# On the Intended Interpretations of Actions

Victor Jauregui, Maurice Pagnucco, and Norman Foo

School of Computer Science & Engineering
The University of New South Wales
Sydney, NSW, 2052, Australia.
{vicj,morri,norman}@cse.unsw.edu.au

**Abstract.** In this paper we address the problem of commonsense reasoning about action by appealing to Occam's razor—we should accept the simplest hypothesis explaining the observed phenomena—to generalise the commonsense law of inertia. In particular, we identify the intended interpretation of an action as the *simplest* transformation induced by an action on a world to produce a possible successor. We formalise the notion of *simplicity* of a transformation as its conditional Kolmogorov complexity. Finally we show that the framework can solve simple commonsense reasoning problems and indicate its role as a first step towards capturing commonsense notions of causation.

## 1 Introduction

The problem of *commonsense reasoning about action* can be characterised as follows: given a (possibly incomplete) description of a world $w$, and a generally incomplete specification of an action $a$, what are the possible *successor worlds*, denoted $Res(a, w)$, which ensue after the action is performed?

Quite generally, an action specifies a mapping, or transformation, between worlds. In the presence of incomplete descriptions, or non-deterministic actions, this mapping may not be unique, but may correspond to a number of possible transformations. When an initial world $w$ is supplied, identifying the successor worlds resulting from an action is tantamount to identifying the intended transformations for the given action description.

The difficulty faced is that, typically, we only characterise an action (incompletely) by its *direct effects*—those effects for which the action is immediately responsible. Consider the following example.

> We have $n > 0$ homogeneous switches, labelled 1 through $n$. Suppose, initially, all our switches are open (off). Assume, further, that we specify an action which makes the $k$-th switch (for some $k$) closed (on).

In particular, note that our action specification has not described the intended transformation completely. We have not, for example, specified what should happen to the $j$-th switch, for any $j \neq k$.

The *commonsense* intuition, however, is clear; given our incomplete action specification, the most plausible interpretation for the action yields that only

the $k$-th switch closes, while the others remain unaffected. The argument is that because our action specification provides no support for any other switch closing, other than the $k$-th, had we intended any other effects, these would have been specified in the action description.

We can characterise these notions more formally as follows. Since our action describes only its direct effects, we admit as our potential *candidate* successors all worlds which satisfy the action's direct effects. We denote this set of candidates, $E(a)$. As action $a$ provides an incomplete description, $E(a)$ is likely to be too permissive—our intended worlds will be a selection among these worlds. This amounts to identifying a function $S$ which considers an initial world $w$ and a set of candidate successors and picks out the intended successors. That is, we can express this as $Res(a, w) = S(w, E(a))$, depicting it graphically in Figure 1.

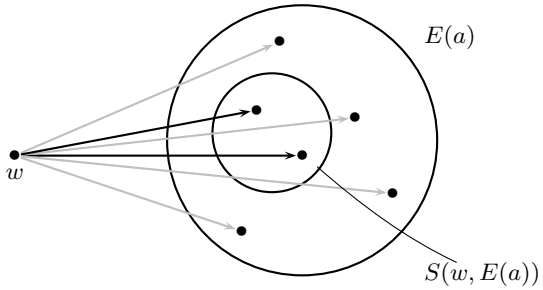


**Fig. 1.** Candidate mappings associated with an action. Intended mappings are represented by solid lines.

This kind of characterisation has been adopted in the approaches of McCain and Turner [1] and Pagnucco and Peppas [2]. We follow a similar direction but place our emphasis on finding the intended interpretation of an action. In particular, our selection function $S$ selects the intended mappings from $w$ into $E(a)$.

In Figure 1 the candidate mappings into $E(a)$ are depicted in grey with the solid arrows indicating the intended mappings among these. Once we have identified the intended mappings we can readily identify the intended successors.

This paper aims to rationalise our appeals to commonsense in reasoning about action by looking to identify the most plausible transformations which corresponds to an incomplete action specification. Traditionally this has been achieved by appealing to some notion of *minimal change*—any change that is not dictated by the action specification should not take place. We attempt to generalise this commonsense notion by appealing to the more general principle of *Occam's razor*; identifying the simplest transformations consistent with the action description as the intended mappings.

