

Semantical considerations for a logic of actions: an imperative manifesto

Victor Jauregui*

National ICT Australia and
School of Computer Science & Engineering
The University of New South Wales
Sydney, Australia, 2052.
vicj@cse.unsw.edu.au

Abstract

This paper proposes an imperative semantics for a logic of actions: through their axiomatisations, actions are interpreted as programs in an imperative programming language. An argument is made for the naturalness of imperative interpretations of actions, and an assessment of the application of this semantics is conducted.

Introduction

Consider a simple example of reasoning about actions: an agent in a building with $n \geq 1$ windows. Expressing and reasoning about statements of fact—the domain of logic—is typically done through *assertions*, such as: “the k -th window is open”. In the absence of further information, under the permissive nature of the logical semantics typically attributed to assertions, from the former assertion one is unable to draw conclusions regarding, for instance, the truth of: “the j -th window is open” (for $j \neq k$): as the former admits models in which the j -th window is closed and some in which it is open, precluding a definite conclusion in favour of either.

The semantics of actions, however, is usually more subtle, especially when the *frame problem* (McCarthy & Hayes 1969) surfaces in connection with incomplete action descriptions. For example, the action which involves: “opening the k -th window” (while leaving all the other windows unaffected), denoted α , can naturally be described by the assertion: “ α opens the k -th window”. Alternatively, α can be identified with the *imperative*: “open the k -th window”. Both descriptions explicitly assert that the action is responsible for opening the k -th window. Nevertheless, although not stated explicitly, the imperative: “open the k -th window”, is naturally taken to implicate that the j -th window (for any $j \neq k$) remains unaffected—in a similar way that, in a typical imperative programming system, the statement: $x \leftarrow x + 1$; represents the machine transformation which explicitly ‘increments the value of variable x by one’, but which implicitly also entails that all other variables remain unchanged. The permissive semantics of assertions like: “ α

opens the k -th window” is unsuitable as, along the lines of the argument presented above, it does not sanction the inference as to whether: “ α opens the j -th window”, for $j \neq k$, holds.

So, while statements such as: “ α opens the k -th window” might be used in a logic of assertions to *concisely* describe an action which opens the k -th window but leaves the others unaffected, its intended semantics is closer to that of the corresponding imperative: “open the k -th window”, as it would be interpreted in a suitable programming system.

The principal claim of this paper is that it is natural to interpret actions, given their axiomatisations, as simple imperatives. This intuition is formalised by furnishing an imperative semantics for a simple fragment of Propositional Dynamic Logic (PDL). Section 2 introduces a machine model used in Section 3 to formulate a semantics of actions based on the definition of *imperative models*—which characterise minimal change reasoning. It has been shown (see, for example, (Lin 1995; McCain & Turner 1995; Thielscher 1997)), that, in certain cases, minimal change reasoning does not suffice, based on which Section 4 develops a causal semantics, extending that of imperative models, in which dependencies between fluents exist. The imperative model semantics emerges as a special case in which fluents do not exhibit (causal) dependencies. Section 5 concludes with a discussion of the results in this paper and further work.

A logic of actions

This section is devoted to the logical language(s) used in this paper to represent, and reason about actions. Three languages are introduced: the first is a language $\mathcal{A}$ of *action terms*, or *action identifiers*, consisting of an, at most countable, alphabet of atomic action terms A . The only action connective considered is ‘;’, denoting composition. $\mathcal{A}$ is defined inductively:

- $A \subseteq \mathcal{A}$;
- if $\alpha, \beta \in \mathcal{A}$ then $\alpha; \beta \in \mathcal{A}$.

The second, and principal language $\mathcal{L}$, which makes reference to $\mathcal{A}$, is one from which assertions, regarding the actions $\alpha \in \mathcal{A}$, can be made. The intuition is that the *formulae*: $\varphi \in \mathcal{L}$, are used to describe the properties of actions: $\alpha \in \mathcal{A}$, such as: “ α opens the k -th window”. $\mathcal{L}$ consists of

*The author would like to thank the National ICT Australia, Ltd. (NICTA) for their support of this research.
Copyright © 2006, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

a simple fragment of PDL (c.f., (Pratt 1976)), sufficient for the present discussion.

Let F be a finite set of atomic propositions, called *fluents*, then $\mathcal{L}$ is inductively defined as follows:

- $F \subseteq \mathcal{L}$;
- if $\varphi \in \mathcal{L}$ then $\neg\varphi \in \mathcal{L}$;
- if $\varphi, \psi \in \mathcal{L}$ then $\varphi \rightarrow \psi \in \mathcal{L}$;
- if $\alpha \in \mathcal{A}$ and $\varphi \in \mathcal{L}$ then $[\alpha]\varphi \in \mathcal{L}$.

