Hybrid ASP

Alex Brik[1] and Jeffrey B. Remmel[2]

1 **Department of Mathematics, UC San Diego, U.S.A**
2 **Department of Mathematics, UC San Diego, U.S.A**

Abstract

This paper introduces an extension of Answer Set Programming (ASP) called Hybrid ASP which will allow 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.

1998 ACM Subject Classification I.2.3 Deduction and Theorem Proving

Keywords and phrases answer set programming, hybrid systems, modeling and simulation

Digital Object Identifier 10.4230/LIPIcs.ICLP.2011.40

1 Introduction

The purpose of this paper is to introduce an extension of Answer Set Programming (ASP) which we call *Hybrid ASP*. The goal of Hybrid ASP is to allow the user to reason about dynamical systems that exhibit both discrete and continuous aspects. The unique feature of Hybrid ASP is that Hybrid ASP rules can be thought of as general input-output devices. In particular, Hybrid 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 [6], the model of tsunami propagation described in [3], and the model of internal tides within Monterey Bay and the surrounding area described in [5] rely on existing PDE solvers and ODE solvers to determine the values of parameters. Such simulations proceed by invoking appropriate algorithms to advance a 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 mechanisms and the logic for making decisions regarding what algorithms to invoke and when are part of the ad-hoc control algorithm. Thus the laws of a system are implicit in the ad-hoc control software.

On the other hand, action languages [2] 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. This severely limited applicability of such languages. Recently, Chintabathina introduced an action language H [1] where he proposed an elegant approach to modeling continuously changing parameters. That is, a program in H describes a state transition diagram of a system where each state models a time interval where the parameter dynamics is a known function of time. However, the implementation of H discussed in [1] cannot use PDE solvers nor ODE solvers. This means that parameters governed by physical processes such as the distribution of heat or air flow, that cannot be described explicitly as functions of time and realistic simulations of

Technical Communications of the 27th International Conference on Logic Programming (ICLP'11).
Editors: John P. Gallagher, Michael Gelfond; pp. 40–50

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

which often require sophisticated numerical methods, cannot be modeled using the current implementations of H.

Hybrid ASP is an extension of ASP that allows users to combine the strength of the ad-hoc approaches, i.e. the use of numerical methods to faithfully simulate physical processes, and the expressive power of ASP that provides the ability to elegantly model laws of a system. Hybrid ASP provides mechanisms to express the laws of the modeled system via hybrid ASP rules which can control execution of algorithms relevant for simulation.

In action languages like H, the goal is to compile an H program into a variant ASP program that can be processed with current variant ASP solvers. Our long term goal is to develop extensions of ASP solvers that can process Hybrid ASP programs. This would allow us to develop Hybrid ASP type extensions of action languages like H that could be compiled to Hybrid ASP programs which, in turn, would be processed by Hybrid ASP solvers.

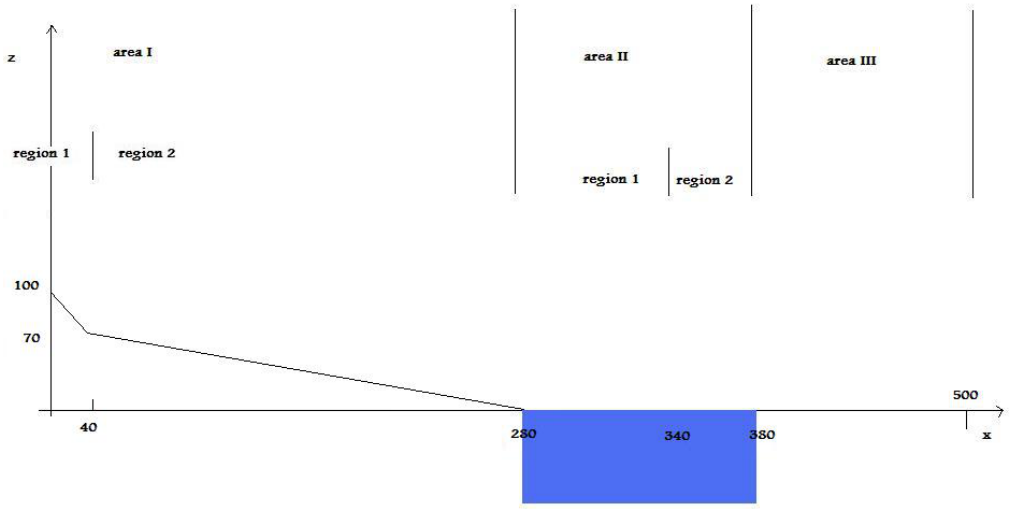

Figure 1 A cross section of the regions to be traversed by Secret Agent 00111.

In this paper, we shall present the basic definitions of Hybrid 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 shown on the diagram **??**. 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 $5m/s$ which makes an angle $\frac{4\pi}{3}$ clockwise from the positive direction of the x-axis. If Secret Agent 00111 is pursued by evil agents on his trip down the mountain, he will accelerate the car at $4m/s^2$ 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 degrees 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 being close to the y-coordinate of the point of his entrance into the lake. To accomplish this, Secret Agent 00111 will be able to steer the boat in directions which

make various angles to the x-axis. Finally upon entering the desert Secret Agent 00111 can again begin to accelerate at $4m/s^2$.

