Managing an Agent's Changing Intentions Using LTL$_f$ Synthesis

Extended Abstract

Giuseppe De Giacomo
University of Oxford
University of Rome "La Sapienza"
Oxford - Rome, UK - Italy
giuseppe.degiacomo@cs.ox.ac.uk

Yves Lespérance
York University
Toronto, Canada
lesperan@eecs.yorku.ca

Gianmarco Parretti
University of Rome "La Sapienza"
Rome, Italy
parretti@diag.uniroma1.it

Fabio Patrizi
University of Rome "La Sapienza"
Rome, Italy
patrizi@diag.uniroma1.it

Renzo Schram
Utrecht University
Utrecht, Netherlands
renzoschram@gmail.com

ABSTRACT

Autonomous agents' intentions (goals they are committed to) typically change as they operate. We develop a new model of intention change for such agents. We assume that the agent operates in a fully observable nondeterministic (FOND) domain and uses Linear Temporal Logic over finite traces (LTL$_f$) to represent intentions. We exploit LTL$_f$ synthesis notions and techniques to generate strategies for the agent to satisfy its intentions and to revise them when the agent adopts new intentions or drops existing ones; this ensures that the agent's intentions always remain realizable. We propose automata-based methods to efficiently manage LTL$_f$ intentions by exploiting auxiliary data structures built during synthesis. We implement a prototype and evaluate its effectiveness experimentally.

KEYWORDS

Intentions' Management; Linear Temporal Logic on Finite Traces (LTL$_f$); Reactive Synthesis; Maximally Permissive Strategies

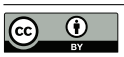
ACM Reference Format:
Giuseppe De Giacomo, Yves Lespérance, Gianmarco Parretti, Fabio Patrizi, and Renzo Schram. 2025. Managing an Agent's Changing Intentions Using LTL$_f$ Synthesis: Extended Abstract. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 3 pages.

1 INTRODUCTION

Autonomous agents' goals generally change as they operate. They adopt new goals and drop existing ones as a result of interactions with others and changes in their own motivations. In the standard Belief-Desire-Intention (BDI) model of agency [2, 4, 10], one distinguishes between desires and *intentions*, i.e., the latter being the desires that the agent is committed to. Intentions trigger planning and lead to action. Intentions must also be consistent with each other and with the agent's beliefs. This essentially means that there must exist a plan or strategy for the agent to satisfy all of them given its beliefs. Also, if the agent wants to adopt a new intention, it must drop current intentions that are inconsistent with it. In principle, one could ensure that intentions remain jointly achievable by checking the existence of a plan/strategy to achieve all of them every time the agent's beliefs change or a new intention is adopted. But this seems infeasible in practice so various alternative approaches have been developed.

The BDI model has inspired the mainstream approach to autonomous agent system development, BDI agent programming [1, 9]. It uses hand-crafted hierarchical plans that are reactively executed to fulfill the agent's intentions, which are achievement goals. A plan to achieve a subgoal/intention is associated with a guard/context condition that specifies when it may be selected. These guards are evaluated against the agent's beliefs, which are assumed to be complete. The deliberation mechanism selects plans whose guard is satisfied to try to realize the active intentions, adopts them as new intentions, and proceeds to execute them, i.e., execute the actions and adopt the subgoals that they contain. The rationale for this approach has been that planning is slow and the environment may change often thus requiring much replanning. However, the deliberation mechanism *does not usually check that all the adopted intentions and selected plans remain jointly acheivable*. In fact, it may not even have a specification of the actions' effects and domain dynamics. This means that the programmer must anticipate all the contingencies that the agent may face and all the possible interactions between plans, and there is no guarantee that the agent will achieve its intentions.

More recently, how to do such deliberation has been formalized as the intention progression problem and competitions for it have been organized [3, 8]. The task is taken to be that of successfully progressing a set of goal-plan trees (GPT) for the agent's intentions, i.e., selecting sub-plans/intentions and interleaving their executions to avoid undesirable interactions. Techniques based on Monte-Carlo tree search have shown promise [13, 14]. There has also been work on handling early achievement of goals and plan failure in the deliberation process [7, 11, 12]. However, such approaches that try to find an interleaving of plans for individual reachability goals to achieve all of them are incomplete and don't generalize to arbitrary temporally extended goals. So if we want to guarantee intentions' joint achievability, it seems we are must fall back on replanning.



Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

In this paper, we develop an alternative approach to managing intention change that builds on *reactive synthesis for Linear Temporal Logic on finite traces (LTL$_f$)* [6, 16]. We propose an intention management framework where we assume that the agent has a dynamically changing set of intentions, which are specified by arbitrary LTL$_f$ formulas. We also assume that the agent operates in a fully observable nondeterministic (FOND) domain, where the agent does not fully control the outcomes of its actions. We use LTL$_f$ synthesis to check that all the agent's intentions remain *realizable*, i.e., jointly achievable no matter how the environment behaves, and to generate strategies to achieve them. In fact, we generate maximally permissive strategies [15], which leaves the agent as much autonomy as possible (perhaps to satisfy some additional preferences/softgoals); in principle, this is not much harder that generating a single strategy. This means that the programmer only has to specify the FOND domain, the intentions that the agent should adopt and drop, and which actions from the synthesized strategy the agent should execute. It also ensures that the agent behaves with a high degree of rationality. Our approach to intention management can be viewed as an advanced form of planning in which the agent generates and executes a plan/strategy for satisfying its current goals/intentions, but is also ready to revise the set of goals and replan during execution.

We present a formal specification of rational intention change and action, i.e., *Rational Intention Management (RIM)*, that exploits LTL$_f$ synthesis notions. An agent's intentions state is defined as a triple $I = \langle \mathcal{D}, s, L \rangle$, where $\mathcal{D}$ is the domain the agent operates in, s is the current domain state, and L is the list of currently adopted LTL$_f$ intentions. The agent's intentions in $L = [\varphi_1, \cdots, \varphi_n]$ are sorted in decreasing order of priority so that φ_1 has the highest priority and φ_n the lowest. At each time step, we must ensure that the agent's intentions list is *realizable*, i.e., there exists a *winning* strategy that jointly achieves all the currently adopted intentions.

We define operations to query and update the agent's intentions state. Query operations retrieve information from the current agent's intentions state. Among such operations some allow the agent to retrieve the actions available in its current intentions state. We distinguish between *winning* and *progressing* actions: the former are those prescribed by winning strategies in the current agent's intentions state; the latter are actions prescribed by winning strategies that achieve the joint intentions in the least number of steps regardless of the environment's response. The sets of winning and progressing actions are respectively retrieved by the RIM operations $I.getWinningActions()$ and $I.getProgressingActions()$.

Another key query operation is $I.isRealizable(\phi, k)$, which decides whether adopting a new intention ϕ at priority index k results in a realizable list of intentions and, possibly, returns the list of indexes of intentions to be dropped for ϕ to be adopted. In fact, intentions' priority plays a key role during adoption of a new intention. The agent can adopt the intention ϕ at priority index k only if ϕ is realizable jointly with all currently adopted higher priority intentions, i.e., $\bigwedge_{i<k} \varphi_i \wedge \phi$ is realizable; if during adoption some lower priority intention φ_j (with $j \geq k$) becomes unrealizable, the agent has to drop φ_j. The formal specification of $I.isRealizable(\phi, k)$ captures this requirement. It returns:

- `True` if $\bigwedge_{\varphi_i \in L} \varphi_i \wedge \phi$ is realizable;
- `False` if $\bigwedge_{i<k} \varphi_i \wedge \phi$ is unrealizable;
- $I.isRealizable(\phi, k, L.size())$, otherwise, where $I.isRealizable(\phi, k, j)$ is defined as:
  - $I.isRealizable(\phi, k, j) = [1, \cdots, k-1]$ if $j < k$;
  - Otherwise: $I.isRealizable(\phi, k, j) =$

$$\begin{cases} I.isRealizable(\phi, k, j-1) + [j] \\ \quad \text{if } (\bigwedge_{\ell \in I.isRealizable(\varphi, k, j-1)} \varphi_\ell) \wedge \phi \wedge \varphi_j \text{ is realizable} \\ I.isRealizable(\phi, k, j-1) \text{ otherwise} \end{cases}$$

Update operations modify the agent's intentions state and include $I.drop(L')$, which drops intentions whose indexes appear in L', and $I.adopt(\phi, k)$, which adopts an intention ϕ at priority index k and whose formal specification is as follows: let `R` = $I.isRealizable(\phi, k)$: (*i*) if `R = True`, it executes $L.insert(\phi, k)$; else; (*ii*) if `R = False`, it does nothing; else (*iii*) if `R = RList`, it executes $I.drop([\ell])$ for every $\ell \notin L'$ and $L.insert(\phi, k)$. We also have $I.prog(a, s')$, which progresses the agent's intentions state to that resulting from the execution of action a and the successor domain state s' (nondeterministically chosen by the environment).

