Default Reasoning in Action Domains with Conditional, Non-Local Effect Actions

Hannes Strass[1] and Michael Thielscher[2]

School of Computer Science and Engineering
The University of New South Wales
mit@cse.unsw.edu.au

Abstract

In a recent paper [2], Baumann et al. provided a comprehensive framework for default reasoning in action theories. Yet, the approach was only defined for a very basic class of domains where all actions have only unconditional, local effects. In this paper, we show that the framework can be substantially extended to domains with action effects that are conditional (i.e. are context-sensitive to the state in which they are applied) and non-local (i.e. the range of effects is not pre-determined by the action arguments). Notably, these features can be carefully added without sacrificing important nice properties of the basic framework, such as modularity of domain specifications or the existence of default extensions. In the last part of the paper, we demonstrate how (a subclass of) the framework can be straightforwardly implemented using the answer set programming paradigm.

1 Introduction

Reasoning about actions and non-monotonic reasoning are two important fields of logic in artificial intelligence. Both areas have received considerable attention and have reached remarkable maturity by now. However, a unifying approach that combines the full expressiveness of both fields was still lacking, until a recent paper by Baumann et al. [2] took an important first step into the direction of uniting these two lines of research.

In this paper, we develop a substantial extension of the work of [2], along various dimensions: First, we significantly generalise the theoretical framework to be able to deal with a broad class of action domains. This is important since the expressiveness of established action theories (e.g. the Situation Calculus [8]) is one of their strong points. Second, we make a first proposal on how the framework can actually be implemented. This is no less important, since it yields a conceptually clean way of putting an expressive theoretical reasoning method to practical use.

The paper proceeds as follows. In the next section, we recall the work of Baumann et al. upon which the present paper builds. However, this will be very brief due to a lack of space; for a complete formal background, the reader should consult the original work [2]. Section 3 extends this basic approach by conditional and non-local effects. In the section thereafter, we show how the theoretical framework considered here gives rise to a straightforward implementation in ASP; Section 5 concludes.

2 Default Reasoning about Unconditional, Local Effect Actions

The approach of [2] combines default logic [7] with the unifying action calculus [9]. Domain axiomatisations are viewed as incomplete knowledge bases that are completed by default statements. It takes as input a description of a particular action domain with defaults. Such a

description comprises (1) a domain signature, that defines the vocabulary of the domain; (2) a description of the direct effects of actions; (3) a set of *state defaults* $\Phi \rightsquigarrow \psi$, constructs that specify conditions Φ under which a fluent literal ψ normally holds in the domain.[1] For example, consider the very simple action domain that uses the fluents $\mathsf{SOP}(x)$ (object x is a sheet of paper) and $\mathsf{PA}(x)$ (object x is a paper aeroplane) along with the action $\mathsf{Fold}(x)$ that turns a sheet of paper x into the paper aeroplane x. This action effect is expressed by $\Gamma_{\mathsf{Fold}(x)} = \{\mathsf{PA}(x), \neg\mathsf{SOP}(x)\}$. The single state default $\delta = \mathsf{PA}(z) \rightsquigarrow \mathsf{Flies}(z)$ of the domain simply says that paper aeroplanes usually fly.

Baumann et al. provide a compilation scheme that transforms such a domain description into a default theory. First, the direct effect descriptions are compiled into effect axioms that provide a solution to the notorious frame problem [5] of reasoning about actions. This effect axiom is of the general form

$$\begin{aligned} \mathit{Poss}(A(\vec{x}), s, t) \supset\ & (\forall f)(\mathit{Holds}(f, t) \equiv \mathit{CausedT}(f, A(\vec{x}), s, t)) \wedge \\ & (\forall f)(\neg\mathit{Holds}(f, t) \equiv \mathit{CausedF}(f, A(\vec{x}), s, t)) \end{aligned} \tag{1}$$

where *CausedT* and *CausedF* are macros that enumerate possible causes that determine a fluent's truth value. As of now, these causes are either persistence (staying true resp. false from s to t) or being a direct action effect, that is, $\mathit{CausedT}(f, A(\vec{x}), s, t) \stackrel{\text{def}}{=} \mathit{FrameT}(f, s, t) \vee \mathit{DirT}(f, A(\vec{x}), s, t)$, where $\mathit{FrameT}(f, s, t) \stackrel{\text{def}}{=} \mathit{Holds}(f, s) \wedge \mathit{Holds}(f, t)$ and $\mathit{DirT}(f, A(\vec{x}), s, t) \stackrel{\text{def}}{=} \bigvee_{F(\vec{x}') \in \Gamma_A, \vec{x}' \subseteq \vec{x}} f = F(\vec{x}')$ is a disjunctive enumeration of A's positive effects as specified in Γ_A. The definitions of *CausedF*, *FrameF* and *DirF* are symmetric. In the example domain, the effect axiom for $\mathsf{Fold}(x)$ thus constructed is