The outline of this paper is as follows. In section 2 we shall introduce Hybrid ASP programs. In section 3 we shall show how H-ASP programs can be used to model a dynamical system for our secret agent problem. In section 4 we shall provide conclusions and directions for further research.

2 Hybrid ASP

The main feature of H-ASP is to view rules as general input-output devices. Informally, this means that given a set of parameter values and a set of facts associated with it, a rule may or may not produce one or more new parameter values and an associated fact. This ensures that H-ASP is suitable for defining control mechanisms for discrete time simulations that require the use of PDE solvers or ODE solvers or both.

A H-ASP program P will have an underlying parameter space S. For instance, in our secret agent example, imagine that we allow Secret Agent 00111 to make decisions every Δ seconds. Then one can think of describing the Secret Agent 00111's position and situation at time $k\Delta$ by a sequence of parameters $\mathbf{x}(k\Delta) = (x_0(k\Delta), x_1(k\Delta), x_2(k\Delta), \ldots, x_m(k\Delta))$ that specify both continuous parameters such as time, position, velocity, and acceleration as well as discrete parameters such as is the car configured as a car or as a boat. In more complicated simulations, the programmer may not know the value Δ ahead of time as the exact value of Δ may be determined by the needs of the parameter passing algorithms. Nevertheless, we shall always assume that any parameter passing algorithm advances time in some discrete time steps which may vary depending on the values of the input parameters. Thus in a H-ASP program, one can always think of the parameter x_0 as specifying time and the range of x_0 is either $\{k\Delta : k = 0, \ldots, n\}$ for some fixed n and $\Delta > 0$ or $\{k\Delta : k \in \mathbb{N}\}$ where $\mathbb{N}$ is the set of natural numbers $\{0, 1, 2, \ldots, \}$. In particular, for a finite H-ASP program, there will be no loss of generality in assuming that the range of x_0 is $\{k\Delta : k = 0, \ldots, n\}$ for some fixed n and $\Delta > 0$. In such a situation, we shall always write an element of S in the form $\mathbf{x} = (k\Delta, x_1(k\Delta), \ldots, x_m(k\Delta))$ for some k. However, in our general definitions, we shall just assume that elements of the parameter space S are of the form $\mathbf{p} = (t, x_1, \ldots, x_m)$ where t is time and we shall let $t(\mathbf{p})$ denote t and $x_i(\mathbf{p})$ denote x_i for $i = 1, \ldots, m$. We refer to the elements of S as *generalized positions*. A H-ASP program will also have an underlying set of atoms At. Then the underlying universe of the program will be $At \times S$.

If $M \subseteq At \times S$, then we let $\widehat{M} = \{\mathbf{x} \in S : (\exists a \in At)((a, \mathbf{p}) \in M)\}$. We will say that M satisfies $(a, \mathbf{p}) \in At \times S$, written $M \models (a, \mathbf{p})$, if $(a, \mathbf{p}) \in M$. For any element $(t, x_1, \ldots, x_m) \in S$, we let $W_M(t, x_1, \ldots, x_m) = \{a \in At : (a, (t, x_1, \ldots, x_m)) \in M\}$ and we shall refer to $W_M(t, x_1, \ldots, x_m)$ as the *world of M at the generalized position* $(t, x_1, \ldots, x_m)$. We let $T(S) = \{t : \exists \mathbf{p} \in S(t = t(\mathbf{p})\}$. We say that M is a **single trajectory model** if for each $t \in T(S)$, there is exactly one generalized position of the form $(t, x_1, \ldots, x_m)$ in $\widehat{M}$. If M is a single trajectory model, then we let $(t, x_1(t), \ldots, x_m(t))$ be the unique element of the form $(t, x_1, \ldots, x_m)$ in $\widehat{M}$ and we can write M as a disjoint union

$$M = \bigsqcup_{t \in T(S)} W_M(t, x_1(t), \ldots, x_m(t)) \times \{(t, x_1(t), \ldots, x_m(t))\}.$$

We will say that M is **multiple trajectory model** if for each $t \in T(S)$, there is at least one generalized position of the form $(t, x_1, \ldots, x_m)$ in $\widehat{M}$ and for some $t \in T(S)$, there are

at least two generalized positions of the form $(t, x_1, \ldots, x_m)$ in $\widehat{M}$. Single trajectory stable models are desirable when the objective of a computation is to obtain a trajectory satisfying certain constraints. For example, in the planning problem, the objective is to find a sequence of actions that achieves a predefined goal given an axiomatized initial situation [8]. Thus solving planning problem provides a motivation for considering single trajectory models. Multiple trajectory models are natural to consider when the objective of a computation is a set of conclusions that depend on all the possible trajectories. For instance such a model would be useful in reasoning about a best strategy for an agent acting within a dynamical system where the consequences of actions are non-deterministic. Thus multiple trajectory models are natural to consider in the context of dynamic programming problems (see [10] for an introduction to dynamic programming).

If we drop the requirement that there is a generalized position $(t, x_1, \ldots, x_m) \in \widehat{M}$ for each $t \in T(S)$ in the definition of single trajectory or multiple trajectory models, we get what we call *partial single trajectory* and *partial multiple trajectory* models. For example, if $T(S) = \{k\Delta : k = 0, \ldots, n\}$, then we may want to allow our parameter passing algorithms the flexibility of advancing time by multiple steps of Δ so that it may be the case that our set of rules will derive no information about what happens a certain time $k\Delta$ in which case $W_M(k\Delta, x_1(k\Delta), \ldots, x_m(k\Delta))$ will be empty for all generalized positions of the form $(k\Delta, x_1(k\Delta), \ldots, x_m(k\Delta)$. Thus it is quite natural to consider partial single trajectory and partial multiple trajectory models.

