Annals of Pure and Applied Logic 165 (2014) 134–163

Contents lists available at SciVerse ScienceDirect

Annals of Pure and Applied Logic

www.elsevier.com/locate/apal

Hybrid Answer Set Programming

Alex Brik [1], Jeffrey Remmel [*]

Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112, USA

ARTICLE INFO

Article history:
Available online 20 August 2013

MSC:
68T27
68T30
03B70
68N17

Keywords:
Answer set programming
Markov Decision Processes
Stable models
Dynamic domains
Modeling and simulation

ABSTRACT

This paper discusses an extension of Answer Set Programming (ASP) called Hybrid Answer Set Programming (H-ASP) which allows the user to reason about dynamical systems that exhibit both discrete and continuous aspects. The unique feature of Hybrid ASP is that it allows the use of ASP type rules as controls for when to apply algorithms to advance the system to the next position. That is, if the prerequisites of a rule are satisfied and the constraints of the rule are not violated, then the algorithm associated with the rule is invoked.

1. Introduction

The main purpose of this paper is to discuss an extension of Answer Set Programming (ASP) which we call *Hybrid Answer Set Programming* (H-ASP). H-ASP was introduced by authors in [5]. The goal of H-ASP is to allow the user to reason about dynamical systems that exhibit both discrete and continuous aspects. The unique feature of H-ASP is that H-ASP rules can be thought of as general input–output devices. In particular, H-ASP programs allow the user to include ASP type rules that act as controls for when to apply a given algorithm to advance the system to the next position.

Modern computational models and simulations such as the model of dog's heart described in [14], the model of tsunami propagation described in [11], the model of internal tides within Monterey Bay and surrounding area described in [13], as well as others, rely on PDE solvers and ODE solvers to determine the values of parameters. Such simulations proceed by invoking appropriate algorithms to advance the system to the next state, which is often distanced by a short time interval into the future from the current state. In this way a simulation of continuously changing parameters is achieved, although the simulation itself is a discrete system. The parameter passing mechanism, i.e. the logic for making decisions regarding what

* Corresponding author.
E-mail addresses: abrik@google.com (A. Brik), jremmel@ucsd.edu (J. Remmel).
[1] Currently at Google Inc.; this work was completed before the change in affiliation.

0168-0072/$ – see front matter
http://dx.doi.org/10.1016/j.apal.2013.07.008

algorithms to invoke and when, is part of the ad hoc control algorithm. Thus the laws of the system are implicit in the ad hoc control software.

On the other hand action languages [10] which are also used to model dynamical systems allow the users to describe the laws of a system explicitly. Initially action languages did not allow simulation of the continuously changing parameters. In particular, action languages do not allow simulations that require the use of PDE solvers or ODE solvers or both. Recently Chintabathina, Gelfond and Watson introduced an action language H [7]. Chintabathina et al. proposed an elegant approach to modeling continuously changing parameters—a program in H describes a state transition diagram of a system where each state models a time interval in which parameter dynamics is a known function of time. However, the implementation of H discussed in [7] cannot use PDE solvers nor ODE solvers. This means that parameters governed by physical processes such as the distribution of heat or air flow, which cannot be easily expressed as functions of time and realistic simulations of which often require sophisticated numerical methods, cannot be modeled using these implementations of H.

As we shall see, H-ASP allows the user to combine the use of numerical and probabilistic algorithms to realistically simulate physical processes and the expressive power of ASP which provides the ability to elegantly model laws of a system. Our approach is related but significantly different from various recent attempts to incorporate probabilistic reasoning into ASP type formalisms such as P-log [2] as well as recent work on integrating constraints logic programming into ASP type formalisms such as the work of Mellarkod, Gelfond, and Zhang [15] and the work of Gebser, Ostrowski, and Schaub [8]. In our approach, there are algorithms associated with rules which allow the user to employ numerical algorithms and/or probabilistic algorithms and the logic is used to control the circumstances in which we employ such algorithms to meet the goals of the user. We then show that the basic Gelfond–Lifschitz fixed point construction of stable models can be extended to such rules to define the analogue of stable models for H-ASP programs. A related approach is the design of modular action languages like $\mathcal{ALM}$ [12] which is intended to facilitate the creation of knowledge representation libraries. In $\mathcal{ALM}$, one separates the problem of describing a domain into two parts: a general theory consisting of several modules that can also be used in modeling other domains and an interpretation of this theory for a specific domain.

One of our long term goals is to develop extensions of ASP solvers that can process H-ASP programs. In action languages such as H, the goal is to compile an H program into a variant ASP program that can be processed with current ASP solvers. Developing H-ASP solvers would allow us to develop H-ASP type extensions of action languages like H that could be compiled to H-ASP programs which, in turn, would be processed by an H-ASP solver. We have implemented limited versions of an H-ASP solver in a language which we call H-ASP# which is described in the first author's [4] thesis.

Our first goal is to give the basic definitions of H-ASP programs and define an analogue of stable models for such programs. To help motivate our definitions, we shall consider the following toy example. Imagine that Secret Agent 00111 (the agent, for short) needs to move through a domain consisting of 3 areas: Area I, Area II, and Area III. The domain's vertical cross section is pictured in Fig. 1. Area I is a mountain, Area II is a lake, and Area III is a desert. Secret Agent 00111 needs to descend down the mountain in his car until he reaches the lake at which point the car can be reconfigured so that it can be used as a boat that can navigate across the lake. We shall assume that the lake has a water current moving with a constant speed of 5 m/s which makes an angle $\frac{4\pi}{3}$ clockwise from the positive direction of the x-axis. The agent is considered to be pursued by evil agents during a time interval of Δ seconds if a shot is heard at the beginning of the time interval, or if two shots were heard before the start of the interval and the shots were separated by Δ seconds. If our agent is pursued by evil agents on his trip down the mountain, then he will accelerate the car at 4 m/s^2 in addition to the acceleration due to gravity. If he is not pursued by evil agents, then he will simply coast down the hill. Furthermore, if the agent is pursued by the evil agents, then he will attempt to travel through the lake as fast as possible, always steering at a 90 degree angle to the opposite shore. If the agent is not pursued by the evil agents, then he would like to exit the lake at a point with a y-coordinate