$$\begin{aligned} \mathit{Poss}(\mathsf{Fold}(\vec{x}), s, t) \supset\ & (\forall f)(\mathit{Holds}(f, t) \equiv (\mathit{Holds}(f, s) \wedge \mathit{Holds}(f, t)) \vee f = \mathsf{PA}(x) \wedge \\ & (\forall f)(\neg\mathit{Holds}(f, t) \equiv (\neg\mathit{Holds}(f, s) \wedge \neg\mathit{Holds}(f, t)) \vee f = \mathsf{SOP}(x) \end{aligned} \tag{2}$$

Next, the state defaults from the domain description are translated into Reiter defaults, where the special predicate symbols $\mathit{DefT}(f, s, t)$ and $\mathit{DefF}(f, s, t)$ are used to express that a fluent f is normally true (false) at a time point t. For each state default δ, two Reiter defaults are created: δ_{Init}, that is used for default conclusions about the initial time point; and δ_{Poss}, that is used for default conclusions about time points that can be reached via action application. The Reiter translations of our example state default are

$$\frac{\mathit{Init}(t) \wedge \mathit{Holds}(\mathsf{PA}(z), t) : \mathit{Holds}(\mathsf{Flies}(z), t)}{\mathit{Holds}(\mathsf{Flies}(z), t)}$$

and

$$\frac{\mathit{Holds}(\mathsf{PA}(z), t) \wedge \neg\,\mathit{Viol}_\delta(s) : \mathit{DefT}(\mathsf{Flies}(z), s, t)}{\mathit{DefT}(\mathsf{Flies}(z), s, t)}$$

where $\mathit{Viol}_\delta(s) = \mathit{Holds}(\mathsf{PA}(z), s) \wedge \neg\mathit{Holds}(\mathsf{Flies}(z), s)$. δ_{Init} on the left hand side lets us conclude that any z which is known to be a paper aeroplane initially does also fly unless there is information to the contrary. The default δ_{Poss} on the right hand side applies whenever an action starting at time point s and ending at time point t occurred, the object z is a paper aeroplane at t and the default was not violated at the starting time point s. (A default is violated whenever the prerequisite Φ is true at s and yet the consequent ψ is false, which is captured by the macro $\mathit{Viol}_\delta(s)$.) It lets us conclude that z flies unless we know otherwise.

[1] Here, Φ, the *prerequisite*, is a fluent formula; ψ, the *consequent*, being a fluent *literal* also allows to express that a fluent normally does *not* hold in the domain.

For a fluent to be true (or false) by default is then built into the effect axiom by accepting it as another possible "cause" to determine a fluent's truth value, that is, the $\mathit{CausedT}$ macro is updated to $\mathit{CausedT}(f, A(\vec{x}), s, t) \stackrel{\text{def}}{=} \mathit{FrameT}(f, s, t) \vee \mathit{DirT}(f, A(\vec{x}), s, t) \vee \mathit{DefT}(f, s, t)$ with FrameT and DirT as before; symmetrically for $\mathit{CausedF}$.

In order to restore the solution to the frame problem in the presence of defaults, one needs to take care of inapplicable default conclusions. This is done by so-called *default closure axioms*. In our example, the default closure axiom for the fluent literal $\mathsf{Flies}(z)$ is $(\neg \mathit{Holds}(\mathsf{PA}(z), t) \vee \mathit{Viol}_\delta(s)) \supset \neg \mathit{DefT}(\mathsf{Flies}(z), s, t)$ and says that the only way $\mathsf{Flies}(z)$ can be true by default is if the respective default prerequisite is true.

The fundamental notion of the solution to the state default problem by [2] is now a default theory where the incompletely specified world consists of a UAC domain axiomatisation augmented by default closure axioms. The default rules are the automatic translations of user-specified state defaults.