Given $M \subseteq At \times S$ and $B = a_1, \ldots, a_n, \neg b_1, ..., \neg b_m$, and $\mathbf{p} \in S$, we say that M satisfies B at the generalized position $\mathbf{p}$, written $M \models (B_i, \mathbf{p})$, if $(a_i, \mathbf{p}) \in M$ for $i = 1, \ldots, n$, and $(b_j, \mathbf{p}) \notin M$ for $j = 1, \ldots, m$. For B as above define $B^- = \neg b_1, ..., \neg b_m$. Note that if B_i is empty. then we assume that $M \models (B_i, \mathbf{p})$ automatically holds.

There are two types of rules in H-ASP programs.

Advancing Rules are of the form

$$\frac{B_1; B_2; \ldots; B_r : A, O}{a} \tag{1}$$

where A is an algorithm, each B_i is of the form $a_1^{(i)}$,...,$a_{n_i}^{(i)}$,$\neg b_1^{(i)}$,...,$\neg b_{m_i}^{(i)}$ where $a_1^{(i)}$,....$a_{n_i}^{(i)}$, $b_1^{(i)}$,...,$b_{m_i}^{(i)}$, and a are atoms, and $O \subseteq S^r$ is such that if $(\mathbf{p}_1, \ldots, \mathbf{p}_r) \in O$, then $t(\mathbf{p}_1) < \ldots < t(\mathbf{p}_r)$, $A(\mathbf{p}_1, \ldots, \mathbf{p}_r) \subseteq S$, and for all $\mathbf{q} \in A(\mathbf{p}_1, \ldots, \mathbf{p}_r)$, $t(\mathbf{q}) > t(\mathbf{p}_r)$. Here and in the next rule, we allow n_i or m_i to be equal to 0 for any given i. Moreover, if $n_i = m_i = 0$, then B_i is empty and we automatically assume that B_i is satisfied by any $M \subseteq At \times S$. We shall refer to O as the *constraint set* of the rule and the algorithm A as the *advancing algorithm* of the rule. The idea is that if $(\mathbf{p}_1, \ldots, \mathbf{p}_r) \in O$ and for each i, B_i is satisfied at the general position $\mathbf{p}_i$, then the algorithm A can be applied to $(\mathbf{p}_1, \ldots, \mathbf{p}_r)$ to produce a set of generalized positions O' such that if $\mathbf{q} \in O'$, then $t(\mathbf{q}) > t(\mathbf{p}_r)$ and $(a, \mathbf{q})$ holds. Advancing rules formalize reasoning steps in which new sets of parameter values are derived.

For instance consider the movement of an object through the 3 dimensional space $\mathbb{R}^3$ with a constant velocity $v = (v_1, v_2, v_3)$ where the parameter space S is defined so that $T(S) = \{k\Delta : k \in \mathbb{N}\}$. Then we can use the following rule to model object's movement: $\frac{:A,S}{T}$, where $A((t, x_1, x_2, x_3)) = \{(t + \Delta,\ x_1 + v_1 \cdot \Delta,\ x_2 + v_2 \cdot \Delta,\ x_3 + v_3 \cdot \Delta)\}$. Here T is a place holder atom for "true". That is, since an advancing rule requires a conclusion, "T" is used to fulfill the requirement and has no significance otherwise.

Stationary rules are of the form

$$\frac{B_1; B_2; \ldots; B_r : H, O}{a} \tag{2}$$

where each B_i is of the form $a_1^{(i)}$,...,$a_{n_i}^{(i)}$, $\neg b_1^{(i)}$,...,$\neg b_{m_i}^{(i)}$ where $a_1^{(i)}$,....$a_{n_i}^{(i)}$, $b_1^{(i)}$,...,$b_{m_i}^{(i)}$ and a are atoms, $O \subseteq S^r$ is such that if $(\mathbf{p}_1, \ldots, \mathbf{p}_r) \in O$, then $t(\mathbf{p}_1) < \cdots < t(\mathbf{p}_r)$, and H is a Boolean algorithm such that for all $(\mathbf{p}_1, \ldots, \mathbf{p}_r) \in O$, $H(\mathbf{p}_1, \ldots, \mathbf{p}_r)$ is defined. We shall refer to O as the *constraint set* of the rule and the algorithm H as the *Boolean algorithm* of the rule.