All other connectives are regarded as abbreviations of these. The modal formula: $[\alpha]\varphi$, reads as: “after α terminates, φ holds”, where α represents an action, process, or program.

The third language: $\mathcal{P}$, consists of imperatives/programs. The intuition is that an action identifier: α , based on its description in $\mathcal{L}$, will correspond to some program in $\mathcal{P}$, which constitutes the action’s *imperative interpretation*. Before proceeding further, the underlying machine: $\mathcal{M}$; on which $\mathcal{P}$ operates must be defined; its operation is as follows:

$\mathcal{M}$ consists of a one-way infinite tape (like a Turing machine tape), with tape squares numbered $1, 2, \dots$, and a program segment (possibly another tape) containing some program: $p \in \mathcal{P}$. $\mathcal{M}$ proceeds by reading an instruction from its program segment and modifying the tape accordingly; the only primitives it can perform on the tape are, for any $i \geq 0$, setting the i -th tape square to 0 or 1.

The language: $\mathcal{P}$, which acts on $\mathcal{M}$, is inductively defined:

- $\lambda \in \mathcal{P}$;
- the pair $(k, b) \in \mathcal{P}$, for every $k \in \mathbb{N}^+$, $b \in \{0, 1\}$;
- if $p, q \in \mathcal{P}$ then $p; q \in \mathcal{P}$.

The *instructions*, or *primitive operations*, or sometimes just *primitives*, of $\mathcal{P}$ are λ and pairs (k, b) , for each $k \in \mathbb{N}^+$, and $b \in \{0, 1\}$. λ denotes the *null-op* (the operation which does nothing), while (k, b) denotes the operation of writing symbol b at tape square k ; these will be defined formally later.

The members of $\mathcal{P}$ are called *programs*. It follows from the definition above that a program: p ; comprises a finite sequence of primitives: $p_0; p_1; \dots; p_n$.¹ The *complexity* of program p is denoted $|p|$, and is defined inductively by:²

$$|\lambda| = 0 \quad |(k, s)| = 1 \quad |p; q| = |p| + |q|.$$

A *configuration* of the machine $\mathcal{M}$ (sometimes referred to as an $\mathcal{M}$ -configuration) is any possible finite sequence of symbols, represented as a string, that can appear on the machine’s tape: that is, a member of $\{0, 1\}^*$. The operation of the machine is defined inductively as follows:

Execution of a program $p_0; p_1; \dots; p_n$, begins at step $i = 0$ executing instruction p_0 in the initial configuration w_0 . If instruction p_i is executed at step i then instruction p_{i+1} is executed at step $i + 1$. The program terminates at the first step $i > n$.

¹If w is a sequence or string then w_i for $i \in \text{dom}(w)$ denotes the i -th element in w .

²This is inspired by the concepts of Kolmogorov complexity (Li & Vitanyi 1997).

The semantics of an instruction i is determined by how it transforms a configuration w to produce the configuration w' , denoted: $w \xrightarrow{i} w'$. Using this notation, the operation of the instructions λ and (k, b) is given by:

$$\begin{aligned} w &\xrightarrow{\lambda} w \\ w &\xrightarrow{(k, b)} w' \end{aligned} \quad \text{where } w'_j = \begin{cases} b & \text{if } j = k; \\ w_j & \text{otherwise.} \end{cases}$$

In words, λ has no effect (i.e., it performs the identity transformation), while the instruction (k, b) writes b at tape square k (overwriting the previous value) but leaves all the other tape squares unaffected.

A *computation* (in $\mathcal{M}$) of program: $p \in \mathcal{P}$; (with $|p| = n$) from initial configuration w_0 , is a sequence of configurations: $w_0, w_1, \dots, w_n$, such that $w_i \xrightarrow{p_i} w_{i+1}$ for each $i = 0, 1, \dots, n-1$. Given initial configuration w_0 , a computation is unique ($\mathcal{M}$ is deterministic). The *transition* corresponding to the computation of p from w_0 , denoted $w_0 \xrightarrow{p} w_n$, constitutes the composition of each $w_i \xrightarrow{p_i} w_{i+1}$, for $0 \leq i < n$. As computations are deterministic the final configuration v of a transition $w \xrightarrow{p} v$ is naturally represented functionally as $p(w) = v$.

