Behavior Synthesis for an ATLAS Humanoid Robot from High-level User Specifications

I'm not sure if "behavior" is descriptive enough. How about "Synthesis of Executable State Machines for ..." ?

Spyros Maniatopoulos*, Philipp Schillinger†, Vitchyr H. Pong*, David C. Conner‡, and Hadas Kress-Gazit*

***Abstract*— In this paper we ...**

Body of my TODO example

LIST OF TODOS, FIXES, OPEN ISSUES

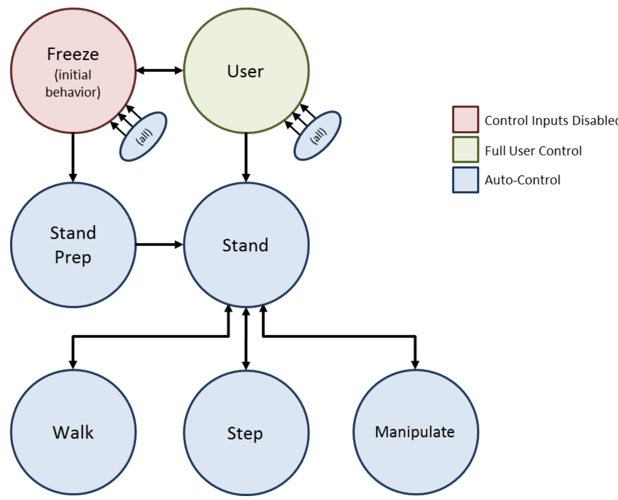

Fig. 1:

Placeholder! Create two subfigures depicting a simple control mode TS and ATLAS doing something.

I. INTRODUCTION

Contributions (brain dump):

- Partial to full specification
  - Most intuitive from the users point-of-view
  - Limited message size over bad comms (send partial specification $\rightarrow$ compile and synthesize onboard)
- Multi-paradigm specification (objectives and initial conditions from user, topology/modes, preconditions, task)

  Also consider mutex for grounding conflicts?
- Generalization of activation–completion paradigm [1]
- Integration with FlexBE and ROS
- Experimental validation on ATLAS

II. PRELIMINARIES

A. *ATLAS Humanoid Robot*

B. *Team ViGIR's Approach to High-level Control*

Action preconditions $Pre(a) \in 2^{(\mathcal{A} \cup \mathcal{M})}$

C. *Linear Temporal Logic and Reactive LTL Synthesis*

...

This work was supported by DARPA / TORC Robotics ...

*Verifiable Robotics Research Group, Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853, USA {sm2296, vhp22, hadaskg}@cornell.edu.

†Robert BOSCH GmbH philipp.schillinger@de.bosch.com.

‡Capable Humanitarian Robotics & Intelligent Systems Lab, Department of Physics, Computer Science and Engineering, Christopher Newport University, Newport News, VA 23606, USA david.conner@cnu.edu.

Each author should review and fix their contact info.

III. Problem Statement

Problem 1 (Discrete Abstraction): Given ATLAS' control mode transition constraints and the available actions $\mathcal{A}$, define a discrete abstraction $\mathcal{D}$ of the robot-plus-software system, S, that captures the execution and outcomes of the software implementation of control mode transitions and actions. In addition, maintain a mapping, $\mathcal{D}_M : S \to \mathcal{AP}$, between the elements of the discrete abstraction and the corresponding software components.

In practice, the mapping $\mathcal{D}_M$ is one-to-many. But we can treat it as one-to-one to simplify the things for the paper.

In general, we would have included the robot's workspace in $\mathcal{D}$. However, that was not necessary in the context of the DRC.

Problem 2 (Formal Task Specification): Given a task, in terms of goals $\mathcal{G}$, the task's initial conditions $\mathcal{I}$, and the discrete abstraction, $\mathcal{D}$, of the system S that is to carry out the task, automatically compile a specification, $\mathcal{T}_S$, that encodes, in a formal language, the task being carried out by S.

Problem 3 (Behavior Synthesis): Given a formal task specification, $\mathcal{T}_S$, and the mapping $\mathcal{D}_M$, automatically generate a software implementation of a discrete, high-level, control strategy that is verifiably guaranteed to satisfy $\mathcal{T}_S$.