The idea is that if $(\mathbf{p}_1, \ldots, \mathbf{p}_r) \in O$ and for each i, B_i is satisfied at the generalized position $\mathbf{p}_i$, and $H((\mathbf{p}_1, \ldots, \mathbf{p}_r))$ is true, then $(a, \mathbf{p}_r)$ holds. Stationary rules formalize reasoning steps where new facts are derived about a particular set of parameters that has already been derived. As an example consider the following. Suppose that we are considering a program where $T(S) = \{k\Delta : k = 0, \ldots, n\}$, where we want to derive a single trajectory stable model, and where once an atom A holds at $(k\Delta, x_1(k\Delta), \ldots, x_m(k\Delta)$, then A must hold at all generalized positions of the form $((k + 2j)\Delta, x_1((k + 2j)\Delta), \ldots, x_m((k + 2j)\Delta)$ where $j \geq 1$. To enforce this condition we can use the following stationary rule: $\frac{A; :H, O}{A}$ where for all $O = \{(\mathbf{p}_1, \mathbf{p}_2) : t(\mathbf{p}_1) < t(\mathbf{p}_2)\}$ and for all $(\mathbf{p}_1, \mathbf{p}_2) \in O$, $H((\mathbf{p}_1, \mathbf{p}_2))$ holds iff and only if $\frac{t(\mathbf{p}_2) - t(\mathbf{p}_1)}{\Delta}$ is a positive even number.

A H-ASP program is a collection of rules of the form (1) and (2). Note that H-ASP programs have the following features.

1. H-ASP advancing rules allows to pass parameters over variable length time steps. Moreover, advancing algorithms are quite general so that the output of an advancing algorithm can depend on only some of the parameters in any given generalized position.
2. Reasoning about future events can proceed with only partial information available about past events and present events.
3. Rules in H-ASP can refer to a finite number of sets of parameter values in the past. This means that to make a conclusion about a set of parameter values for a time t, a rule can use information about the sets of parameter values for times $t_1 < t_2 < ... < t_k < t$.
4. H-ASP allows users to perform reasoning when the computations made by the algorithms are imprecise. That is, the advancing algorithm in an advancing rule is set valued. This allows us to consider advancing algorithms that use random bits or advancing algorithms that give approximate solutions.
5. H-ASP provides an indirect mechanism by which algorithms can specify values for only some of the parameters. This is accomplished when some of the parameters of generalized positions in an output set of an advancing algorithm are limited to few values, whereas other parameters are allowed to take all the possible values. This will be illustrated by the following example. Suppose that the parameter space S consists of triples of values (t, x, y), where x ranges over set X, and y ranges over set Y. Our program P consists of two rules, $\frac{: A, S}{set\,(x)}$, $\frac{: B, S}{set\,(y)}$, where $set\,(x)$ and $set\,(y)$ are atoms. Suppose further that an algorithm A can only specify the value for x which is 2 at time t'. Then the output of algorithm should be $\{(t', 2, y) : y \in Y\}$. If algorithm B specifies the value of y which is 1 at time t' then the output of B should be $\{(t', x, 1) : x \in X\}$. Intuitively we would want to select only $\mathbf{p} = (t', 2, 1)$ as a valid generalized position, since only in $\mathbf{p}$ among the generalized positions produced by A and by B both x and y are correctly specified. However besides $\mathbf{p}$ the algorithms will produce many other generalized positions. To restrict the valid generalized positions to $\mathbf{p}$, we could add the following two rules, $\frac{\neg set\,(x) : \text{TRUE}, S}{set\,(x)}$, $\frac{\neg set\,(y) : \text{TRUE}, S}{set\,(y)}$, where TRUE is a Boolean algorithm

that returns 1 on any input. These rules will restrict any stable model to contain only $\mathbf{p}$ as a generalized position with time equal to t'.

Next we define an analogue of stable models for H-ASP programs. To do this, we first must define H-ASP Horn programs and a one-step provability operator for H-ASP Horn programs.

A *H-ASP Horn program* is a H-ASP program that does not contain any negated atoms in At. A *consistent H-ASP Horn program* P is a H-ASP Horn program such that if whenever two pairs of an advancing algorithm and a constraint set, (A, O) and (A', O'), appear in P and $O, O' \subseteq S^r$, then $A \restriction_{O \cap O'} = A' \restriction_{O \cap O'}$. Intuitively, consistent H-ASP Horn programs will be used to derive single trajectory stable models while H-ASP Horn programs without the consistency constraint will be used to derive multiple trajectory stable models.

Let P be a H-ASP Horn program and $I \in S$ be an initial condition. Then the one-step provability operator $T_{P,I}$ is defined so that given $M \subseteq At \times S$, $T_{P,I}(M)$ consists of M together with the set of all $(a, J) \in At \times S$ such that

1. there exists a stationary rule $C = \frac{B_1;B_2;\ldots;B_r:H,O}{a}$ and $(\mathbf{p}_1, \ldots, \mathbf{p}_r) \in O \cap \left(\widehat{M} \cup \{I\}\right)^r$ such that $(a, J) = (a, \mathbf{p}_r)$ and $M \models (B_i, \mathbf{p}_i)$ for $i = 1, \ldots, r$ and $H(\mathbf{p}_1, \ldots, \mathbf{p}_r) = 1$.
2. there exists an advancing rule $C = \frac{B_1;B_2;\ldots;B_r:A,O}{a}$ and $(\mathbf{p}_1, \ldots, \mathbf{p}_r) \in O \cap \left(\widehat{M} \cup \{I\}\right)^r$ such that $J \in A(\mathbf{p}_1, \ldots, \mathbf{p}_r)$ and $M \models (B_i, \mathbf{p}_i)$ for $i = 1, \ldots, r$.

