psi\}$ and thus $2^{cl(\phi)} = \{\emptyset, \{\phi_1\}, \{\neq \phi_1\}, \{\phi_1, \neg\phi_1\}\}$ and $2^{cl(\psi)} = \{\emptyset, \{\psi\}, \{\neg\psi\}, \{\psi, \langle D \rangle\psi\}, \{\psi, \neg\langle D \rangle\psi\}, \{\neg\psi, \langle D \rangle\psi\}, \{\neg\psi, \neg\langle D \rangle\psi\}\}$. $N_\psi = \{\neg\phi_1, \{\phi_1, \neg\phi_1\}\}$ (because these sets contain ϕ) and $N_\psi = \{\{\psi_1, \langle D \rangle\psi_1\}, \{\neg\psi_1, \langle D \rangle\psi_1\}$ (because these sets contain $\langle D \rangle\psi_1$). Transitions ρ_{LTL} and ρ_{HS^D} are defined in such a way that sets from N_ϕ are initial in ρ_{LTL} and N_ψ are initial for ρ_{HS^D}.

D. Product automaton $\mathcal{A}_{FTS} \times \mathcal{A}_{LTL,HS^D}$

We have already defined the automaton $\mathcal{A}_{FTS}$, which describes the finite transition system and the automaton $\mathcal{A}_{LTL,HS^D}$ that represents the initial temporal logic-based specification of the motion environment. There is a need to reconcile both automata in order to construct our open-loop hybrid controller. For this purpose, it seem to be reasonable to restrict a *spectrum* of the possible transitions to these of them, which can ensure some form of observability.

For this purpose we introduce a product automaton $\mathcal{A} = \mathcal{A}_{FTS} \times \mathcal{A}_{LTL,HS^D}$ with a new transition $\to_\mathcal{A}$ between pairs: $(q_i, w_i) \to_\mathcal{A} (q_j, w_j)$. We assume that this transition holds between such pairs if and only if $q_i \to_{FTS} q_j$ and $(w_i; \pi(q_j)) \to_{LTL} w_j$. This last condition means that the last transition has an input that contains an action $\pi(q_j)$ being an observation of the newly achieved (in sense of $\to_{FTS}$) state q_j.

Definition 8. *Let $\mathcal{A}_1 = \langle \Sigma_1, S, S_0, \to_{\mathcal{A}_1}, \mathcal{F}^\alpha, \alpha_1, \alpha_2 \ldots \rangle$ and $\mathcal{A}_2 = \langle \Sigma_2, T, T_0, \to_{\mathcal{A}_2}, \mathcal{F}^\beta, \beta_1, \beta_2 \ldots \rangle$ are preferential*

[3]Let us observe that defining of $\mathcal{F}$ as accepted words determines, somehow, a set of final states by this way of state definition – just by formulas, what seems to justify $\mathcal{F}$ as a subset of $2^{cl(\phi_{LTL}) \cup cl(\phi^D_{HS})}$.

automata, than their preferential product automaton is the automaton of the form:

$$\langle \Sigma_1 \times \Sigma_2, S \times T, S_0 \times T_0, \rightarrow_{A_1 \times A_2}, \mathcal{F}^{\alpha} \times \mathcal{F}^{\beta}, \alpha_1, \alpha_2, \beta_1, \beta_2 \ldots \rangle , \tag{7}$$

where $\rightarrow$ is a product transition defined such that: $(s_i, t_i) \rightarrow (s_{i+k}, t_{i+k})$ holds for natural $i \in I$ and $1 \leq k$ if and only if $s_i \rightarrow_{A_1} s_{i+k}$ and $t_i \rightarrow_{A_2} t_{i+k}$ and $\alpha_1, \alpha_2, \beta_1, \beta_2 \in [0, 1]$ are fuzzy values expressing preference degrees of accepting words from $\mathcal{F}^{\alpha}$ and $\mathcal{F}^{\beta}$ (resp.)