In Section 2 we motivate our work by showing how our intuitions manifest themselves in the situation calculus—the most common framework used to describe reasoning about action problems. In Section 3 we characterise the simplicity of a mapping between worlds by its conditional Kolmogorov complexity, allowing us, in Section 4, to formulate the problem of commonsense reasoning about action in terms of the complexity of a transformation. More specifically,

the intended interpretation of an action is identified with the simplest transformations. Section 5 shows that we can capture some of our commonsense intuitions in this framework, providing an elegant solution to Hanks & McDermott's Yale Shooting Problem [3]. We also highlight some limitations in the current formalism indicating there is still substantial work to be done to get the formalisation 'right'. Finally, we conclude in Section 6, with a summary and some motivation for the present framework as an initial step towards the formal characterisation of commonsense notions of causation.

## 2 Background

The *situation calculus* (see, [4]) is a logical formalism for reasoning about action. It consists of *situations*, which indicate the state of the system we are modelling at a given instant in time; *fluents*, which identify the properties of that system we are interested in modelling; and *actions* which dictate the transitions between states that can take place. To say that the property described by a fluent $f$ holds in situation $s$ we write $Holds(f, s)$. So, for example, if we use the fluent $sw(k)$ to denote that switch $k$ is closed in situation $s$, we would write $Holds(sw(k), s)$.

The system described in our example above would consist of the fluents: $sw(1), sw(2), \ldots, sw(k), \ldots, sw(n)$ describing whether switch $k$, for $k = 1, \ldots n$, is open or not. The initial situation, in which all the switches are open, would correspond to the situation term $S_0$, and would be described by the sentences:

$$\neg Holds(sw(1), S_0), \neg Holds(sw(2), S_0), \ldots, \neg Holds(sw(n), S_0) \qquad (S_0)$$

Actions invoke state transitions which are captured by adding a function term $Result$, which maps an action and a situation to a successor situation.

Describing the direct effects of the action $close(k)$, which, as mentioned earlier, specifies that the $k$-th switch becomes closed, would be done by the *effect axiom*: $Holds(sw(k), Result(close(k), s))$.

Once supplied an action $a$ and a world $w$, it remains to characterise the intended successors $Res(a, w)$, via the intended interpretation of the action. In our example, the intuition that the desired mapping is the one that leaves all switches, other than the $k$-th, unaffected is captured by the *Commonsense Law of Inertia* which according to Shanahan [5] states, among other things, that:

> Normally, given any action (or event type) and any fluent, the action doesn't affect the fluent. [p18]

Moreover, Shanahan argues:

> As a scientific claim, the commonsense law of inertia wouldn't stand up to much scrutiny. But it's much better thought of, not as a statement about the world, but either as a useful representational device or as a strategy for dealing with incomplete information. [p18]

From our perspective the commonsense law of inertia gives a particular rule for capturing our selection criteria $S$.

In a logical framework, we eliminate unwanted models by adding extra axioms. We characterise the role of $S$ similarly here. In general, for selection criteria $S$ we can identify a set of formulae $\Gamma_S$ which admits only the intended models. We can think of $\Gamma_S$ as supplying a description of the intended transformations.

In particular, when $S$ corresponds to the commonsense law of inertia, we can capture the intended mapping by adding *frame axioms*, which describe inertial transformations. Consider the switches example we described earlier with $n$ fluents and possibly $m$ actions ($close(k)$ for $1 \leq k \leq m \leq n$). The following, approximately $\approx 2 \times n \times m$ frame axioms, capture the inertial mapping—with one such set of $\approx 2 \times n$ axioms for each action $close(k)$:

$$\begin{array}{c} Holds(sw(1), s) \rightarrow Holds(sw(1), Result(close(k), s)) \\ \neg Holds(sw(1), s) \rightarrow \neg Holds(sw(1), Result(close(k), s)) \\ \vdots \\ Holds(sw(n), s) \rightarrow Holds(sw(n), Result(close(k), s)) \\ \neg Holds(sw(n), s) \rightarrow \neg Holds(sw(n), Result(close(k), s)) \end{array} \qquad (\Gamma_S)$$