There is a nice subclass of H-ASP programs which are particularly well behaved that we call *Basic H-ASP* (BH-ASP) programs. In a BH-ASP program, we assume that the underlying parameter space S has the property that $T(S)$ is of the form $\{k\Delta : k = 0, \ldots, n\}$ or $\{k\Delta : k \in \mathbb{N}\}$ for some fixed $\Delta > 0$. Moreover, we do not allow the rules to refer to multiple times in the past and we assume all advancing algorithms define functions that just give us information about the next time step. That is, a BH-ASP program consists of collections of the following two types of rules.

Basic Stationary rules are of the form

$$\frac{a_1, ..., a_s, \neg b_1, ..., \neg b_t : O}{a} \tag{3}$$

where $a, a_1, ..., a_s, b_1, ..., b_t \in At$, O is a set of generalized positions in the parameter space S. The idea is that if for a generalized position $\mathbf{p} \in O$, $(a_i, \mathbf{p})$ holds for $i = 1, ..., s$ and $(b_j, \mathbf{p})$ does not hold for $j = 1, ..., t$, then $(a, \mathbf{p})$ holds. Thus stationary rules are typical general logic programming rules relative to a fixed world $W_M(\mathbf{p})$.

Basic Advancing rules are of the form

$$\frac{a_1, ..., a_s, \neg b_1, ..., \neg b_t : A, O}{a} \tag{4}$$

where $a, a_1, ..., a_s, b_1, ..., b_t \in At$, O is a set of generalized positions in the parameter space S, and A is an algorithm such that for any generalized position $\mathbf{p} \in O$, $A(\mathbf{p})$ is defined and is an element of S. Here as in H-ASP advancing rules, A can be any sort of algorithm that might require solving a differential or integral equation, solving a set of linear equations or linear programming equations, running a program or automaton, etc. However, we assume that if $\mathbf{p} = (k\Delta, x_1, \ldots, x_m) \in O$, then $A(\mathbf{p})$ is of the form $((k+1)\Delta, y_1, \ldots, y_m)$ for some $y_1, \ldots, y_m$.

In that case, we can prove the following result for consistent BH-ASP programs by induction.

▶ **Theorem 1.** *Let $I = (0, x_1(0), \ldots, x_m(0))$ be an initial condition.*

1. *Let P be a consistent BH-ASP Horn program over the parameter space S and set of atoms At. Then the least model of P is a partial single trajectory model of the form $\bigsqcup_{k \in V} W_M(k\Delta, x_1(\Delta), \ldots, x_m(k\Delta)) \times \{(k\Delta, x_1(\Delta), \ldots, x_m(k\Delta))\}$ where either $V = \mathbb{N}$ or V is some initial segment of $\mathbb{N}$.*
2. *Let P be a BH-ASP Horn program over parameter space S and set of atoms At. Then if $(a, (k\Delta, x_1, \ldots, x_m))$ is in the least model of P, then for all $1 \leq j \leq k$, there must be an element of the form $(a_j, (j\Delta, x_1^j, \ldots, x_m^j))$ in the least model of P.*

Part 2 of the theorem has a simple interpretation - if the least model contains an element for time $k\Delta$ then it also contains an element for every (discrete) time preceding $k\Delta$.

We define the stable model semantics for general H-ASP programs as follows. Suppose that we are given a H-ASP program P over a set of atoms At and a parameter space S, a set $M \subseteq At \times S$, and an initial condition $I \in S$ such that $t(I) = 0$. Then we form the Gelfond-Lifschitz reduct of P over M and I, $P^{M,I}$ as follows.

1. Eliminate from P all advancing rules $C = \frac{B_1;\ldots,B_r:A,O}{a}$ such that for all $(\mathbf{p}_1, \ldots, \mathbf{p}_r) \in O \cap \left(\widehat{M} \cup \{I\}\right)^r$, there is an i such that $M \not\models (B_i^-, \mathbf{p}_i)$ or $A(\mathbf{p}_1, \ldots, \mathbf{p}_r) \cap \widehat{M} = \emptyset$.
2. If the advancing rule $C = \frac{B_1;\ldots,B_r:A,O}{a}$ is not eliminated by (1), then replace it by $\frac{B_1^+;\ldots,B_r^+:A^+,O^+}{a}$ where for each i, B_i^+ is the result of removing all the negated atoms from B_i, O^+ is equal to the set of all $(\mathbf{p}_1, \ldots, \mathbf{p}_r)$ in $O \cap \left(\widehat{M} \cup \{I\}\right)^r$ such that $M \models (B_i^-, \mathbf{p}_i)$ for $i = 1, \ldots, r$ and $A(\mathbf{p}_1, \ldots, \mathbf{p}_r) \cap \widehat{M} \neq \emptyset$, and $A^+(\mathbf{p}_1, \ldots, \mathbf{p}_r)$ is defined to be $A(\mathbf{p}_1, \ldots, \mathbf{p}_r) \cap \widehat{M}$.
3. Eliminate from P all stationary rules $C = \frac{B_1;\ldots,B_r:H,O}{a}$ such that for all $(\mathbf{p}_1, \ldots, \mathbf{p}_r) \in O \cap \left(\widehat{M} \cup \{I\}\right)^r$, either there is an i such that $M \not\models (B_i^-, \mathbf{p}_i)$ or $H(\mathbf{p}_1, \ldots, \mathbf{p}_r) = 0$.
4. If the stationary rule $C = \frac{B_1;\ldots,B_r:H,O}{a}$ is not eliminated by (3), then replace it by $\frac{B_1^+;\ldots,B_r^+:H|_{O^+},O^+}{a}$ where for each i, B_i^+ is the result of removing all the negated atoms from B_i, O^+ is equal to the set of all $(\mathbf{p}_1, \ldots, \mathbf{p}_r)$ in $O \cap \left(\widehat{M} \cup \{I\}\right)^r$ such that $M \models (B_i^-, \mathbf{p}_i)$ for $i = 1, \ldots, r$ and $H(\mathbf{p}_1, \ldots, \mathbf{p}_r) = 1$.