Our example domain is axiomatised by the formula set $\Sigma = \Pi \cup \Upsilon \cup \Sigma_0$, where Π contains the action precondition axiom $\mathit{Poss}(\mathsf{Fold}(x), s, t) \equiv \mathit{Holds}(\mathsf{SOP}(x), s) \wedge s < t$, Υ contains the above-mentioned effect axiom (1) and in the initial state the object P is a sheet of paper and not (yet) a paper aeroplane: $\Sigma_0 = \{\mathit{Init}(s) \supset (\mathit{Holds}(\mathsf{SOP}(\mathsf{P}), s) \wedge \neg \mathit{Holds}(\mathsf{PA}(\mathsf{P}), s))\}$. Enhancing this with the Reiter defaults seen earlier, we get the domain axiomatisation with state defaults $(\Sigma \cup \Sigma_\Delta, \Delta_{\mathit{Init}} \cup \Delta_{\mathit{Poss}})$. Using default knowledge and sceptical reasoning now gives us the desired conclusion that a sheet of paper initially folded into a paper aeroplane indeed flies:[2] $\Sigma \cup \Sigma_\Delta \mid\!\!\sim^{\mathit{scept}}_{\Delta_{\mathit{Init}} \cup \Delta_{\mathit{Poss}}} (\mathit{Init}(t_0) \wedge \mathit{Poss}(\mathsf{Fold}(\mathsf{P}), t_0, t_1)) \supset \mathit{Holds}(\mathsf{Flies}(\mathsf{P}), t_1)$.

3 Default Reasoning about Conditional, Non-Local Effect Actions

We first investigate how the framework of [2] can be extended to conditional effect actions. As we will show, there is subtle interdependence between conditional effects and default conclusions, which require a revision of the Reiter defaults constructed there. We begin by formalising how to represent conditional effects in the domain specification language. Recall that in the unconditional case, action effects were just literals denoting the positive and negative effects. In the case of conditional effects, theses literals are augmented with a fluent formula that specifies the conditions under which the effect materialises.

Definition 1. Let Φ be a fluent formula that may contain equality atoms and ψ be a fluent literal. The pair Φ/ψ is called a *conditional effect expression*. Φ/ψ is called *normalised*, if ψ is of the form $F(\vec{x}_F)$ or $\neg F(\vec{x}_F)$ for some function symbol F : FLUENT and matching sequence of variables $\vec{x}_F$.

Let A be a function into sort ACTION, $\vec{x}_A$ a sequence of variables matching A's arity, $\vec{y}$ be a sequence of variables disjoint from $\vec{x}_A$ and Γ_A be a set of normalised conditional effect expressions with free variables from $\vec{x}_A, \vec{y}$.

$$\mathit{DirT}(f, A(\vec{x}), s, t) \stackrel{\text{def}}{=} \bigvee_{\Phi^+_F/F(\vec{x}', \vec{y}') \in \Gamma_A} (\exists \vec{y}')(f = F(\vec{x}', \vec{y}') \wedge \Phi^+_F[s]) \tag{3}$$

While this extended definition of action effects is straightforward, it severely affects the correctness of default reasoning in the action theory: one cannot naïvely take this updated

[2] The notation $\Sigma \mid\!\!\sim^{\mathit{scept}}_\Delta \varphi$ means that formula φ is contained in every extension of default theory (Σ, Δ).

version of the $DirT, DirF$ macros and use the effect axioms and Reiter defaults as before. In general, whenever there exists a default $\Phi_D \rightsquigarrow \psi$ with conclusion ψ whose negation $\neg\psi$ might be brought about by a conditional effect $\Phi_C/\neg\psi$, one might make the faulty inference $\Phi_D[t] \supset Def(\psi, s, t) \supset \psi[t] \supset \neg Dir(\neg\psi, s, t) \supset \neg\Phi_C[s]$ – that is, from the truth of the default prerequisite at the ending time point t we derive the default conclusion, from which we conclude falsity of the conditional effect, which implies falsity of the effect's precondition in the starting state s. To interrupt this undesired inference chain, we will have to prevent the default conclusion ψ for cases in which we don't know whether the conditional effect $\neg\psi$ will occur.