(For simplicity, we will shortly write $\rightarrow$ instead of $\rightarrow_{A_1 \times A_2}$, when it will not lead to any confusion.)

E. Complementation of conditions for a product automaton

After the product automaton construction the design of an open-loop hybrid controller for motion planning reduces to the finding problem the accepting execution of this automaton. Nevertheless, this construction requires a small complementation. In fact, some components of $\mathcal{A}$ being singletons can have no outgoing transitions. In order to ensure a normal work of the automaton, we add the so-called stutter extension [8] rule, which adds a self-transition on the blocking states. More formally: for all states $s \in$ domain of $\mathcal{A}$ we define a new transition $\rightarrow_{\mathcal{A}^*} = \rightarrow_{\mathcal{A}} \cup (s \rightarrow_{\mathcal{A}} s)$, where $\rightarrow_{\mathcal{A}}$ is the transition of the automaton $\mathcal{A}$ [4], earlier defined. In such a framework it holds the following:

Theorem 1. *(adopted from [6]) An execution of FTS that satisfies the specification in terms of LTL and HS-formulas exists iff the language of A is non empty.*

We omit the proof details. For a case of LTL-specification it can be found in[7]. For a case of HS^D it follows from the existence of the automaton accepting formulas of this logic from [17].

V. Part II: Implementation

In last part of the paper we gave a theoretic outline of the hybrid controller construction – based on some product automaton. In addition, a description of the robot motion environment and the robot plan have been rendered in LTL extended by some fragment of HS-logic. In this part we intend to illustrate these ideas by proposing a concrete construction of such a controller. According to the earlier arrangements – this construction will be multi-stage and it will contain the following stages:

1) a presentation of the robot motion environment ,
2) a formal description of the environment and the plan of the robot in terms of $LTL \bigcup HS^{D,L}$,
3) a formal description of the environment and a real plan performing by the robot in terms of $LTL \bigcup HS^{D,L}$

[4]In essence we consider a projection of $\rightarrow_{\mathcal{A}}$ for the set of s-states, because $\rightarrow_{\mathcal{A}}$ works for pairs of states.

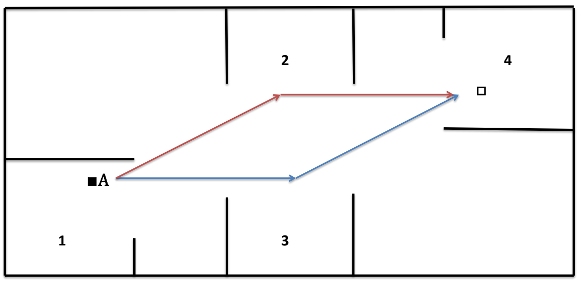

Fig. 4. The polygonal environment of the robot's motion with 4 rooms. The blue broken line illustrates the planned trajectory of the robot's move from a room no. 1 to the room no. 4. The red one illustrates the deviated trajectory of the robot's move.

4) a construction of the appropriate Büchi automata for both the cases (the first one – for a desired plan, the second one – for a real plan performing by the robot) ,
5) a construction of a product automaton built up from automata from a point (4).
6) a PROLOG-description of the product automaton in order to detect eventual discrepancies between a plan and its performing by the robot .

A. The motion robot environment and its $LTL \bigcup HS^{D,L}$-specification

Let us consider a robot, say R, in some polygonal environment with 4 rooms as depicted on a picture below. Assume that R performs a task to dislocate a black block A from a room 1 to the room 4 and put in on a block B there and the planned (preferred) move trajectory leads from the room 1 by a neighborhood of the room 3 to the room 4 (the blue line on a picture). Let also assume that our robot exchanged this trajectory for another one (marked by a red line).
Therefore, the robot motion environment and plan specification in $LTL \cup HS^{L,D}$ may be rendered as follows:

- **Plan + Preferences**:
  1) Take a block A.
  2) Move from R_1 to R_3 (more preferable) or Move from R_1 to R_4 (less preferable).
  3) If you are in R_3, move from R_3 to R_4.
  4) Go to the room R_4).
  5) Put a block A on the block B.