We then say that M is a *general stable model of P with initial condition I* if $T_{P^{M,I}}(\emptyset) \uparrow \omega = M$.

3 The Example

We will now present the use of these definitions in the Secret Agent 00111 example. In order to keep things simple, our presentation will be restricted to the agent's movement in the Area I of the domain. We will also use the generalized positions (t, d, y, v, s) where d denotes the displacement, i.e. the distance traveled from the top of the mountain, v is the agent's velocity in the direction of displacement, y is the agent's y-coordinate, and $s = 1$ if a shot is heard at time t and $s = 0$ if a shot is not heard at time t. It should be noted that the parameter y will not be used below, however parameter y would be important for modeling the agent's movement in the lake. Since in the lake the agent can choose the steering direction, the y-coordinate will vary according to the steering.

A central consideration in determining the proper constraint sets associated with H-ASP rules is the problem of modeling the interaction between regions. In the case of our example, this means modeling accurately the agent's movements as the agent crosses the boundary between regions. In our example, the issue arises as the agent moves from region 1 of Area I into region 2 of Area I.

Recall that the agent will simply coast down the mountain unless pursued by the evil agents. If the agent is pursued, he will apply additional acceleration of $4m/s^2$ in the direction of the movement. The agent is considered to be pursued during a time interval of Δ seconds if a shot is heard in the beginning of the interval, or if two shots were heard before the start of the interval and the shots were separated by Δ seconds.

For the rest of the presentation we will set $\Delta = 1$. To keep the numbers simple, we shall round the acceleration due to gravity to 10m/s. We have carefully picked the slopes so that we have (3,4,5) right triangle in region 1 and a (7,24,25) right triangle in region 2 so that the acceleration due to gravity in region 1 in the direction of displacement will be $\frac{3}{5}10 = 6m/s^2$ and the acceleration due to gravity in the direction of displacement will be $\frac{7}{25}10 = \frac{14}{5}m/s^2$ in region 2 of Area I. We shall only model the agents behavior at times $t = 0, 1, 2, 3, 4, 5$ and we shall assume that the agent cannot exit Area I within 5 seconds so that we will not concern ourselves with the boundary between Area I and Area II.

The set of atoms is $At = \{\text{PURSUED}, A, T\}$ where PURSUED will indicate that the agent is being pursued, A will indicate that two shots separated by 1 second were heard in the past, and T is a place holder atom.

Given a generalized position $\mathbf{p} = (t, d, y, v, s)$ and an acceleration a let $B(\mathbf{p}, a) = b$ be such that if the agent is at distance d as measured from the top of the mountain in region 1 on the path along the mountain's slope and has initial velocity v and he accelerates continuously with the acceleration a in the direction of his movement, then at time $t + b$ he will reach the boundary of region 1 of Area I which is a generalized position with $d = 50m$. That is, we want $50 = \frac{a}{2}b^2 + vb + x$ so that $B((t, x, y, v, s), a) = \frac{-v+\sqrt{v^2-2a(x-50)}}{a}$. Thus $O_1 = \{(t, x, y, v, s) : 0 \leq d < 50 \text{ and } B((t, x, y, v, s), 6) \geq 1\}$ is the set of generalized positions in region 1 where the agent coasting down hill will not reach the boundary between region 1 and region 2 in < 1 second and $O_2 = \{(t, x, y, v, s) : 0 \leq d < 50 \text{ and } B((t, x, y, v, s), 6) < 1\}$ is the set of generalized positions in region 1 where the agent coasting down hill will reach the boundary between region 1 and region 2 in < 1 second. Similarly, $O_3 = \{(t, x, y, v, s) : 0 \leq d < 50 \text{ and } B((t, x, y, v, s), 10) \geq 1\}$ is the set of generalized positions in region 1 where the agent accelerating at $4m/s^2$ in addition to gravity will not reach the boundary between region 1 and region 2 in < 1 second and $O_4 = \{(t, x, y, v, s) : 0 \leq d < 50 \text{ and } B((t, x, y, v, s), 10) < 1\}$ is the set of generalized positions in region 1 where the agent accelerating at $4m/s^2$ in addition to gravity will reach the boundary between region 1 and region 2 in < 1 second. Finally let O_5 be the set of all the generalized positions in region 2.