Definition 2. Let φ, ψ be fluent literals. We say that φ *conflicts with* ψ and write $\varphi \lightning \psi$ if and only if $sign(\varphi) \neq sign(\psi)$ and $|\varphi|$ and $|\psi|$ are unifiable – in this case we set $\theta_{\varphi\psi} \stackrel{\text{def}}{=} mgu(|\varphi|, |\psi|)$.[3]

Let $\delta = \Phi_D \rightsquigarrow \psi_D$ be a state default with free variables $\vec{y}$ and $\varepsilon = \Phi_E/\psi_E$ be a conditional effect expression. We say that δ *conflicts with* ε and write $\delta \lightning \varepsilon$ if and only if $\psi_D \lightning \psi_E$. In this case, $\theta_{\psi_D\psi_E}$ is abbreviated by $\theta_{\delta\varepsilon}$. For a function A into sort ACTION, we accordingly say that δ *conflicts with* A and write $\delta \lightning A$ if there is a conditional effect expression $\varepsilon \in \Gamma_A$ such that $\delta \lightning \varepsilon$.

For each state default that has potential conflicts with action effects, we create several Reiter defaults that incorporate prevention of conflicts: one for each conflicting action, and one for the case where a non-conflicting action happens. All of these default rules extend the previous default prerequisite $Pre_\delta(s,t) \stackrel{\text{def}}{=} \Phi_D[t] \wedge \neg Viol_\delta(s)$ by additional constraints. The Reiter default δ^A_{Poss} is the variant of δ that is safely applicable whenever the conflicting action $A(\vec{x})$ is executed. The first additional prerequisite $Poss(A(\vec{x}), s, t)$ restricts the default rule to cases where a specific instance $A(\vec{x})$ of the conflicting action happens. The last prerequisite ensures that whenever the rule instance at hand might conflict with an action effect, then the respective instance of the effect's precondition is known to be false (and hence the conflict for this particular instance cannot arise).

$$\delta^A_{Poss} \stackrel{\text{def}}{=} \left(\frac{Pre_\delta(s,t) \wedge Poss(A(\vec{x}), s, t) \wedge (\forall)\neg\Phi_E[s] : Def(\psi_D, s, t)}{Def(\psi_D, s, t)} \right) \theta_{\delta\varepsilon}$$

The Reiter default δ_{Poss} is the variant of δ that is safely applicable whenever no conflicting action is executable. In particular, if no conflicting action exists, this definition of δ_{Poss} is equivalent to the one of Baumann et al. without incorporation of conflict prevention.

$$\delta_{Poss} \stackrel{\text{def}}{=} \frac{Pre_\delta(s,t) \wedge Impossible_\delta(\vec{y}, s, t) : Def(\psi_D, s, t)}{Def(\psi_D, s, t)}$$

$$Impossible_\delta(\vec{y}, s, t) \stackrel{\text{def}}{=} \bigwedge_{\substack{A:\text{ACTION},\\ \varepsilon\in\Gamma_A, \delta\lightning\varepsilon}} (\forall\vec{z})(|\psi_D| = |\psi_E|\,\theta_{\delta\varepsilon} \supset \neg Poss(A(\vec{x}), s, t)\theta_{\delta\varepsilon})$$

The macro $Impossible_\delta(\vec{y}, s, t)$ guarantees that all action instances that are conflicting with the actual default instance $\delta(\vec{y})$ at hand are indeed inapplicable.

In our example domain, the action $\mathsf{Fold}(x)$ had only local effects, that is, the set of objects that is affected by the action was somewhat fixed. In general, this is a restriction because it can make the specification of certain actions at least cumbersome or utterly impossible, e.g. actions that affect a vast number of (or all of the) domain elements at once. Our current definitions of effect axioms and default rules can however also cope with the more general case of non-local effects.

[3] The operator $|\cdot|$ extracts the affirmative component of a fluent literal, that is, $|\neg f| = |f| = f$.

The solution to the state default problem for the more general class of action domains considered in this paper is now essentially the solution of [2], updated with our new versions of effect axioms and Reiter defaults. This extension is particularly well-behaved: the increase in expressiveness comes at practically no cost, since the existence of extensions for domain axiomatisations with state defaults can still be guaranteed. Additionally, it is easy to see that the domain specifications provided by the user are still modular: different parts of the specifications, such as conditional effect expressions and state defaults, are completely independent of each other from a user's point of view. Yet, the intricate semantic interactions between them are correctly dealt with.

4 Implementation

Focusing on the formal foundations of integrating default reasoning with reasoning about actions, our work thus far has stayed entirely on the theoretical level. In this section, we describe a first implementation of our theory on the basis of an old result on default logic and stable models, combined with modern implementations of the latter by means of Answer Set Programming (ASP) [3].