We can also extract the following 'behavioral' rule for the robot as a condition of an effective plan performing.

- **Condition for the plan performing/behavioral rules**:
  1) Always in a future, if you take a block A, go to the room R_3.

Finally, we should give a short description of the polygonal robot motion environment.

- **The robot environment**:
  1) Environments consists of 4 rooms,
  2) In a room R_1, block A is initially located,
  3) In a room R_4, block B is initially located,
  4) In rooms R_2, R_3 no blocks are located,
  5) In room R_4, block A is finally located,
  6) The robot motion area is always located on the left of a room R_1,
  7) The robot's motion area is always located on the right of a room R_4.

All these conditions may be regarded now in terms of $LTL \bigcup HS^{D,L}$ in the corresponding way as follows:

- **Plan + Preferences**:
  1) $Take(A)$.
  2) $Move(R_1^A, R_3) \vee Move(R_1^A, R_4)$.
  3) $HOLDS(R_3^R) \to Move(R_3, R_4)$.
  4) $Put(A)$.
  5) $HOLDS(R_4^A)$.
- **Condition for the plan performing/behavioral rules**:
  1) $G(take(A) \to \langle L \rangle go(R_3)$.
- **The robot environment**:
  1) $R_1 \wedge R_2 \wedge R_3 \wedge R_4$,
  2) $R_1^A \wedge R_4^B$ $(HOLDS_{Init}(R_1^A))$,
  3) R_4^A $(HOLDS_{Fin}(R_1^A))$,
  4) $[D](R_1 \to Left)$
  5) $[D](R_4 \to Right)$.

Let us return now the initial assumption that the robot deviated from the planned path and has chosen a red line from $R_1 to R_4$, so via R_2. We can trace this deviation for the following juxtaposition of two formal descriptions in terms of $LTL \bigcup HS^{D,L}$ for both situations.

plan of the robot	**the real plan performing**
$Take(A)$ $Move(R_1^A, R_3) \vee Move(R_1^A, R_4)$ $HOLDS(R_3^R) \to Move(R_3, R_4)$ $Put(A)$ $HOLDS(R_4^A)$	$Take(A)$ $Move(R_1^A, R_2)$ $Move(R_2^A, R_4)$ $Put(A)$ $HOLDS(R_4^A)$
behavioral rule	**behavioral rule**
$G(take(A) \to \langle L \rangle go(R_3)$	?

B. From $LTL \cup HS^L$ to the Büchi automaton

The next stage of the plan controller construction consists in a translation of $LTL \cup HS^L$-formulas to states of an automaton which usually is not given *a priori*, but it must be constructed in the appropriate way. The automaton in our case will be constructed due to [21], [22] *via* such that states are identified with subsets of closures of the $LTL \cup HS^L$-formulas. In order to illustrate this procedure let us consider

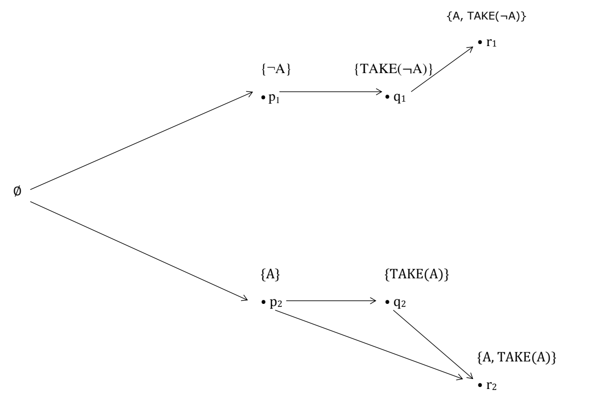

Fig. 5. Fragment of the Büchi automaton with states for a closure of the LTL-formula Take(A).