We will have two types of advancing algorithms. That is, if we are using a constant acceleration a in a region, then by the usual formulas for displacement $d(t) = \frac{a}{2}t^2 + v(0)t + d(0)$ and velocity $v(t) = at + v(0)$ at time t, one can see that our advancing algorithm should be

$$A_a(t, d, y, v, s) = \{(t + 1, \frac{a}{2} + v + d, y, a + v, s) : s \in \{0, 1\}\}.$$

If at $\mathbf{p} = (t, d, y, v, s)$, we know that we will cross from region 1 to region 2 so that we reach the boundary $d = 50$ of region 1 with a velocity of $aB(\mathbf{p}, a) + v$ and then in region 2, our acceleration is a', one can see that our advancing algorithm should be

$$C_{a,a'}(t, d, y, v, s) = \{(t + 1, d', y, v', s) : s \in \{0, 1\}\}.$$

where $d' = \frac{a'}{2}(1 - B(\mathbf{p}, a))^2 + (aB(\mathbf{p}, a) + v)(1 - B(\mathbf{p}, a)) + 50$ and $v' = a'(1 - B(\mathbf{p}, a)) + aB(\mathbf{p}, a) + v$.

We then have the following rules for when our agent is not being pursued by evil agents.

$$\frac{\neg \text{PURSUED} : A_6, O_1}{T} \quad \frac{\neg \text{PURSUED} : C_{6,\frac{14}{5}}, O_2}{T} \quad \frac{\neg \text{PURSUED} : A_{\frac{14}{5}}, O_5}{T}$$

We have the following rules for when our agent is begin pursued by evil agents.

$$\frac{\text{PURSUED} : A_{10}, O_3}{T} \quad \frac{\text{PURSUED} : C_{10,\frac{34}{5}}, O_4}{T} \quad \frac{\text{PURSUED} : A_{\frac{34}{5}}, O_5}{T}.$$

The atom A is to be derived at time t if a shot was heard at time $t - \Delta t$ and at time t. Also atom A is to be derived at time t if A was already derived at time $t - \Delta t$. This is formalized by the following two stationary rules:

$$\frac{: H, G}{A} \quad \frac{A; : \text{TRUE}_2, G}{A}$$

where $G = \{(\mathbf{p}_1, \mathbf{p}_2) \in S^2 : 1 + t(\mathbf{p}_1) = t(\mathbf{p}_2)\}$, $\text{TRUE}_2(\mathbf{p}_1, \mathbf{p}_2) = 1$, and and $H(\mathbf{p}_1, \mathbf{p}_2) = \chi(s(\mathbf{p}_2) - s(\mathbf{p}_1) = 1)$. Here for any statement Q, we let $\chi(Q) = 1$ if Q is true and $\chi(Q) = 0$ if Q is false.

Finally the atom PURSUED is to be derived at time t if a shot is heard at time t or if atom A is derived at time t. Hence, we have the following two stationary rules

$$\frac{: U, S}{\text{PURSUED}} \quad \frac{A : \text{TRUE}_1, S}{\text{PURSUED}}$$

where $U(\mathbf{p}) = \chi(s(\mathbf{p}) = 1)$ and for all $\mathbf{p} \in S$, $\text{TRUE}_1(\mathbf{p}) = 1$.

Let P be the program consisting of the rules described above. We will now consider a particular single trajectory stable model M of P. According to this stable model there is a shot heard at time $t = 0$ then at time $t = 2$ and then at time $t = 3$. We will only consider the elements of the stable model up to and including the time $t = 5$.

Suppose that at time $t = 0$ the agent has displacement $5m$ and initial velocity $2m/s$. That is $I = (0, 5, 0, 2, 1)$. Then the following table describes a stable model M of P.

t	d (m)	y (m)	v (m/s)	s	$W_M(t, d, y, v, s)$
0	5	0	2	1	T, PURSUED
1	$5 + 2 + 5 = 12$	0	$10 + 2 = 12$	0	T
2	$3 + 12 + 12 = 27$	0	$6 + 12 = 18$	1	T, PURSUED
3	$5 + 18 + 27 = 50$	0	$10 + 18 = 28$	1	T, A, PURSUED
4	$\frac{17}{5} + 28 + 50 = \frac{407}{5}$	0	$\frac{34}{5} + 28 = \frac{174}{5}$	0	T, A, PURSUED
5	$\frac{17}{5} + \frac{174}{5} + \frac{407}{6} = \frac{598}{5}$	0	$\frac{34}{5} + \frac{174}{5} = \frac{208}{5}$	0	T, A, PURSUED

M induces the following Gelfond-Lifschitz reduct

$$\frac{:A_6^+, \{(1,12,0,12,0)\}}{T} \quad \frac{\text{PURSUED}:A_{10}^+, \{(0,5,0,2,1),(2,27,0,18,1)\}}{T}$$
$$\frac{\text{PURSUED}:A_{\frac{34}{5}}^+, \left\{(3,50,0,28,1),\left(4,\frac{407}{5},0,\frac{174}{5},0\right),\left(5,\frac{598}{5},0,\frac{208}{5},0\right)\right\}}{T}$$

$$\frac{:H|_{G^+}, G^+}{A} \quad \frac{A; :\text{TRUE}_2|_Y, Y}{A} \quad \frac{:U|_{S^+}, S^+}{\text{PURSUED}} \quad \frac{A:\text{TRUE}_1|_{\widehat{M}}, \widehat{M}}{\text{PURSUED}}$$

where $G^+ = \{((2, 27, 0, 18, 1), (3, 50, 0, 28, 1))\}$, $S^+ = \{(0, 5, 0, 2, 1), (2, 27, 0, 18, 1), (3, 50, 0, 28, 1)\}$, $Y = \{((0, 5, 0, 2, 1), (1, 12, 0, 12, 0)), ((1, 12, 0, 12, 0), (2, 27, 0, 18, 1)), ((2, 27, 0, 18, 1), (3, 50, 0, 28, 1)),$