On the choice of machine model

While the machine model: $\mathcal{M}$; and the corresponding language $\mathcal{P}$ used to program it are both very simple, this choice of model exhibits a number of desirable features for reasoning about action. Firstly, the instructions allow for random access to the tape, which overcomes some of the difficulties encountered when Turing machines, with their local primitives, and generality, are used for a similar purpose (see (Jauregui, Pagnucco, & Foo 2004)). $\mathcal{M}$ still affects only local changes, in the sense that it only writes to one tape square in a single instruction.

Ultimately, the final justification is that this formulation allows for the results which appear subsequently in this paper, and which accord well with the motivations of commonsense reasoning about action. Further discussion will be postponed until these results have been presented.

Action semantics

With the underlying framework out of the way, this section focuses on the development of a logical semantics of action.

Let $\mathcal{L}$ be a language containing $n > 0$ fluents: $F = \{f_1, \dots, f_n\}$; and some, at most countable, set of atomic action identifiers: A ; furnishing an action language: $\mathcal{A}$; as before. An $\mathcal{L}$ -structure is a triple $(\mathcal{W}, \nu, \mu)$; where $\mathcal{W}$ is some set of abstract possible worlds, $\nu : F \rightarrow 2^{\mathcal{W}}$, termed a *valuation*, assigns to each fluent the set of possible worlds in which it holds, and $\mu : \mathcal{A} \times \mathcal{W} \rightarrow \mathcal{P}$ maps an action term, in a given world, to a program in $\mathcal{P}$. $\mu(\alpha, w)$ is referred to as α ’s *imperative interpretation* in w , and is further constrained to programs: $p \in \mathcal{P}$; such that $p(w) \in \mathcal{W}$.

To meaningfully assign a program $\mu(\alpha, w)$ to α , it must be possible to associate with each $w \in \mathcal{W}$ an $\mathcal{M}$ -configuration. A natural way to do this is to use ν to construct a string w

such that:³

$$w_k = \begin{cases} 1 & \text{if } w \in \nu(f_k); \\ 0 & \text{otherwise.} \end{cases}$$

A *standard* $\mathcal{L}$ -structure is an $\mathcal{L}$ -structure for which the set of possible worlds $\mathcal{W}$ is (isomorphic to) a subset of n -bit strings (given $|F| = n$), each corresponding to an $\mathcal{M}$ -configuration; that is, $\mathcal{W} \subseteq \{0, 1\}^n$. Because the possible worlds in a standard structure carry the information regarding ν , when referring to a standard structure ν will often be omitted: $(\mathcal{W}, \mu)$. For notational convenience, when the language $\mathcal{L}$, and hence F and $n = |F|$, is understood, then μ alone will be used to identify the structure.

The satisfiability relation between a model μ , a formula φ and a world $w \in \mathcal{W}$, denoted $\mu \models_w \varphi$, is inductively defined as follows:

$$\begin{aligned} \mu \models_w f & \text{ if } w \in \nu(f); & (f \in F) \\ \mu \models_w \neg \varphi & \text{ if } \mu \not\models_w \varphi; \\ \mu \models_w \varphi \rightarrow \psi & \text{ if when } \mu \models_w \varphi, \text{ then } \mu \models_w \psi; \\ \mu \models_w [\alpha]\varphi & \text{ if } \mu \models_{\mu(\alpha, w)} \varphi. \end{aligned} \quad (1)$$

Moreover, μ is a model of φ if for every $w \in \mathcal{W}$, $\mu \models_w \varphi$. If $\Gamma \subseteq \mathcal{L}$ is a set of formulae then μ is a model of Γ if it is a model of every $\varphi \in \Gamma$; that is $\mu \models \varphi$, for every $\varphi \in \Gamma$. All structures must interpret actions according to the corresponding *standard model* criterion of PDL; namely:⁴

$$\mu(\alpha; \beta, w) = \mu(\alpha, w); \mu(\beta, \mu(\alpha, w)(w)). \quad (2)$$

Let $\Gamma \subseteq \mathcal{L}$ be a set of formulae, and $\varphi \in \mathcal{L}$ a formula, then φ is a *logical consequence* of Γ , denoted $\Gamma \models \varphi$, if every standard model of Γ is a model of φ .

Action theories

Consider again the windows example that was outlined earlier. The language for this example consists of fluents $F = \{\text{open}(w_j) \mid 1 \leq j \leq n\}$, for $n > 1$, designating a fluent to capture whether or not each window is open; and a single atomic action $A = \{a\}$, which is intended to correspond to the action of: “opening the k -th window”. For this example all standard models have $\mathcal{W} = \{0, 1\}^n$.