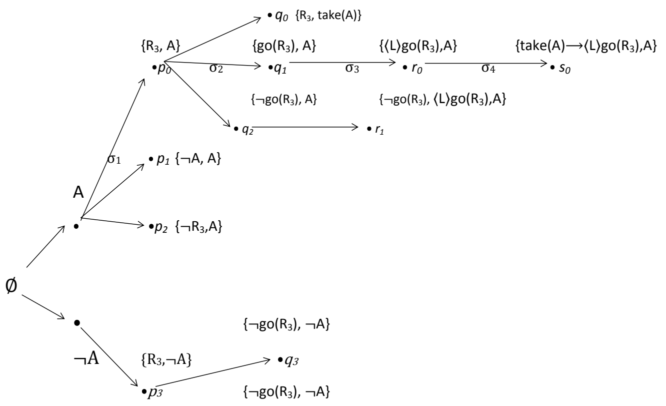

Fig. 6. Fragment of the Büchi automaton with states for a closure of the LTL-formula $Take(A) \to \langle L \rangle go(R_3)$.

a case of a single LTL-formula Take(A). Due to definition 5 from p. 5 a closure of a formula ϕ contains all of its sub-formulas and its negations. In this case we get the following sets of formulas: $\emptyset$, $\{\neg A\}, \{A\}$, $\{TAKE(\neg A)\}$, $\{A, TAKE(\neg A)\}, \{TAKE(A)\}, \{A, TAKE(A)\}$ – as depicted on a fig. 3.

The fragment of Büchi automaton for a more complicated formula $LTL \cup HS^L$-formula $take(A) \to \langle L \rangle go(R_3)$ was presented on a diagram 4.

The fragment of the Büchi automaton, say $\mathcal{A}$, for LTL-formula $Move(R_1^A, R_3)$ expressing the second step of the robot's plan is more complicated and it looks like depicted on fig.3. The same principle determines a construction way

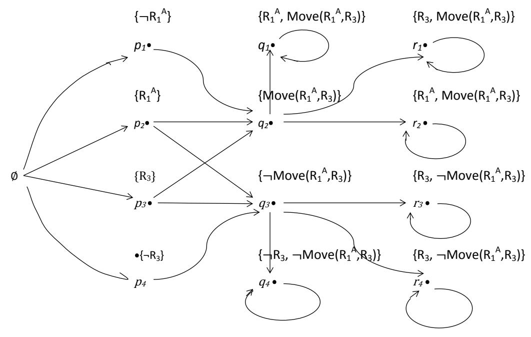

Fig. 7. Fragment of the Büchi automaton with states for a closure of the LTL-formula a$Move(R_1^A, R_3)$.

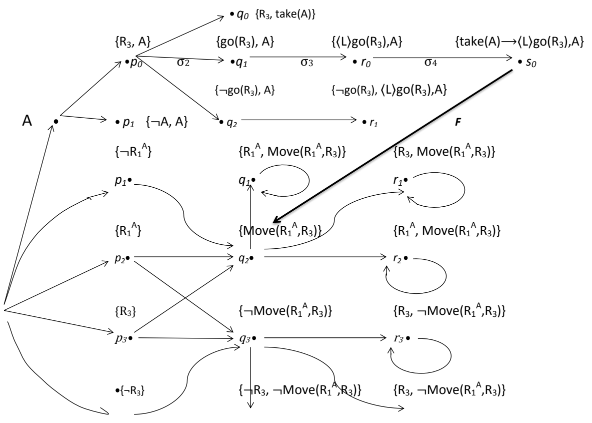

Fig. 9. Fragment of the Büchi automaton for $Move(R_1^A, R_3)$ and for a fragment of a formula $take(A) \rightarrow \langle L \rangle go(R_3)$.