IV. Discrete Abstraction

A. Control Modes

We model ATLAS' control mode interface (c.f. Fig. 1) as a transition system $(\mathcal{M}, \to)$, where $\mathcal{M}$ is the set of states, each corresponding to one control mode, $m \in \mathcal{M}$, and $\to$ is a set of valid control mode transitions (subset of $\mathcal{M} \times \mathcal{M}$). In addition, we define $Adj(m) = \{m' \in \mathcal{M} \mid (m, m') \in \to\}$ and also allow self-transitions, i.e., $m \in Adj(m)$, $\forall m \in \mathcal{M}$.

B. Proposition Grounding

Write LTL formulas in terms of any possible outcome, $o \in Out(a)$, or only of completion and failure, $\{c, f\}$?

We abstract the discrete actions, $a \in \mathcal{A}$, that ATLAS can perform using one system proposition, π_a, per action and one environment proposition, π_a^o, per possible outcome of that action, $o \in Out(a)$. Similarly,[1] for each control mode, $m \in \mathcal{M}$, we have a system proposition π_a and a number of outcome propositions π_m^o. For both actions and control mode transitions, the outcomes that are of most interest in the context of this paper are completion (c) and failure (f) of the action. That is, $Out(a) = Out(m) = \{c, f\}$ Therefore, the set of atomic propositions AP is given by Eq. (1):

$$\mathcal{Y} = \bigcup_{a \in \mathcal{A}} \pi_a \bigcup_{m \in \mathcal{M}} \pi_m, \tag{1a}$$

$$\mathcal{X}' = \mathcal{X} \bigcup_{a \in \mathcal{A}} \bigcup_{o \in Out(a)} \pi_a^o \bigcup_{m \in \mathcal{M}} \bigcup_{o \in Out(m)} \pi_m^o, \tag{1b}$$

[1]The distinction between action and control mode propositions is purely for the sake of clarity of notation. There is nothing special about either.

where $\mathcal{X}$ contains environment propositions other than outcome propositions. For example, propositions that abstract sensors, as per [2].

V. Formal Task Specification

A. Multi-Paradigm Specification

...

B. LTL Specification for ATLAS

1) Generic Formulas: The system safety requirements (2) dictate that an activation proposition should turn `False` once an outcome has been returned.

$$\bigwedge_{o \in Out(a)} \square \left(\pi_a \wedge \bigcirc \pi_a^o \Rightarrow \bigcirc \neg \pi_a \right) \tag{2}$$

The environment safety assumptions (3) dictate that the outcomes of an action are mutually exclusive. For example, an action cannot both succeed and fail.

$$\bigwedge_{o \in Out(a)} \square \left(\bigcirc \pi_a^o \Rightarrow \bigwedge_{o' \neq o} \bigcirc \neg \pi_a^{o'} \right) \tag{3}$$

2) Action-specific Formulas: The environment safety assumptions (4) govern the value of outcomes in the next time step. Specifically, formula (4a) says that if an outcome has been returned, and the corresponding action is re-activated, then any outcome can become `True`. Formula (4b) dictates that, if an outcome is `False` and the corresponding action is not activated, then that outcome should remain `False`. This pair of formulas is a generalization of the "fast-slow" formulas (3) and (4) in [1].

Formula (4a) is outdated. It doesn't account for the activation–deactivation paradigm!

$$\square \left(\bigvee \pi_a^o \wedge \pi_a \Rightarrow \bigvee \bigcirc \pi_a^o \right) \tag{4a}$$

$$\bigwedge_{o \in Out(a)} \square \left(\neg \pi_a^o \wedge \neg \pi_a \Rightarrow \bigcirc \neg \pi_a^o \right) \tag{4b}$$

The environment safety assumptions (5) dictate that the value of an outcome should not change if the corresponding action has not been activated again. In other words, the outcome persists.

$$\bigwedge_{o \in Out(a)} \square \left(\pi_a^o \wedge \neg \pi_a \Rightarrow \bigcirc \pi_a^o \right) \tag{5}$$

The environment liveness assumption (6c) is a fairness condition. It states that, (always) eventually, either the activation of an action will return an outcome, (6a), or that the robot will "change its mind", (6b). Formula (6a) is a generalization of $\varphi_a^{completion}$ in [1], whereas formula (6b) is exactly the same as φ_a^{change} in [1], since it consists of activation propositions only.

$$\varphi_a^{return} = \Big(\pi_a \wedge \bigvee \bigcirc \pi_a^o\Big) \vee \Big(\neg\pi_a \wedge \bigwedge \bigcirc \neg\pi_a^o\Big) \quad (6a)$$

$$\varphi_a^{change} = \big(\pi_a \wedge \bigcirc \neg\pi_a\big) \vee \big(\neg\pi_a \wedge \bigcirc \pi_a\big) \quad (6b)$$

$$\square\diamond\big(\varphi_a^{return} \vee \varphi_a^{change}\big) \quad (6c)$$

The system safety requirement (7) demonstrates how a formula encoding the preconditions of an action, $Pre(a)$, looks like in the activation–outcomes paradigm.