A natural axiomatisation for action a in this example is: $\Gamma = \{[a]\text{open}(w_k)\}$, indicating that: “action a opens the k -th window”. Consider the model with $\mu(a, w) = (k, 1)$, for each $w \in \mathcal{W}$; i.e., the model in which a is interpreted, for any window configuration/world, as the program which opens (only) the k -th window: that is, setting the k -th tape square to 1.

For this model, $\mu \models \neg \text{open}(w_j) \rightarrow [a]\neg \text{open}(w_j)$, for any $j \neq k$; that is, in any initial world in which the j -th window is closed, after opening the k -th window, the j -th

window remains closed. In this respect, this model is representative of the imperative semantics outlined earlier, reflecting the fact that Γ is to be interpreted as the imperative: “open the k -th window”.

However, the semantics presented so far is unable to reach the (desirable) conclusion that $\Gamma \models \neg \text{open}(w_j) \rightarrow [a]\neg \text{open}(w_j)$, for every $j \neq k$. To see this consider the model with: $\mu(a, w) = (k, 1); (j, 1)$, for each $w \in \mathcal{W}$; that is, a is interpreted as opening both the k -th and j -th window. Indeed, any interpretation for a which includes the instruction $(k, 1)$ will produce a model of Γ . This permissiveness, as with the semantics of assertions, makes it impossible to conclude $\neg \text{open}(w_j) \rightarrow [a]\neg \text{open}(w_j)$, for any $j \neq k$, from Γ .

Imperative models

Interpreting $[a]\text{open}(w_k)$ imperatively by: “open the k -th window”, as with an imperative programming language, renders the additional implication: “... but leave the other x ’s as they were”,⁵ both natural and intuitive. More generally, this involves attributing to an action identifier: α ; the most primitive, or *simplest* interpretation p , measured by its complexity $|p|$, consistent with its axiomatisation. In this case the most primitive interpretation consistent with $[a]\text{open}(w_k)$ involves assigning to a the program $(k, 1)$. There is, however, one small consideration: if $\text{open}(w_k)$ holds in the current world then nothing needs to be done in order to open the window; for which λ is the most primitive and natural interpretation; in general, the interpretation of an action depends on the world in which it is performed. This is precisely why μ is a function of the action $\alpha \in \mathcal{A}$ and the world $w \in \mathcal{W}$. The following definition formalises this intuition.

An *imperative model* μ , of a formula $\varphi \in \mathcal{L}$, in a world $w \in \mathcal{W}$, is a model of φ which, for each $\alpha \in \mathcal{A}$, there is no model μ' of φ which yields $|\mu'(\alpha, w)| < |\mu(\alpha, w)|$. In words: an imperative model of φ , in a world w , is a standard model which admits no simpler interpretation, for each α , in w .

If μ is an imperative model of φ , in w , it is denoted $\mu \models_w^{\mathcal{I}} \varphi$. Moreover, μ is an imperative model of φ , denoted $\mu \models^{\mathcal{I}} \varphi$, if it is an imperative model of φ in every $w \in \mathcal{W}$. A model μ is an imperative model of a set of formulae $\Gamma \subseteq \mathcal{L}$, denoted $\mu \models^{\mathcal{I}} \Gamma$, if it is an imperative model of each $\varphi \in \Gamma$. A formula φ is an *imperative consequence* of a set of formulae Γ , denoted $\Gamma \models^{\mathcal{I}} \varphi$ if every imperative model of Γ is a model of φ .

For the windows example it can be observed that the model:

$$\mu(a, w) = \begin{cases} (k, 1) & \text{if } w_k \neq 1; \\ \lambda & \text{otherwise,} \end{cases} \quad (3)$$

is an imperative model of Γ . In fact, a standard model can be shown to be unique modulo the concatenation of λ ’s and permutations of instructions. In view of this, when a model is given it will often represent the equivalence class of all such interpretations.

⁵Here the x ’s stand for windows.

³That is, with each world: $w \in \mathcal{W}$; is associated the string with a 1 at the k -th position if the k -th fluent: f_k ; holds, and 0 otherwise.

⁴This simply states that, if α is mapped to program p in world w and β is mapped to program q in world $p(w)$, then $\alpha; \beta$ must be mapped to $p; q$ in w . In this way, when the mapping $\mu(a, w)$ is determined for all atomic action identifiers $a \in \mathcal{A}$, then $\mu(\alpha, w)$ is uniquely determined for all $\alpha \in \mathcal{A}$.