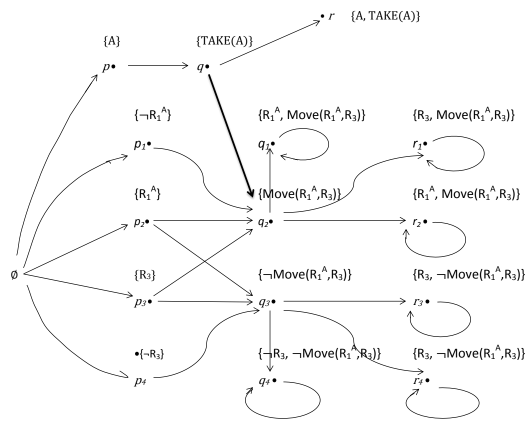

Fig. 8. Fragment of the Büchi automaton with states for a closure of the LTL-formulas $Take(A)$ and $Move(R_1^A, R_3)$.

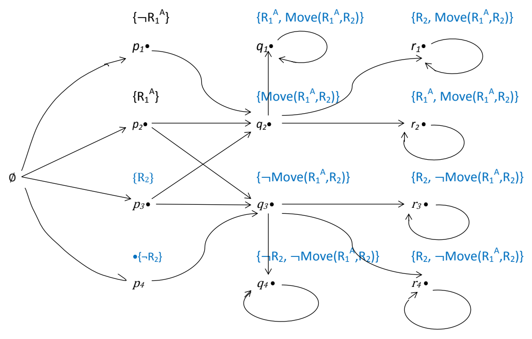

Fig. 10. Fragment of the Büchi automaton for the real task performing for the closure of LTL-formula $Move(R_1^A, R_2)$.

of the automaton fragment for $Move(R_1^A, R_3)$ – as depicted on a Fig. 5. In addition, the fragment of automaton for both formulas: $Take(A)$ and $Move(R_1^A, R_3)$ (taken together) is demonstrated on a Fig. 6. Finally, a more extended fragment of this automata for $Move(R_1^A, R_3)$ and the 'behavioral rule' of the robot $Take(A) \rightarrow \langle L \rangle go(R_3)$ – providing the plan performing – was presented on a Fig. 7.

It is not difficult to observe that a 'global size' of such automata for a complete plan of the robot and its motion environment is very large and its complete presentation would be difficult and non-suggestive. It is enough to observe that the full automaton is a composition of the appropriate fragments enriched by some 'move-lines' (the dark line on a Fig. 7) in order to connect the appropriate fragments of this automaton.

In order to detect discrepancies between the plan-automaton (for $Move(R_1^A, R_3)$ and for the 'behavioral rule' $take(A) \rightarrow \langle L \rangle go(R_3)$) with the corresponding part of automaton for the real performing of the plan, it is enough to compare both pictures. (The different states, i.e. with different formulas satisfied in them, are marked in a blue color.) Moreover, the plan-automata is larger as it also contains the transition 'branch' for the 'behavioral rule' of the robot. This branch cannot be added to the second automaton – due to observations from last section.

It has emerged that a discrepancy on the level of $LTL \cup HS^L$-formalism can be naturally reflected by its corresponding automata – as depicted on the Fig. 8. and Fig. 6.

C. *The product automaton*

As it has already been mentioned – each product automaton preserves some portion of information encoded by two automata: by the plan-automata, say $\mathcal{A}^{plan}$ and by its 'rival' for the real task performing, say $\mathcal{A}^{perf}$. As each product structure, the product automaton $\mathcal{A}^{plan} \times \mathcal{A}^{perf}$ is built up from pairs of states of the form: (s_1^{plan}, s_2^{perf}), where each $s_1^{plan} \in \mathcal{A}^{plan}$ and each $s_2^{perf} \in \mathcal{A}^{perf}$. [5] The similar pairs can be constructed with respect to transitions taken from $\mathcal{A}^{plan}$ and their equivalents from $\mathcal{A}^{perf}$ (if there are)[6].

