Artificial Intelligence 231 (2016) 64–106

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Agent planning programs

Giuseppe De Giacomo[a], Alfonso Emilio Gerevini[b], Fabio Patrizi[c], Alessandro Saetti[b], Sebastian Sardina[d,*]

[a] *Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Italy*
[b] *Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Italy*
[c] *KRDB Research Centre for Knowledge and Data, Faculty of Computer Science, Free University of Bozen–Bolzano, Italy*
[d] *School of Computer Science and IT, RMIT University, Melbourne, Australia*

A R T I C L E   I N F O

Article history:
Received 19 July 2014
Received in revised form 27 June 2015
Accepted 10 October 2015
Available online 3 November 2015

Keywords:
Agent-oriented programming
Automated planning
Reasoning about action and change
Synthesis of reactive systems

A B S T R A C T

This work proposes a novel high-level paradigm, *agent planning programs*, for modeling agents behavior, which suitably mixes automated planning with agent-oriented programming. Agent planning programs are finite-state programs, possibly containing loops, whose atomic instructions consist of a guard, a maintenance goal, and an achievement goal, which act as precondition-invariance-postcondition assertions in program specification. Such programs are to be executed in possibly nondeterministic planning domains and their execution requires generating plans that meet the goals specified in the atomic instructions, while respecting the program control flow. In this paper, we define the problem of automatically synthesizing the required plans to execute an agent planning program, propose a solution technique based on model checking of two-player game structures, and use it to characterize the worst-case computational complexity of the problem as EXPTIME-complete. Then, we consider the case of deterministic domains and propose a different technique to solve agent planning programs, which is based on iteratively solving classical planning problems and on exploiting goal preferences and plan adaptation methods. Finally, we study the effectiveness of this approach for deterministic domains through an experimental analysis on well-known planning domains.

1. Introduction

This work proposes a novel paradigm for programming intelligent agents and controllers in a task-oriented way that mixes automated planning with agent-oriented programming w.r.t. behavior specification.[1] Generally speaking, we envision the designer providing a high-level model of the "space of deliberation" of the agent—called an *agent planning program*—that is meant to be "realized" into an executable program via automatic synthesis. Agent planning programs are finite-state programs, possibly containing loops, whose atomic instructions are classical *precondition-invariance-postcondition* declarative assertions. Such programs are to be executed in possibly nondeterministic domains. A "realization" of such programs in the domain of concern amounts, basically, to a collection of inter-related plans that meet the assertions in the atomic

* Corresponding author.
E-mail addresses: degiacomo@dis.uniroma1.it (G. De Giacomo), alfonso.gerevini@ing.unibs.it (A.E. Gerevini), patrizi@dis.uniroma1.it (F. Patrizi), alessandro.saetti@ing.unibs.it (A. Saetti), sebastian.sardina@rmit.edu.au (S. Sardina).
[1] This work integrates and extends [34,44].

http://dx.doi.org/10.1016/j.artint.2015.10.001
0004-3702/© 2015 Elsevier B.V. All rights reserved.

instructions while respecting the program control flow in its totality (that is, a plan for an assertion should not preclude the realization of potential future instructions).

Technically, the dynamics of the world is described with a *planning domain* and a given initial state, as usually done in domain-independent planning [46] and reasoning about action [88]. On top of such (rooted) domain, an agent planning program is modeled as a finite transition system, typically including loops, in which states represent *choice points* and transitions specify possible objectives that the agent may decide to pursue. Such transitions constitute the high-level actions available to the agent, and are characterized by: *(i)* a *guard*, which poses executability preconditions; *(ii)* a *maintenance goal*, which specifies invariants that are guaranteed to hold for the course of actions to execute; and *(iii)* an *achievement goal*, which specifies the postcondition of the transition. In other words, those triples are a direct counterpart of the classical triple precondition-invariant-postcondition, used in program specification [38,42,55] and nowadays in "design-by-contract" or "code-by contract" development [74].

Intuitively, agent planning programs are meant to work as follows: at any point in time, based on the current state of the domain and that of the agent planning program, the agent decides, autonomously, which enabled program transition to pursue. A (synthesized) plan satisfying the assertions in the chosen transition is then executed, thus moving the domain and the program to their next states, from which a new transition will be "requested" by the agent, a new plan executed again, and so on. The agent planning program is said to be *realized* if an adequate plan can always be associated to the execution of transitions, according to the planning program control flow. Note that, although the planning program is a finite transition system, it may generate, due to loops, an infinite computation tree; in principle, one needs to synthesize plans for each of the infinitely many transitions of such a tree. A key point is that, in synthesizing a plan for a particular transition, one needs to take into account that the resulting state of the domain must not only satisfy the corresponding achievement goal assertion, but also must allow for the existence of plans for each possible next transition, and this must hold again after such plans, and so on.

By combining declarative and procedural approaches to behavior specification, together with automatic synthesis techniques, the agent planning program approach has the potential to provide convenient and powerful specification of behavior in complex scenarios. For example, they can be used to encode knowledge-intensive business processes (processes reflecting "preferred work practices" whose execution is controlled by contingent agent decision making, coupled with contextual data and knowledge production) [27,99],[2] or even non-linear *storylines* behind characters' actions in a video game [18,85]. Planning programs can also be a convenient model of an embedded system for a smart house controller [52] or a Holonic manufacturing system [50], in which the actual concrete manner of doing things may vary from setting to setting. Last, but not least, they can be used to specify the requirements for a web-service [72]. The assumption is that the agent (e.g., a human interacting with a business process, embedded system, or a game narrative generator) issues, step-by-step, *goal requests within the given space of deliberation*, which are to be fulfilled by appropriate plans computed by the solver.

In this paper, we study the above realizability (and associated synthesis) problem and provide the following contributions:

- A formal definition of the problem of realizing an agent planning program and its solution.
- A correct and terminating technique for synthesizing realizations, which resorts to automated synthesis for certain kinds of Linear-time Temporal Logic (LTL) specifications based on model checking game structures [80,82]. Interestingly, such a technique can be readily implemented using available tools for synthesis based on model checking of game structures, such as the well-known TLV [84], JTLV [83], or the more recent NuGaT [17], which we use for our experiments.
- A worst-case complexity characterization of the problem as EXPTIME-complete, where we use the above technique for establishing membership in EXPTIME. The EXPTIME-hardness comes from the EXPTIME-completeness of conditional planning with full observability in nondeterministic domains [90], which is a special case of our problem: a planning program formed by a single transition labelled with an achievement goal.

The output obtained from our general realization technique is akin to a sort of sophisticated form of universal plan [92], which is obviously a costly solution [48]. To deal with this, in the second part of the paper, we look for alternative computational approaches based on exploiting state-of-the-art classical planning systems. In particular, we focus on the case of deterministic underlying domains, widely studied in automated planning, for which classical planning systems have shown excellent performance. The contributions for this case are the following:

- We show that the worst-case complexity of the problem remains EXPTIME-complete even for deterministic domains. In particular, for membership we can still use the general algorithm, while for the hardness we show a reduction from the service composition problem [76].
- We devise a technique for realizing planning programs that is based on classical planning tools, which involves iteratively *constructing* and *synchronizing* a set of plans. Importantly, the technique makes use of *goal preferences* and *plan adaptation* to considerably speed up plan synthesis and synchronization when realizing looping transitions.

[2] In particular, in [27], an early version of agent planning programs was used for expressing behavioral routines for people with special needs in dedicated smart homes.