Assessment

A preliminary characterisation of this semantics can already be made, but before supplying this, an assessment is presented based on collections of pairs (Γ, φ) with respect to which the present semantics, that is, the relation $\Gamma \models^{\mathcal{I}} \varphi$, acquits itself intuitively.

Simple examples The first Γ is that of the windows example discussed so far. From the discussions in the preceding section it can be observed that, for any $j \neq k$, $\Gamma \models^{\mathcal{I}} \neg \text{open}(w_j) \rightarrow [a] \neg \text{open}(w_j)$. The proof involves consideration of the two cases in (3), and the observation that the semantics given by (1) is determined by a program's transitions; making it invariant under the addition of superfluous λ 's to an action's interpretation and the permutation of instructions.

Another example is the Yale Shooting Problem (YSP) (taken from (Hanks & McDermott 1987)). In this example, there is a turkey and a gun, which may or may not be alive (fluent a) and loaded (fluent l), respectively. The actions are: to "Load" (Lo) the gun, after which it will be loaded; to "Shoot" (Sh) the turkey with the gun, after which, if the gun was loaded the turkey will be dead; and finally the action "Wait" (Wa) which involves doing nothing, and hence requires no axiomatisation: whence, $A = \{Lo, Wa, Sh\}$, and $F = \{a, l\}$, with axiomatisation $\Gamma = \{[Lo]l, l \rightarrow [Sh] \neg a\}$. For perspicuity, a mapping between a fluent: $f \in F$ and its corresponding tape square in a standard structure will be used. So, for instance, the program instruction $(f, 1)$ denotes setting the tape square corresponding to fluent f to 1.

It is expected that after loading, waiting, and then shooting the gun, the turkey should be dead; that is, $\Gamma \models^{\mathcal{I}} [Lo; Wa; Sh] \neg a$. Consider an imperative model μ , then:⁶

$$\begin{aligned} \mu(Lo, w) &= \begin{cases} (l, 1) & \text{if } w \notin \nu(l); \\ \lambda & \text{otherwise,} \end{cases} \\ \mu(Wa, w) &= \lambda \quad \forall w \in \mathcal{W}; \\ \mu(Sh, w) &= \begin{cases} (a, 0) & \text{if } w = al; \\ \lambda & \text{otherwise.} \end{cases} \end{aligned}$$

By the standard model property (2), the following obtains:

$$\mu(Lo; Wa; Sh, w) = \begin{cases} (l, 1); \lambda; \lambda & \text{if } w = \overline{al}; \\ (l, 1); \lambda; (a, 0) & \text{if } w = al; \\ \lambda; \lambda; \lambda & \text{if } w = \overline{al}; \\ \lambda; \lambda; (a, 0) & \text{if } w = al. \end{cases}$$

For each of these interpretations it can readily be observed, by inspection, that, in the given state w , the corresponding program p (e.g., $p = (l, 1); \lambda; \lambda$ in $w = \overline{al}$) yields $\mu \models_{p(w)}^{\mathcal{I}} \neg a$. Consequently, for every $w \in \mathcal{W}$, $\mu \models_w [Lo; Wa; Sh] \neg a$, and hence $\Gamma \models^{\mathcal{I}} [Lo; Wa; Sh] \neg a$.

Examples with logical dependencies The windows and YSP examples do not impose any restrictions on possible worlds: no logical constraints exist between the fluents. A modification of the YSP, taken from (McCain & Turner 1995), introduces such dependencies.

⁶Here, the notation $w = a\overline{l}$, denotes that $w \in \nu(a)$ and $w \notin \nu(l)$; i.e., $w \in \nu(a) \cap \overline{\nu(l)}$.

In this example the gun is always loaded but the turkey, in addition to being alive (a), may be walking (w). The dependency/constraint exists which states that only walking turkeys may be alive. This yields: $A = \{Sh\}$, $F = \{a, w\}$ and $\Gamma = \{[Sh] \neg a, w \rightarrow a\}$. Standard models of Γ will have $\mathcal{W} = \{\overline{aw}, a\overline{w}, aw\}$ —note that $\overline{aw} \notin \mathcal{W}$, as it violates the constraint $w \rightarrow a$.

The point needs to be made that, while the semantics permits only programs with transitions which take worlds from $\mathcal{W}$ to other worlds in $\mathcal{W}$, it does not prohibit transitions involving computations which traverse (but do not terminate in) configurations not in $\mathcal{W}$.