We can observe a significant degree of regularity in these axioms, however. Taking advantage of this we can simplify our description, by treating each fluent uniformly, to $\approx 2 \times m$ axioms of the form (one of these for each action): [1]

$$\begin{array}{c} \forall f \neq sw(k)\ Holds(f, s) \rightarrow Holds(f, Result(close(k), s)) \\ \forall f \neq sw(k)\ \neg Holds(f, s) \rightarrow \neg Holds(f, Result(close(k), s)) \end{array} \qquad (\Gamma'_S)$$

The important point we wish to make here is that the commonsense law of inertia constitutes a simple rule for completing the description of a transformation given an incomplete action description. The development above showed that this notion of simplicity can be captured by observing regularities to produce a short description of the intended transformation. On this basis, we make more precise Shanahan's claim that the commonsense law of inertia is a useful representational device for dealing with incomplete information. It is 'useful' in the following senses:

- it has a physical justification in Newton's mechanical law of inertia;
- it furnishes a simple rule to describe the intended transformations when provided an incomplete action description.

What remains is to give a more precise notion of simplicity with which to identify 'simple' transformations.

[1] Moreover, if we take advantage of the regularity in the actions (which is a feature of this example but would not generalise readily), we can simplify things even further to just two axioms:

$$\begin{array}{c} \forall k\ \forall j \neq k\ Holds(sw(j), s) \rightarrow Holds(sw(j), Result(close(k), s)). \\ \forall k\ \forall j \neq k\ \neg Holds(sw(j), s) \rightarrow \neg Holds(sw(j), Result(close(k), s)). \end{array}$$

## 3 Measuring Simplicity

Essentially, the main idea of this paper is *to equate commonsense, in reasoning about action, with the simplicity of a transformation.* In the sense that we showed that the commonsense law of inertia appeals to a simple rule, we argue that, in the absence of further information, commonsense inferences are the simplest inferences we can make from the given evidence.

Unfortunately, the simplest inferences we can make are often outside the scope of the language we use for our representations. This is the case, for example, when we use circumscription (see [6]) to characterise the commonsense law of inertia. So the simple analysis provided above, of adding a brief collection of formulae $\Gamma_S$ to capture the intended transformations, may not be feasible.

Instead, our approach is to look at a well established measure of simplicity—or, its dual, complexity—called *Kolmogorov complexity* (see Li & Vitanyi [7]), which gives the complexity of a string $x$, and proceed to encode our logical theories into strings to make use of this measure. Let us first define Kolmogorov complexity (the following is an adaptation taken from Li & Vitanyi [7]):

**Definition 1.** *The* ***Kolmogorov complexity*** *of a string $x$ (over some alphabet), denoted $K(x)$, is the length of the shortest program $p$ (usually encoded over the binary alphabet) which when supplied to a universal Turing machine, $U$, produces $x$. That is:*

$$K(x) = \min\{|p| : U(p) = x\}. \tag{1}$$

For our purposes we want a measure of the simplicity of a transformation. There is a variant of Kolmogorov complexity we can use for this (see Li & Vitanyi [7]):

**Definition 2.** *The* ***conditional Kolmogorov complexity*** *of a string $x$ given a string $y$, denoted $K(x|y)$, is the length of the shortest program $p$ which, when given $y$ as input, computes $x$. That is:*

$$K(x|y) = \min\{|p| : U(p, y) = x\}. \tag{2}$$

The intuition we want to capture is that, if we encode worlds $w$ and $v$ as strings, then the simplicity of the mapping from $w$ to $v$, imposed by performing action $a$ in world $w$, is determined by the conditional Kolmogorov complexity $K(v|w)$.

## 4 Formalising Commonsense Reasoning About Action

Our formalism involves mapping situations to strings, on which the intended mappings correspond to the simplest transformations.

Consider the switches example used earlier. We can map the initial situation $S_0$, in which all the $n$ switches are initially open (off), to the binary string: $w = 0^n$. Our mapping is straight forward: the truth value of a fluent (i.e., whether the fluent holds or not) is determined by the value of the corresponding bit in the string. In our example, the $k$-th bit determines whether the $k$-th switch is open or closed. As such, a world under this encoding is just an $n$-bit string.