In [4], Marek and Truszczyński consider the following translation from normal logic programs into default logic: every pure Horn clause $\mathtt{p} \leftarrow \mathtt{q_1}, \ldots, \mathtt{q_m}$ in the program is mapped to the implication $q_1 \wedge \ldots \wedge q_m \supset p$, and every other clause $\mathtt{p} \leftarrow \mathtt{q_1}, \ldots, \mathtt{q_m}, \neg\mathtt{r_1}, \ldots, \neg\mathtt{r_n}$ is mapped to the default $q_1 \wedge \ldots \wedge q_m : \neg r_1, \ldots, \neg r_n / p$. Let W be the set of implications and Δ the set of defaults thus obtained, then Theorem 3.7 in [4] establishes a one-to-one correspondence between the extensions of the default theory (W, Δ) and the stable models of the original logic program.

In order to apply this result to obtain an ASP-based encoding of a given default theory, the axioms and defaults need to be of a form that allows to reverse Marek and Truszczyński's translation. This can be achieved with the help of the standard way of adding classical negation to normal logic programs, which introduces additional predicate symbols "$-\mathtt{p}$" for every predicate $\mathtt{p}$ in the language. Given an action domain axiomatisation $(\Sigma \cup \Sigma_\Delta, \Delta_{Init} \cup \Delta_{Poss})$, this amounts to rewriting $\Sigma \cup \Sigma_\Delta$ to a set of Horn implications and $\Delta_{Init} \cup \Delta_{Poss}$ to a set of defaults of the above form. The latter are accompanied by defaults $p \wedge -p : \neg f / f$ for every atom p, with f being a nullary predicate that does not occur elsewhere. These additional defaults translate back to the logic programming clause $\mathtt{f} \leftarrow \mathtt{p}, -\mathtt{p}, \neg\mathtt{f}$, which is the standard way of dealing with classical negation in ASP via so-called *integrity constraints* [3].

Due to lack of space we cannot give the reader an impression of the resulting answer set program by showing the implementation of our example domain; alas, the mappings of the effect axioms, default rules and default closure axioms to program clauses are straightforward. Moreover, the resulting program can directly be used for sceptical query answering using a standard off-the-shelf ASP system, such as [6]: if the ASP together with the negation of a query formula admits no answer set, then the formula is contained in each extension of the original default theory.

5 Discussion

We have presented an extension to a recently introduced framework for default reasoning in theories of actions and change. The extension increases the range of applicability of the framework while fully retaining its desirable properties: we can now deal with context-dependent effects of actions as well as with actions with a potentially global effect range – all the while

domain descriptions have not become significantly more complex, and default extensions of the framework still provably exist. In addition, we provided a method of expressing suitable instances of action domains with defaults as answer set programs, thereby paving the way for a practically usable implementation.

For a discussion of related theoretical work, we have to refer the reader to [2]. Concerning related practical work, there is, to the best of our knowledge, only one implemented action language that supports Reiter-style default reasoning: the Causal Calculator [1] provides a constructor *default* to model static default statements. The constructor however cannot express default prerequisites and is therefore not suitable to implement state defaults in our sense, as we have shown earlier [2].

In the future, we will carry out a formal analysis of the meta-theoretical properties of our extended framework; we will be concerned with further extending the approach to non-deterministic domains; lastly and most importantly, we will focus on developing the implementation concept further with the ultimate goal of a provably correct translation from action domain descriptions to answer set programs.

References

[1] The Causal Calculator, 1997. `http://www.cs.utexas.edu/users/tag/cc/`.

[2] Ringo Baumann, Gerhard Brewka, Hannes Strass, Michael Thielscher, and Vadim Zaslawski. State Defaults and Ramifications in the Unifying Action Calculus. In *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning*, pages 435–444, Toronto, Canada, May 2010.

[3] Michael Gelfond. Answer Sets. In *Handbook of Knowledge Representation*, pages 285–316. Elsevier, 2008.

[4] V. Wiktor Marek and Miroslaw Truszczyński. Stable semantics for logic programs and default theories. In *North American Conference on Logic Programming*, pages 243–256. The MIT Press, 1989.

[5] John McCarthy and Patrick J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Machine Intelligence*, pages 463–502. Edinburgh University Press, 1969.

[6] Potassco. Potsdam answer set solving collection, 2008. Available at `http://potassco.sourceforge.net`.

[7] Raymond Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81–132, 1980.

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

[9] Michael Thielscher. A Unifying Action Calculus. *Artificial Intelligence*, 2010. In press.