Let μ be an imperative model for Γ , satisfying:

$$\begin{aligned} \mu(Sh, \overline{aw}) &= \lambda; \\ \mu(Sh, a\overline{w}) &= (a, 0); \\ \mu(Sh, aw) &= (a, 0); (w, 0). \end{aligned}$$

The corresponding transitions induced by these interpretations p , are, for:

$$\begin{aligned} p = \lambda : & \quad p(\overline{aw}) = \overline{aw}; \\ p = (a, 0) : & \quad p(a\overline{w}) = \overline{aw}; \\ p = (a, 0); (w, 0) : & \quad p(aw) = \overline{aw}. \end{aligned}$$

Note, in particular, that $\mu(Sh, aw) = (a, 0); (w, 0)$, traverses world $w' = \overline{aw} \notin \mathcal{W}$, which due to the constraint $w \rightarrow a$, induces the additional transformation $(w, 0)$ to ensure that $p(aw) \in \mathcal{W}$.

This analysis indicates that the framework proposed so far furnishes a faithful characterisation of a solution to the frame problem based on the principle of minimal change, overcoming some of the problems encountered by other approaches, e.g., naively applying circumscription (McCarthy 1980) to solve the Yale Shooting Problem (for an overview see (Shanahan 1997)). It has been noted in the literature, however, that sometimes fluents exhibit extra-logical dependencies (see (Lifschitz 1990), (Lin 1995), (McCain & Turner 1995), (Thielscher 1997)); e.g., causal dependencies. A framework for reasoning about action must acquit itself of these to be deemed adequate. The problem will be outlined below; the solution will be left to the next section.

Examples with non-logical dependencies Consider an electrical circuit containing two switches (sw_1 and sw_2) connected in series to a power source (possibly a battery) and a light (l): $F = \{sw_1, sw_2, l\}$.⁷ The only action considered is: "closing switch sw_1 ", denoted C_1 : $A = \{C_1\}$. The system can be axiomatised by: $\Gamma = \{[C_1]sw_1, sw_1 \wedge sw_2 \leftrightarrow l\}$. It would be desirable to obtain $\Gamma \models^{\mathcal{I}} \neg sw_1 \wedge sw_2 \rightarrow [C_1]l$: for example, in the world $w = \overline{sw_1}, sw_2, \overline{l}$, in which the first switch is off, the second is on, and, therefore, the light is off, if the first switch is closed (switched on) then it should be inferred that the light turns on.

The following can be shown to be an intended imperative model of Γ :

$$\mu(C_1, w) = \begin{cases} \lambda & \text{if } w \in \nu(sw_1); \\ (sw_1, 1) & \text{if } w \notin \nu(sw_2); \\ (sw_1, 1); (l, 1) & \text{otherwise.} \end{cases}$$

⁷This example is taken from (Thielscher 1997).

This model satisfies $\mu \models^{\mathcal{I}} \neg sw_1 \wedge sw_2 \rightarrow [C_1]l$. However, it can be shown that the following is an anomalous/unintended model of Γ :

$$\mu'(C_1, w) = \begin{cases} \lambda & \text{if } w \in \nu(sw_1); \\ (sw_1, 1) & \text{if } w \notin \nu(sw_2); \\ (sw_1, 1); (sw_2, 0) & \text{otherwise.} \end{cases}$$

The latter model entails that sw_2 mysteriously opens and, hence, that $\mu' \not\models^{\mathcal{I}} \neg sw_1 \wedge sw_2 \rightarrow [C_1]l$; making it the case that $\Gamma \not\models^{\mathcal{I}} \neg sw_1 \wedge sw_2 \rightarrow [C_1]l$. The problem with the latter model—the reason it defies our intuitions—is that, given background knowledge of the domain, while it is natural to associate a dependency between sw_1 and l , the switches sw_1 and sw_2 are independent of each other in this example; making it counterintuitive for sw_2 to change as a result of affecting sw_1 . In the next section this argument is formalised to generalise the imperative semantics so far developed into a causal semantics.

Causal semantics

The intuition this section intends to capture is that, along with any logical dependencies imposed by the constraints on a domain axiomatisation Γ , there may be additional, non-logical dependencies between fluents, as might be captured with a binary relation $R \subseteq F \times F$. So, for example, in the switches example outlined earlier it is natural to attribute a dependency of fluent l on both sw_1 and sw_2 , but no other; in which case, $R = \{(sw_1, l), (sw_2, l)\}$.