Corresponding to the action description $Holds(sw(k), Result(close(k), s))$, which admits as possible candidate successors those worlds $v$ consistent with the $k$-th switch being closed, we define our effect function $E$, such that $E(close(k))$ consists of the set of $n$-bit strings with the $k$-th bit set to one, i.e., $E(close(k)) = \{x1y : |x1y| = n, |x| = k - 1\}$. Once we have $w$ and have determined $E(a)$ for action $a$, we need to select the intended mappings from among those candidate mappings of $w$ into $E(a)$. As we outlined above, the intended mappings are the ones with the simplest transformations, which correspond to the shortest programs that transform $w$ into some $v \in E(a)$. Formally:

**Definition 3.** *Given a world $w$ and an action $a$, the set of possible successor worlds is the subset of candidate worlds $v \in E(a)$ with minimal conditional Kolmogorov complexity given $w$. That is:*

$$Res(a, w) = S(w, E(a)) = \min_{v} \{K(v|w) : v \in E(a)\}. \tag{3}$$

The intuition being that our selection function identifies the simplest transformations; taking the worlds these map to as the desired successors.

The intention is that, because an action will generally be incompletely specified through its direct effects, the various underlying mechanisms that bring about these effects are non-monotonically implied. Moreover, in the absence of further information, we cannot rationally justify any more mechanisms than are necessary to bring about said effects. Li & Vitanyi [7] express this as follows:

> We are to admit no more causes of natural things (as we are told by Newton) than such as are both true and sufficient to explain their appearances.

Significantly, we note that inertial transformations will generally feature prominently among the simplest transformations. This is perhaps indicative of why the commonsense law of inertia is a good heuristic for commonsense reasoning about action. In particular, if we choose our universal Turing machine $U$ in the definition of $K(x|y)$ (2), so that the empty program does altogether nothing, then the inertial transformation, which simply corresponds to the identity map, yields the simplest possible transformation.

**Proposition 1.** *Let $U$ in (2) be such that $U(\varepsilon, y) = y$, then $K(x|y)$ receives its least value 0, when $x = y$.*

*Proof.* Since the empty string $\varepsilon$ is the unique string with $|\varepsilon| = 0$, and all programs $p$ are encoded by strings, then $\varepsilon$ supplies the unique program of length 0 which yields $K(x|x) = 0$. □

## 5 Results and Discussion

In general, we expect that in a world in which few things change when an action is performed, any changes not specified by the direct effects of an action would require some form of elaboration corresponding to the conjecture of a cause,

or explanation, for these changes. In this sense, deviations from inertia would require longer descriptions, excluding them from among the simplest mappings.

Earlier we conjectured that the inertial map would feature prominently among the simplest transformations when few things tend to change. It may be, though, that an action changes many things in a regular (though not inertial) manner, or that the world itself evolves regularly. In such cases the assumptions that justify the commonsense law of inertia break down.

In these instances the most rational choice of succession is that which most closely exhibits the pattern of change indicated by the action description. Our concerns here largely coincide with Solomonoff's in his theory of inductive inference [8]. Indeed, prediction problems, such as the one we are interested in, typically appeal to some form of inductive inference based on past experience. Our framework accommodates such inductive inferences naturally.

We now show that this framework can capture our intuitions with the simple switches example. The simplest transformation (i.e., the intended mapping) which maps a string of $n$ 0's to a string with the $k$-th bit set to one (that is, into the set $E(close(k))$), intuitively, is the following:

$$\underbrace{0\ldots000\ldots0}_{n} \xrightarrow{close(k)} \underbrace{0\ldots0\overbrace{1}^{k-1}0\ldots0}_{n}$$

This transformation corresponds to the program which moves to the $k$-th position on the Turing machine tape and writes a '1'. We are faced with the problem that encoding the value of $k$ in our program would incur an overhead which would suggest a preference for changing the earliest bits in the string; as these incur smaller overheads to specify. As no such preference should be implied, we overcome this by factoring out the particular value of $k$ from our program. The way we do this is to define our universal machine $U$ in (2), to have three tapes. The first is the *world-tape*, on which is encoded the initial world $w$. The world tape will also contain the output after it has undergone the specified transformation. The second tape, called the *program-tape*, contains the program we identify with the transformation to take place on the world tape. The third tape is a *data-tape* containing particular information about the action.