The product automata – due to the earlier definition of preferential product automaton – $\mathcal{A}^{plan} \times \mathcal{A}^{perf}$ must have a general form:

$$\langle \Sigma_1 \times \Sigma_2, S^{plan} \times S^{perf}, S_0^{plan} \times S_0^{perf}, \to, \mathcal{F}^{\alpha} \times \mathcal{F}^{\beta}, \alpha_1, \alpha_2 \ldots \rangle \,. \tag{8}$$

Assuming that the choice of $Move(R_1, R_3)$ is preferable with a degree, say $\frac{2}{3}$ and the choice of $Move(R_1, R_4)$ – with a degree $\frac{1}{3}$ in the robot plan, the fragment of product automata $\mathcal{A} \times \mathcal{A}^{pref}$ for formulas $Move(R_1, R_3) \vee Move(R_1, R_4)$ and $Move(R_1, R_2)$ – due to definition – will have the following algebraic representation:

$$\Sigma_1 \times \Sigma_2 = \mathcal{L}(LTL \cup HS^L) \times \mathcal{L}(LTL \cup HS^L)\,,$$
$$S^{plan} \times S^{perf} =$$
$$2^{cl(Move(R_1,R_3) \vee Move(R_1,R_4)} \times 2^{cl(Move(R_1,R_2)}\,,$$
$$S_0^{plan} \times S_0^{perf} = \{\emptyset, \{R_1\}, \{\neg R_1\} \ldots\}^2\,,$$
$$\mathcal{F}^{\alpha} \times \mathcal{F}^{\beta} =$$
$$\{R_1, R_3, R_4, Move(R_1, R_3)^{\frac{2}{3}}, Move(R_1, R_4)^{\frac{1}{3}}\} \times$$
$$\{R_1, R_2, Move(R_1, R_2)\}\,,$$
$$\alpha_1 = \tfrac{2}{3}, \alpha_2 = \tfrac{1}{3}\,.$$

The diagram presentation of the product automaton $\mathcal{A} \times \mathcal{A}^{pref}$ is simple and much more suggestive. For example, a diagram of the product automaton fragment for formulas $Take(A), Move(R_1^A, R_3)$ 'produced' with $Move(R_1^A, R_2)$ would be a product of single automata from Fig.6 and Fig. 8. (Both these automata – so to speak – combined together). The preferences imposed on some transition paths must be only marked by the appropriate values such as $\frac{2}{3}$, $\frac{1}{3}$ [7].

D. PROLOG-*description of the product automaton*

The last step of our plan controller construction is to encode the product automaton – earlier described in details – in a declarative language such as PROLOG. An idea of encoding is simple: a 'status' and localization of automaton states will be described by such predicates as: final(x), initial(x), but the transitions between them may be rendered by 3-argument predicates arc(x,y,z).

In this framework the PROLOG-description of the fragment of $\mathcal{A}^{plan}$-automaton for two formulas: $Take(A) \cup Move(R_1^A, R_3)$ may be as follows:

```
initial(0).
 final(r3).   final(r1).   final(r2).
final(r3).   final(r4).   final(q4).
arc(0,p1). arc(0,p2). arc(0,p3).
arc(0,p4).
arc(p1,q2). arc(p2,p2). arc(p2,q2).
arc(p2,q3). arc(p3,q2). arc(p3,q3).
arc(p4,q3).
arc(q1,q1). arc(q2,q1). arc(q2,r1).
arc(q2,r2). arc(q3,r3). arc(q3,r4).
arc(q3,q4).
arc(r1,r1). arc(r2,r1). arc(r3,r3).
arc(r4,r4).
arc(0, p). arc(p,q). arc(q,r). arc(q,q2).
```