In this respect, the anomalous model presented earlier: $\mu'(C_1, (\overline{sw}_1, sw_2, \overline{l})) = (sw_1, 1); (sw_2, 0)$, is less feasible as the succession of affected fluents in this interpretation: sw_1, sw_2 ; do not exhibit a dependency, in contrast with the pair sw_1, l , endorsed by the intended model, which does exhibit a dependency under R . The following formalises this intuition.

A *causal theory* is a pair (Γ, R) , where $\Gamma \subseteq \mathcal{L}$ is a logical theory and $R \subseteq F \times F$ is a relation of *dependence* among fluents. Let $p \in \mathcal{P}$ be a program where $p = (f_{g(1)}, b_1); \dots; (f_{g(n)}, b_n)$.⁸ The *causal sequence* of fluents corresponding to p consists of $f_{g(1)}, \dots, f_{g(n)}$. With a program $p \in \mathcal{P}$ is associated a function $\delta : \mathcal{P} \rightarrow \mathbb{N}$ which assigns to p the number, $\delta(p)$, of *independencies* of p . The function δ is defined (for $p = (f_{g(1)}, b_1); \dots; (f_{g(n)}, b_n) \neq \lambda$) by:

$$\begin{aligned} \delta(\lambda) &= 0; \\ \delta(p) &= 1 + |\{(f_{g(i)}, f_{g(i+1)}) \notin R \mid 1 \leq i < n\}|. \end{aligned}$$

In words, an ‘independence’ instance exists for every pair $(f_{g(i)}, f_{g(i+1)})$, for $1 \leq i < n$, in the causal sequence corresponding to p , which does not occur in the dependency relation R ; the extra one accounts for the effect of the action on the first fluent, which is always assumed independent—an action is always assumed to be initiated in a *stable* world; that is, a world in which no active causal forces are immediately in operation.

A *causal model* of a causal theory: (Γ, R) ; is an imperative model of Γ which minimises $\delta(\mu(\alpha, w))$, for each

$\alpha \in \mathcal{A}$, and $w \in \mathcal{W}$, over all standard models of (Γ, R) . Intuitively, a causal model is one which assumes as few independencies as possible. For $R = \emptyset$; i.e., where no dependencies between fluents exist, it can be shown that causal models reduce to imperative models: observe that $\delta(\lambda) = 0 = |\lambda|$; and $\delta(p) = n = |p|$; the proof follows by induction. This result is pleasing in that it naturally incorporates imperative models, and hence minimal-change reasoning, with scenarios in which no extra-logical dependencies are considered (i.e., the case where $R = \emptyset$).

Returning to the switches example presented earlier, it can now be seen that the causal model criteria means that, with $R = \{(sw_1, l), (sw_2, l)\}$, the causal theory (Γ, R) , with Γ as before yields, for $w = \overline{sw}_1, sw_2, \overline{l}$, that $\delta(\mu(Sh, w)) < \delta(\mu'(Sh, w))$. So, the intended model μ is preferred over the anomalous model μ' , as it warrants fewer independencies.

The approach taken here, incorporating a relation of dependence, is reminiscent of that of (Thielscher 1997) who adopts an ‘influence relation’ which captures a similar intuition. The necessity for an appeal to some such notion is suggestive; although it may be possible to incorporate the dependencies in R by suitably extending the language and axiomatisation, doing so likely involves an overhead such as to make it prohibitive. The approach presented obviates such concerns by making the dependency consideration a separate component of a causal theory.

Conclusion

Summarising, this paper presents a system for reasoning about actions reflecting the intuition that a natural semantics for incomplete action descriptions is that of imperatives; as with statements of typical imperative programming languages. One of the central themes that this paper addresses is the view that the most plausible commonsense inferences an agent makes are often the *simplest* feasible conclusions; in the context of actions, under an imperative semantics, this means the simplest programs/processes/transformations. This perspective is also entertained by (Jauregui, Pagnucco, & Foo 2004). The present treatment is less ambitious but arguably better tailored to the concerns of commonsense reasoning about actions. It shares commonalities with the work of (Zhang & Foo 2001) on extending PDL for reasoning about actions; however, their approach to the frame problem is syntactic (a form of lazy evaluation of frame axioms) leaving their semantics for the frame problem unclear. The present paper borrows many intuitions and has benefitted from the work of (Łukaszewicz & Madalińska-Bugaj 1995) and (Meyer & Doherty 1999).

