Journal of Applied Logic 13 (2015) 676–719

Contents lists available at ScienceDirect

Journal of Applied Logic

www.elsevier.com/locate/jal

Approximate postdictive reasoning with answer set programming

Manfred Eppe [a,*], Mehul Bhatt [b]

[a] *International Computer Science Institute, Berkeley, USA*
[b] *University of Bremen, Germany*

ARTICLE INFO

Article history:
Accepted 29 June 2015
Available online 28 August 2015

Keywords:
Commonsense reasoning
Action and change
Epistemic reasoning
Answer set programming

ABSTRACT

We present an answer set programming realization of the h-approximation ($\mathcal{HPX}$) theory [8] as an efficient and provably sound reasoning method for epistemic planning and projection problems that involve postdictive reasoning. The efficiency of $\mathcal{HPX}$ stems from an approximate knowledge state representation that involves only a linear number of state variables, as compared to an exponential number for theories that utilize a possible-worlds based semantics. This causes a relatively low computational complexity, i.e, the planning problem is in NP under reasonable restrictions, at the cost that $\mathcal{HPX}$ is incomplete. In this paper, we use the implementation of $\mathcal{HPX}$ to investigate the incompleteness issue and present an empirical evaluation of the solvable fragment and its performance. We find that the solvable fragment of $\mathcal{HPX}$ is indeed reasonable and fairly large: in average about 85% of the considered projection problem instances can be solved, compared to a $\mathcal{PWS}$-based approach with exponential complexity as baseline. In addition to the empirical results, we demonstrate the manner in which $\mathcal{HPX}$ can be applied in a real robotic control task within a smart home, where our scenario illustrates the usefulness of postdictive reasoning to achieve error-tolerance by abnormality detection in a high-level decision-making task.

1. Introduction

Starting with the seminal work by [20], a huge body of work concerning logical formalizations of reasoning about action, change and knowledge has been developed (e.g. [23,5,6,15]). In this field, we are particularly interested in the practical application of *epistemic* action theories, e.g. [21,26,25], which are concerned with reasoning about the *knowledge* of an agent.

Our existing work [8] focuses in particular on postdictive inference in epistemic action theory. It shows that it is possible to solve the projection problem in the context of reasoning about actions and knowledge in polynomial time, whilst allowing for elaboration tolerant postdictive reasoning. The planning problem is solvable in NP.

* Corresponding author.
E-mail addresses: eppe@icsi.berkeley.edu (M. Eppe), bhatt@informatik.uni-bremen.de (M. Bhatt).

http://dx.doi.org/10.1016/j.jal.2015.08.002
1570-8683/

In [8], we describe a theoretical transition-function based approach to model the postdictive $\mathcal{HPX}$ theory. However, we left open the question how to implement our theory, and we also did not account for the usability of the theory in terms of its solvable fragment and its actual application in practice. In this paper, we fill this gap and develop a formalization of $\mathcal{HPX}$ in *answer set programming* (ASP) [2]. We use this implementation to investigate the solvable fragment of the theory and to perform experiments with real robots.

Postdictive reasoning

As stated in [8], "[we] regard postdiction as a form of reasoning accounting for causal relations between temporally ordered states. Postdiction is abductive reasoning, in the sense that it can be used to explain an observation. However, technically, it can be implemented in a deductive manner, as shown throughout this paper. Within an epistemic action theory, postdictive reasoning can be applied to verify action success and to infer new knowledge about the past."

An illustrative example is the Litmus test, which was originally introduced in the context of postdictive reasoning in Moore [21]. It illustrates how postdiction determines a world property, which is not directly perceivable, by observing another world property which is a causal consequence of the imperceivable world property. In the litmus example, the acidic-ness of a liquid is determined by observing the color of the paper.

Example 1 *(The litmus test).* (See [8].) To find out whether a liquid is acidic or alkaline, one can hold a litmus paper into the liquid. If the paper is red, one can postdict that the liquid is acidic, and if the paper is blue, one knows that the liquid is alkaline.

In this paper, we take a particular look at using postdiction for abnormality detection in the application domain of robotic environments. For example, a robot can postdict that a door is closed when it tries to pass the door, but its location sensors tell that it does actually not arrive behind the door. However, as we emphasize in Section 5, robotics is only one example domain where postdiction is important.

Contributions: implementation and application of a postdictive epistemic action theory

The key contributions emanating from the research presented in this paper are (C1–C2):

C1. Implementation of $\mathcal{HPX}$ within the framework of answer set programming.
C2. Empirical evaluation and application of $\mathcal{HPX}$ in a robotics-based smart home environment.

C1. Formalisation of $\mathcal{HPX}$ as an Answer Set Program

The transition function semantics described in [8] provides a clear formal view on the $\mathcal{HPX}$ theory, but it does not provide an actual implementation which is necessary for an empirical evaluation and the application in real robotic environments. To address this, we present $\mathcal{HPX}$ in terms of Answer Set Programming (ASP). The ASP formalization captures the theory in a model-theoretic form, similar to logic programming implementations of the action language $\mathcal{A}$ [13] or the Discrete Event Calculus (DEC) [22]. The well-understood stable model semantics that underlies ASP makes it possible to show that our implementation is actually sound wrt. the operational semantics (Theorem 1). Consequently, the implementation shares all properties of the operational semantics that are mentioned in [8], i.e. native postdiction, linear number of state variables, temporal knowledge, and concurrent acting and sensing.

Our ASP formalization is implemented a set of translation rules and a set of domain independent logic programming rules. The translation rules compile a domain description specified in an action language syntax into a set of domain specific logic programming rules (denoted Γ_{world}), which are then combined with the domain independent part (denoted Γ_{hpx}). The domain independent part is a fixed kernel that covers inertia,