We developed an automata-based implementation of a Rational Intention Management System (RIMS) that uses LTL$_f$ synthesis [5, 6] of maximally permissive strategies [15] to implement the RIM operations. This implementation is based on the following result: LTL$_f$ synthesis can be reduced to solving games over deterministic finite automata (DFA) obtained from the FOND domain and LTL$_f$ intentions [5, 6]. Since these games are exponential (resp. doubly-exponential) in the domain's (resp. intentions') size, we devised a data structure that stores the DFAs of the currently adopted intentions and the maximally permissive strategy that allows the agent maximum autonomy while ensuring that it achieves all intentions. Such a data structure is used to minimize the executions of computationally expensive operations, e.g., the transformation of LTL$_f$ intentions into DFAs, each of which costs 2EXPTIME [6]. To improve performance, we also devised a compositional game-resolution technique that uses the winning strategies for each individual intention. The automata-based RIMS is implemented in a tool called RIMS4LTLF that is available at https://github.com/GianmarcoDIAG/rims4ltlf.git.

We evaluated our approach empirically and showed that it can be used to effectively implement agents that operate in non-trivial domains and adopt numerous intentions over the course of a run (dropping is similar but easier). The experiments show that RIMS4LTLF scales better than a basic approach that synthesizes the maximally permissive strategy for the conjunction of all intentions. In particular, the time required by RIMS4LTLF to adopt a new intention at lowest priority is often one order of magnitude less than the time required by the basic approach. This result suggests that RIMS4LTLF benefits from using our compositional game-resolution technique and is better than the basic approach for an agent that adopts (and drops) intentions in an incremental/online manner.

ACKNOWLEDGMENTS

This work is supported in part by the ERC Advanced Grant WhiteMech (No. 834228), the PRIN project RIPER (No. 20203FFYLK), the PNRR MUR project FAIR (No.PE0000013), the Sapienza MARLeN Project, the UKRI AI Hub on Mathematical and Computational Foundations of AI, the National Science and Engineering Research Council of Canada, and York University. Gianmarco Parretti is supported by the Italian National Ph.D. on AI at "La Sapienza".

REFERENCES

[1] Rafael H. Bordini, Jomi F. Hübner, and Michael J. Wooldridge. 2007. *Programming Multi-Agent Systems in AgentSpeak Using Jason.* Wiley.

[2] Michael E. Bratman. 1987. *Intention, plans and practical reason.* Harvard University Press.

[3] Simon Castle-Green, Alexi Dewfall, and Brian Logan. 2020. The Intention Progression Competition. In *EMAS@AAMAS.*

[4] Philip R. Cohen and Hector J. Levesque. 1990. Intention is Choice with Commitment. *Artif. Intell.* 42 (1990).

[5] Giuseppe De Giacomo and Sasha Rubin. 2018. Automata-Theoretic Foundations of FOND Planning for LTLf and LDLf Goals. In *IJCAI.* 4729–4735.

[6] Giuseppe De Giacomo and Moshe Y. Vardi. 2015. Synthesis for LTL and LDL on Finite Traces. In *IJCAI.*

[7] Koen V. Hindriks, Frank S. de Boer, Wiebe van der Hoek, and John-Jules Ch. Meyer. 2000. Agent Programming with Declarative Goals. In *ATAL.*

[8] Brian Logan, John Thangarajah, and Neil Yorke-Smith. 2017. Progressing Intention Progression: A Call for a Goal-Plan Tree Contest. In *AAMAS.*

[9] Anand S. Rao. 1996. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In *MAAMAW.*

[10] Anand S. Rao and Michael P. Georgeff. 1991. Modeling Rational Agents within a BDI-Architecture. In *KR.*

[11] Birna van Riemsdijk, Mehdi Dastani, Frank Dignum, and John-Jules Ch. Meyer. 2004. Dynamics of Declarative Goals in Agent Programming. In *DALT.*

[12] Michael Winikoff, Lin Padgham, James Harland, and John Thangarajah. 2002. Declarative & Procedural Goals in Intelligent Agent Systems. In *KR.*

[13] Yuan Yao and Brian Logan. 2016. Action-Level Intention Selection for BDI Agents. In *AAMAS.*

[14] Yuan Yao, Brian Logan, and John Thangarajah. 2014. SP-MCTS-based Intention Scheduling for BDI Agents. In *ECAI.*

[15] Shufang Zhu and Giuseppe De Giacomo. 2022. Synthesis of Maximally Permissive Strategies for LTL_f Specifications. In *IJCAI.*

[16] Shufang Zhu, Lucas M. Tabajara, Jianwen Li, Geguang Pu, and Moshe Y. Vardi. 2017. Symbolic LTL_f Synthesis. In *IJCAI.*