In similitude with Thielscher’s work on inferring causal ramifications (Thielscher 1997), Halpern and Pearl’s work on actual causation (see (Halpern & Pearl 2001)) incorporates the notion of dependence between fluents through their *causal/structural models*. The latter’s concerns are largely distinct with the ones of this paper, but it is encouraging that the notion of dependence resonates in both; lending support to the incorporation of a relation of dependence in the present work. (Castilho, Gasquet, & Herzig 1999) also uses

⁸Here g yields a permutation of $1, \dots, n$.

a dependence relation to address the frame problem, however, their relation is between action and fluents; in contrast, the present paper takes dependencies between fluents as the fundamental relation.

A shortage of space prohibits an adequate discussion of the limitations of the system presented here—a number of which the author has already identified. One of the more notable omissions regards the simplicity of the languages used, both $\mathcal{A}$ and $\mathcal{L}$, as well as $\mathcal{P}$. Indeed, it is largely due to the restrictions on $\mathcal{A}$ (and hence, $\mathcal{P}$) that the framework is so sympathetic to the principle of minimal change. If space limitations have prevented it from being conveyed adequately in the text, it is the principle of *simplicity* which this paper regards as the more fundamental of the two with respect to commonsense reasoning. Here the complexity of (the interpretation of) an action is the key notion. In this respect, this paper constitutes a preliminary step to a more comprehensive account of the role of simplicity/complexity in commonsense reasoning.

Ultimately, a more systematic analysis approaching the rigour of (Sandewall 1994) needs to be conducted to ascertain where this framework sits in comparison to the many others that exist for commonsense reasoning about actions; an investigation which is already underway.

Acknowledgements

The author would like to thank the KR 2006 reviewers for their comments and suggestions. Special mention is reserved for Tri Minh Cao and Patrick Caldon for their helpful feedback and suggestions on earlier drafts of this paper.

References

- Castilho, M.; Gasquet, O.; and Herzig, A. 1999. Formalizing action and change in modal logic I: the frame problem. *Journal of Logic and Computation* 9:701–735.
- Halpern, J. Y., and Pearl, J. 2001. Causes and explanations: A Structural-Model approach: Part 1: Causes. In Breese, J., and Koller, D., eds., *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI-01)*, 194–202. Morgan Kaufmann.
- Hanks, S., and McDermott, D. 1987. Nonmonotonic logic and temporal planning. *Artificial intelligence* 33:379–412.
- Jauregui, V.; Pagnucco, M.; and Foo, N. 2004. On the intended interpretations of actions. In Zhang, C.; Guesgen, H. W.; and Yeap, W.-K., eds., *Proceedings of the 8th pacific rim international conference on artificial intelligence*, volume 3157 of *Lecture Notes in Computer Science*, 13–22. Springer.
- Li, M., and Vitanyi, P. M. B. 1997. *An Introduction to Kolmogorov Complexity and Its Applications*. Berlin: Springer-Verlag, 2nd edition.
- Lifschitz, V. 1990. Frames in the space of situations (research note). *Artificial intelligence* 46(3):365–376.
- Lin, F. 1995. Embracing causality in specifying the indirect effects of actions. In Mellish (1995), 1985–1991.
- Łukaszewicz, T., and Madalińska-Bugaj, E. 1995. Reasoning about action and change using dijkstra’s semantics for programming languages: preliminary report. In Mellish (1995), 1950–1955.
- McCain, N., and Turner, H. 1995. A causal theory of ramifications and qualifications. In Mellish (1995), 1978–1984.
- McCarthy, J., and Hayes, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. *Machine intelligence* 4:463–502.
- McCarthy, J. 1980. Circumscription—a form of non-monotonic reasoning. *Artificial intelligence* 13:27–39.
- Mellish, C., ed. 1995. *Proceedings of the 14th international conference on artificial intelligence*. Morgan Kaufmann.
- Meyer, J.-J. C., and Doherty, P. 1999. Preferential action semantics (preliminary report). In *Formal Models of Agents*, 187–201. Berlin: Springer.
- Pratt, V. R. 1976. Semantical considerations on Floyd-Hoare logic. In *IEEE Symposium on Foundations of Computer Science*, 109–121.
- Sandewall, E. 1994. *Features and Fluents*. Oxford university press.
- Shanahan, M. 1997. *Solving the frame problem: a mathematical investigation of the common sense law of inertia*. Cambridge, MA, USA: MIT Press.
- Thielscher, M. 1997. Ramification and causality. *Artificial intelligence* 89(1–2):317–364.
- Zhang, D., and Foo, N. Y. 2001. EPDL: A logic for causal reasoning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 131–138.