Demonstrate how, given partial specification, we can bring in only those actions and modes that are necessary.

$$\square\big(\bigvee_{p\in Pre(a)} \neg\pi_p^c \Rightarrow \neg\pi_a\big) \quad (7)$$

where the superscript $c \in Out(p)$ stands for "completion".

3) Control Mode Formulas: For brevity of notation, let $\varphi_m = m \wedge \bigwedge_{m'\neq m} \neg\pi_{m'}$. Activating φ_m takes into account the mutual exclusion between the control modes $m \in \mathcal{M}$.

The system safety requirements (8) encode a topological transition relation, such the BDI control mode transition system.

$$\bigwedge_{m\in\mathcal{M}} \square\Big(\bigcirc \pi_m^c \Rightarrow \bigvee_{m'\in Adj(m)} \bigcirc \varphi_{m'} \vee \bigcirc \varphi_{\mathcal{M}}^{none}\Big) \quad (8)$$

where $\varphi_{\mathcal{M}}^{none} = \bigwedge_{m\in\mathcal{M}} \neg\pi_m$ being `True` stands for not activating any control mode transitions.

The environment safety assumptions (9) enforce mutual exclusion between the BDI control modes.

This formula requires the $\bigcirc$ operators to synthesize properly (slugs), but intuitively, they shouldn't be there.

$$\bigwedge_{m\in\mathcal{M}} \square\Big(\bigcirc \pi_m^c \Leftrightarrow \bigwedge_{m'\neq m} \bigcirc \neg\pi_{m'}^c\Big) \quad (9)$$

The environment safety assumptions (10) govern how the active control mode can change in a single time step in response to the activation of a control mode transition.

$$\bigwedge_{m\in\mathcal{M}} \bigwedge_{m'\in Adj(m)} \square\Big(\pi_m^c \wedge \varphi_{m'} \Rightarrow \big(\bigcirc \pi_m^c \bigvee_{o\in Out(m')} \bigcirc \pi_{m'}^o\big)\Big) \quad (10)$$

The environment safety assumptions (11) constrain the outcomes control mode transitions.

$$\bigwedge_{m\in\mathcal{M}} \bigwedge_{o\in Out(m)} \square\Big(\neg\pi_m^o \wedge \neg\pi_m \Rightarrow \bigcirc \neg\pi_m^o\Big) \quad (11)$$

The environment safety assumptions (12) dictate that the value of the outcomes of control mode transitions must not change if no transition is being activated, i.e., they must persist.

$$\bigwedge_{m\in\mathcal{M}} \bigwedge_{o\in Out(m)} \square\Big(\pi_m^o \wedge \varphi_{\mathcal{M}}^{none} \Rightarrow \bigcirc \pi_m^o\Big) \quad (12)$$

The environment liveness assumption (13c) is the equivalent of the fairness condition (6c) for control mode propositions.

$$\varphi_{\mathcal{M}}^{return} = \bigvee_{m\in\mathcal{M}} \Big(\varphi_m \wedge \bigvee_{o\in Out(m)} \bigcirc \pi_m^o\Big) \quad (13a)$$

$$\varphi_{\mathcal{M}}^{change} = \bigvee_{m\in\mathcal{M}} \Big(\varphi_m \wedge \bigcirc \neg\varphi_m\Big) \quad (13b)$$

$$\square\diamond\big(\varphi_{\mathcal{M}}^{return} \vee \varphi_{\mathcal{M}}^{change} \vee \varphi_{\mathcal{M}}^{none}\big) \quad (13c)$$

4) Initial Conditions: For each action, a, and control mode, m, in the initial conditions, $\mathcal{I}$, the completion proposition should be `True` in the environment initial conditions (14). All other outcome propositions corresponding to those actions and control modes, as well as all outcome propositions corresponding to any other actions and control modes, should be `False`.

$$\varphi_i^e = \bigwedge_{i\in\mathcal{I}} \Big(\pi_i^c \bigwedge_{o\in Out(i)\setminus\{c\}} \neg\pi_i^o\Big) \wedge \bigwedge_{j\notin\mathcal{I}} \bigwedge_{o\in Out(j)} \neg\pi_j^o \quad (14)$$