$((3, 50, 0, 28, 1), (4, \frac{407}{5}, 0, \frac{174}{5}, 0)), ((4, \frac{407}{5}, 0, \frac{174}{5}, 0), (5, \frac{598}{5}, 0, \frac{208}{5}, 0))\}$, and
$\widehat{M} = \{(0, 5, 0, 2, 1), (1, 12, 0, 12, 0), (2, 27, 0, 18, 1), (3, 50, 0, 28, 1), (4, \frac{407}{5}, 0, \frac{174}{5}, 0), (5, \frac{598}{5}, 0, \frac{208}{5}, 0)\}$.

The fact that M is stable model can be easily verified by computing the least model of the Gelfond-Lifschitz reduct of P with respect to M and I.

Remark: We should note that the restriction on O in both extended stationary rules and extended advancing rules that $t(\mathbf{p}_1) < \cdots < t(\mathbf{p}_r)$ can in fact be dropped by slightly modifying our definitions of the one step provability operator and stable models. This could in principle allow one to use rules that refer to both past and future times or to multiple generalized positions that occur at the same time. We put in the restriction $t(\mathbf{p}_1) < \cdots < t(\mathbf{p}_r)$ mainly for ease of presentation and the fact that we did not have space to develop applications of these more extended rules.

4 Conclusion

In this paper, we introduced Hybrid ASP (H-ASP) - an ASP type system to reason about dynamical systems which exhibit both continuous and discrete aspects. The key feature of the system is that it includes advancing rules that incorporate an algorithm to allow for parameter passing from one time step to the next and stationary rules which incorporate an auxiliary algorithm to allow non-logical checking that govern the applicability of the rule. We then defined an analogue of the stable model semantics for H-ASP by defining an appropriate analogues of the one-step provability operator for Horn programs and the Gelfond-Lifschitz reduct of normal logic programs.

We envision that an actual implementation of a H-ASP solver will require that the system makes calls to other modules. That is, the algorithms that are part of advancing rules or extended stationary rules are allowed to be any sort of algorithms which require solving a differential or integral equation, solving a set of linear equations or linear programming equations, running a program or automaton, etc. Thus a H-ASP-solver should naturally allow calls to specialized software outside the system to run such algorithms. We have implemented preliminary version of our system where the algorithm required solving PDEs associated with the Heat equation (for a discussion of the Heat equation see [4] and [9]). This type of application goes well beyond the simple toy model that we used to illustrate our ideas in this paper and will be the subject of future papers.

We view the extension of the answer set programming paradigm, H-ASP, that we introduced in this paper as a first step for further work that will lead to both theoretical tools used for the modeling and analysis of dynamic systems and for computer applications that simulate dynamical systems. There is considerable work to be done in developing a theory of such programs which is similar to the theory that has been developed for ASP programs. For example, a careful analysis of the complexity of the stable models of H-ASP programs as a function of the complexity of the advancing and Boolean algorithms in the program needs to be done. One can ask under what circumstances are there analogues of the forward chaining algorithm of [7] for H-ASP programs. One can consider more extended sets of rules that allow for partial parameter passing or allow different rules to instantiate disjoint sets of parameters for the next time step. These issues will be pursued in later papers. Nevertheless, we believe that the point of view of thinking of rules as general input-output devices has the potential for many new applications of ASP techniques.

References

1 Sandeep Chintabathina. 2010. Towards Answer Set Programming Based Architectures for Intelligent Agents. *PhD thesis, Texas Tech University.*

2 M. Gelfond and V. Lifschitz. 1998. Action languages. *Electronic Transactions on AI*, 3(16).

3 David L. George, Randall J. LeVeque. 2006. Finite Volume Methods and Adaptive Refinement for Global Tsunami Propagation and Local Inundation. *Science of Tsunami Hazards,* Vol. 24, No. 5, pp. 319-328.

4 Gustafsson, Kreiss, Oliger. 1995. Time Dependent Problems and Difference Methods. *John Wiley & Sons, Inc.*

5 S. M. Jachec, O. B. Fringer, M. G. Gerritsen, R. L. Street. 2006. Numerical Simulation of Internal Tides and the Resulting Energetics within Monterey Bay and the Surrounding Area. *Geophysical Research Letters,* Vol. 33, L12605, doi: 10.1029/2006GL026314.

6 Kerckhoffs, Neal, Gu, Bassingthwaighte, Omens, McCulloch. January 2007. Coupling of a 3D Finite Element Model of Cardiac Ventricular Mechanics to Lumped Systems Models of the Systemic and Pulmonic Circulation. *Annals of Biomedical Engineering*, Vol. 35, No. 1, pp. 1-18.

7 W. Marek, A. Nerode, and J.B. Remmel. 1999 Logic programs, well orderings, and forward chaining. Annals of Pure and Applied Logic Vol. 96 , 231-276.

8 Reiter. Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems.2001. *The MIT Press.*

9 John C. Strikwerda. 2004. Finite Difference Schemes and Partial Differential Equations. Second Edition. *SIAM.*

10 Richard S. Sutton, Andrew G. Barto. Reinforcement Learning. 1998. *The MIT Press.*