In the similar way one can encode the fragment of the second automaton $\mathcal{A}^{pref}$ for $Move(R_1^A, R_3)$. Since each state is determined by a set of formulas satisfied in it and most of states of this automaton is determined by formulas different from the earlier ones, one should denote the corresponding states – when needed – by capital letters: R_1, R_2, Q_1, Q_2 etc. In order to encode them in PROLOG we will represent them by $'stateR_1','stateR_2'$ etc.[8] Thus the PROLOG-description of the required fragment of $\mathcal{A}^{pref}$ looks as follows:

```
initial(0).
final(r3).   final(r1).   final(r2).
final(r3).   final(r4).   final(q4).
arc(0,p1). arc(0,p2). arc(0,P3).
arc(0,P4).
arc(p1,Q2). arc(p2, stateP2).
arc(p2,stateQ2). arc(p2,stateQ3).
arc(stateP3,stateQ2).
arc(stateP3,stateQ3).
arc(stateP4,stateQ3).
arc(stateQ1,stateQ1).
arc(stateQ2,stateQ1).
arc(stateQ2,stateR1).
arc(stateQ2,stateR2).
arc(stateQ3,stateR3).
arc(stateQ3,stateR4).
arc(stateQ3,stateQ4).
arc(stateR1,stateR1).
arc(stateR2,stateR1).
arc(stateR3,stateR3).
arc(stateR4,stateR4).
```

In order to better elucidate differences between both PROLOG-encodings, let us write in the lines which differentiate both codes:

[5]It is not completely correct, because we should pedantically state that s_1^{plan} belongs to some set S^{plan} of states of $\mathcal{A}^{plan}$, but we will omit this distinction for a simplicity and some suggestiveness of analysis.

[6]Of course, both automata $\mathcal{A}^{plan}$ and $\mathcal{A}^{perf}$ are defined over the same alphabet $\Sigma = \mathcal{L}(LTL \cup HS^{L,D})$.

[7]Naturally, these paths (with associated values) could by also distinguished among other ones in other way on a diagram representation of the automaton.

[8]This encoding follows from the fact that capital letters in PROLOG are variables.

```
arc(stateQ1,stateQ1).
arc(stateQ2,stateQ1).
arc(stateQ2,stateR1).
arc(stateQ2,stateR2).
arc(stateQ3,stateR3). arc(stateQ3,R4).
arc(stateQ3,statesQ4).
arc(tateR1,stateR1). arc(stateR2,stateR1).
arc(stateR3,stateR3).
arc(stateR4,stateR4).
```

In addition, the third line of this first code for $\mathcal{A}^{plan}$ is partially incompatible with the corresponding line of the second code. The lack of the last line of the first code in the second code is accidental, since it follows from the automaton construction for $Move(R_1^A, R_2)$ only, but the "branch" for $Take(A)$ could be also added to this fragment of $\mathcal{A}^{pref}$. Nevertheless, the branch for $Take(A) \to \langle L \rangle go(R_3)$ cannot be added to is as the robot did not respect this 'behavioral rule' in its real task performing what it is reflected by $\mathcal{A}^{pref}$.

It remains to enrich these PROLOG–descriptions by some preferential component – as it it has been made with respect to automata. Due to our convention – preferences are denoted by rational numbers from a fuzzy set $[0, 1]$ and they are associated to paths/arcs between automaton states. For simplicity of the PROLOG-representation, we can assume that they can be associated to final states of such paths/arcs. Assume, however, that we do not know how they are associated to concrete formulas or states, but we only known that each of formula can take one of values from the set $\{0, \frac{1}{2}, \frac{2}{3}, 1\}$. If we define the automaton branches for a $Take(A)$-formula by PROLOG lists, say X and Y, we also need to add a piece of information about possible fuzzy values that can be considered:

X ins $0, \frac{1}{2}, \frac{2}{3}, 1$, and Y ins $0, \frac{1}{2}, \frac{2}{3}, 1$.

These coding examples do not exhaust the list of possible ways of PROLOG-encoding, but they are used for illustration and can be extended and specified in many ways.

VI. Conclusions

It has just been demonstrated how the hybrid plan controller can be constructed beginning with a formal description of the robot motion environment. As it could be observed – we were mostly interested in a theoretic side of a construction of such a controller, putting aside its real robotic materialization. This issue could constitute a subject of further research direction.

Anyhow, it has emerged that the initial discrepancy between a plan and its real performing by a robot can be encoded at each stage of the controller construction without losing of any portion of information. In fact, the same discrepancy at the stage of the LTL-description can be transformed to the stage of the automaton construction and – finally – could be visible at the stage of its PROLOG-representation, too. One could venture a thesis that attempts with other languages of a declarative paradigm give the similar results.