Activation propositions are `False` regardless of whether that action or control mode is in the initial conditions or not (15). The reason being that, intuitively, if we want something to be an initial condition, then we shouldn't have the resulting controller re-activate it at the beginning of execution.

$$\varphi_i^s = \bigwedge_{i\in\mathcal{I}} \neg\pi_i \wedge \bigwedge_{j\notin\mathcal{I}} \neg\pi_j \quad (15)$$

5) Success and Failure: The system initial condition (16), safety requirements (17) and (18), and liveness requirement (19) are used to reason about the satisfaction of the system's goals, $g \in \mathcal{G}$, in a finite run (as opposed to infinite execution, which is what LTL is defined over). In this finite run paradigm, the synthesized state machine (SM) itself has outcomes, $o \in Out(SM)$. The propositions corresponding to the SM's outcomes, π_{SM}^o, are system, not environment, propositions. The system propositions, μ_g, serve as memory of having accomplished each goal (c.f. [3]).

$$\bigwedge_{g\in\mathcal{G}} \neg\mu_g \quad (16)$$

$$\bigwedge_{g\in\mathcal{G}} \square\Big(\bigcirc \pi_g^c \Rightarrow \bigcirc \mu_g\Big) \quad (17a)$$

$$\bigwedge_{g\in\mathcal{G}} \square\Big(\mu_g \Rightarrow \bigcirc \mu_g\Big) \quad (17b)$$

$$\bigwedge_{g\in\mathcal{G}} \square\Big(\neg\mu_g \wedge \bigcirc \neg\pi_g^c \Rightarrow \bigcirc \neg\mu_g\Big) \quad (17c)$$

$$\Box\Big(\pi^c_{SM} \Leftrightarrow \bigwedge_{g\in\mathcal{G}} \mu_g\Big) \tag{18a}$$

$$\Box\Big(\pi^f_{SM} \Leftrightarrow \bigvee_{\pi\in\mathcal{Y}} \pi^f\Big) \tag{18b}$$

$$\bigwedge_{o\in Out(SM)} \Box\Big(\pi^o_{SM} \Rightarrow \bigcirc\, \pi^o_{SM}\Big) \tag{18c}$$

$$\Box\Diamond\,(\bigvee_{o\in Out(SM)} \pi^o_{SM}) \tag{19}$$

The time complexity of synthesis is cubic in the number of liveness requirements. We save by only having one. Although that's probably dominated by the complexity being exponential in the number of propositions.

Formulas (17) do not guarantee that the goals will be achieved in a specific order. However, that is often desirable. To account for it, we can define the goals as an ordered set $\mathcal{G} = \{g_1, g_2, \ldots, g_n\}$, where $g_i < g_j$ for $i < j$, and the relation $g_i < g_j$ means that goal g_i has to be achieved before g_j. With this definition, we can add the optional safety requirement (20), whenever strict goal order is desired.

$$\bigwedge_{i=1}^{n} \Box\left(\neg\mu_{g_{i-1}} \Rightarrow \bigcirc\,\neg\mu_{g_i}\right),\ \mu_{g_0} \triangleq \texttt{True} \tag{20}$$

Finally, these auxiliary (memory and SM outcome) propositions have to be added to the system propositions:

$$\mathcal{Y}' = \mathcal{Y} \bigcup_{g\in\mathcal{G}} \mu_g \bigcup_{o\in Out(SM)} \pi^o_{SM}$$

VI. ROS Implementation

VII. Experimental Validation

Provide data on how computationally costly/cheap behavior synthesis is. Time vs number of actions?

VIII. Conclusions and Future Work

Acknowledgments

The authors thank all other members of Team ViGIR, especially Alberto Romay, Stefan Kohlbrecher, and Prof. Oskar von Stryk from Technische Universität Darmstadt.

References

[1] V. Raman, N. Piterman, and H. Kress-Gazit, "Provably Correct Continuous Control for High-Level Robot Behaviors with Actions of Arbitrary Execution Durations," in *IEEE Int'l. Conf. on Robotics and Automation*, 2013.

[2] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal logic based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[3] V. Raman, B. Xu, and H. Kress-Gazit, "Avoiding forgetfulness: Structured English specifications for high-level robot control with implicit memory," in *IEEE/RSJ Int'l. Conf. on Intelligent Robots and Systems*, 2012.