Naturally, the preferential extension of automata and the whole construction of the plan controller that we have just proposed forms a kind of an 'external' extension. In fact, we have not introduced any explicit preferential language to LTL extended by HS-language with $\langle D \rangle$ and $\langle L \rangle$. It seems that this task could be feasible in some preferential extension of HS or LTL.

References

[1] M. Antonniotti and B. Mishra. Discrete event models+ temporal logic = supervisory controller: Automatic synthesis of locomotion controllers. *Proceedings of IEEE Intern. Conf. on Robotics and Automation*, 1999.

[2] F. Bacchus and F. Kabanza. Using temporal logic to express search control knowledge for planning. *Artificial Intelligence*, 116, 2000.

[3] R. Buchi. On a decision method in restricted second-order arithmetic. *Stanford University Press*, 1962.

[4] G. Fainekos, H Kress-gazit, and G. Pappas. Hybrid controllers for path planning: A temporal logic approach. *Proceeding of the IEEE International Conference on Decision and Control, Sevilla*, December:4885–4890, 2005.

[5] G. Fainekos, H Kress-gazit, and G. Pappas. Temporal logic moton planning for mobile robots. *Proceeding of the IEEE International Conference on Robotics and Automaton*, pages 2032–2037, 2005.

[6] D. Giacomo and M Vardi. Automata-theoretic approach to planning for temporally extended goals. *Proceeding of the 5th European Conference on Planning*, 1809:226–238, 2000.

[7] J. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38:935–962, 1991.

[8] G. Holzmann. *The spin model checker, primer and reference manual.* Addison-Wesley, 2004.

[9] D. Hristu-Varsakelis, M. Egersted, and S. Krishnaprasad. On the complexity of the motion description language mdle. *Proceedings of the 42 IEEE Conference on Decision and Control*, December:3360–3365, 2003.

[10] K. Jobczyk and A. Ligeza. Multi-valued halpern-shoham logic for temporal allen's relations and preferences. *Proceedings of the annual international conference of Fuzzy Systems (FuzzIEEE)*, page to appear, 2016.

[11] K. Jobczyk and A. Ligeza. Systems of temporal logic for a use of engineering. toward a more practical approach. In *Intelligent Systems for Computer Modelling*, pages 147–157, 2016.

[12] K. Jobczyk, A. Ligeza, and K. Kluza. *Proceedings of ICAISC'16*, LNAI II:to appear, 2016.

[13] K. Jobczyk and J. Ligeza, A. nad Kaczmarczuk. Fuzzy-temporal approach to the handling of temporal interval relations and preferences. *Proceedings of INISTA2015*, pages 1–8, 2015.

[14] E. Klavins. A language for modelling and programming cooperative control systems. *Proceedings of the 42 IEEE Conference on Robotics and Automaton, New Orleans*, April, 2004.

[15] M. Mach-Krol. Perspectives of using temporal logics for knowledge management. *Proceedings of FedCsIS*, 49:935–938, 2012.

[16] M. Mach-Krol. Perspects of using temporal logics for knowledge management. *ABICT*, 49:41–52, 2012.

[17] J. Marcinkowski and J. Michaliszyn. The last paper on the halpern-shoham interval temporal logic. 2010.

[18] L. Maximova. Temporal logics with operator 'the next' do not have interpolation or beth property. In *Sibirskii Matematicheskii Zhurnal*, pages 109–113, 32(6)1991.

[19] A. Montanari and P. Sala. Interval logics and omegab-regular languages. *LATA*, LNCS:431–444, 2013.

[20] P. Tabauda and G. Pappas. From discrete specification to hybrid control. *Proceedings of the 42IEEE Conference on Decision and Control*, 2003.

[21] M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. *Proceedings of the 1st Symposium on Logic in Computer Science*, June:322–331, 1986.

[22] M. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.