In our example, for the action $close(k)$, on the data tape would appear $k-1$ '1's, constituting the $k-1$ shifts needed to identify the $k$-th bit. The program tape would then refer to the data tape for this particular information rather than having it coded within the program. By discounting the data tape from consideration of the complexity (size) of the program we remove the bias introduced by the particular, arbitrary ordering of bits (fluents) imposed by the tape.

The key motivation behind this is to keep the program/data distinction. Li & Vitanyi [7] make a similar distinction when they consider *two-part codes*. The program encodes what is referred to as the *model*, which in our case we wish to identify with the *nature* of the transformation (in this case the setting of a bit). The data tape simply encodes particular information regarding the action (in this case the particular value which identifies the bit to alter). With these considerations, below (on the right) we encode the shortest program which

transforms $n$ zeros to a string with a '1' at the $k$-th bit, coinciding with the intended/simplest mapping (on the left):

$$\underbrace{0\ldots000\ldots0}_{n} \xrightarrow{close(k)} \underbrace{0\ldots0\overbrace{1}^{k-1}0\ldots0}_{n} \quad : \quad \begin{array}{l}(\mathtt{q_0},\ {}^{\times}_{1},\ {}^{\mathtt{R}}_{\mathtt{R}},\ \mathtt{q_0})\\ (\mathtt{q_0},\ {}^{\times}_{\_},\ {}^{1}_{\times},\ \mathtt{q_H})\end{array}$$

The only alternative simplest program which would map $w$ into $E(close(k))$, is the program which writes $k$ '1's on the tape. This would correspond to:

$$\underbrace{\overbrace{0\cdots0}^{k}0\ldots0}_{n} \xrightarrow{close(k)} \underbrace{\overbrace{1\cdots1}^{k}0\ldots0}_{n} \quad : \quad \begin{array}{l}(\mathtt{q_0},\ {}^{\times}_{1},\ {}^{1}_{0},\ \mathtt{q_0})\\ (\mathtt{q_0},\ {}^{\times}_{0},\ {}^{\mathtt{R}}_{\mathtt{R}},\ \mathtt{q_0})\\ (\mathtt{q_0},\ {}^{\times}_{\_},\ {}^{1}_{\times},\ \mathtt{q_H})\end{array}$$

We see that this unintended map is more complex (has a longer program). In fact, this is only the case because we have omitted in the intended program the Turing machine tuples when we don't care about reading a '0' on the data tape. This is no severe restriction as we can always choose a reference machine $U$, in (2), which adheres to the common convention that if a state-input pair is not found then the machine simply halts.

This example shows that the formalism captures the same intuitions that the commonsense law of inertia does when an action changes few things. The next example shows that it allows us to capture our intuitions when appeals to the principle of minimal change fail.

Consider the Yale Shooting Problem as proposed initially by Hanks & McDermott [3] and cited in Shanahan [5]. The scenario consists of a turkey and a gun; which is used to shoot the turkey. We identify two fluents: *Alive* and *Loaded* to indicate that the turkey is alive and the gun is loaded, respectively. There are also three actions *Load*, $Wait$ and *Shoot*, with the obvious meanings. Suppose our initial situation $S_0$ has the turkey alive and the gun unloaded. These actions are specified according to the following effect axioms:

$$Holds(Loaded, Result(Load, s))$$
$$Holds(Loaded, s) \rightarrow \neg Holds(Alive, Result(Shoot, s))$$

Note that, as the wait action is intended to do nothing, its effect axiom is omitted.

Consider performing the sequence of actions, *Load* then $Wait$ followed by *Shoot*. Intuitively we expect the following model, which we have depicted pictorially:

$$\overset{A,\overline{L}}{\bullet} \xrightarrow[\Delta L]{Lo} \overset{A,L}{\bullet} \xrightarrow{Wa} \overset{A,L}{\bullet} \xrightarrow[\Delta\overline{A}]{Sh} \overset{\overline{A},\overline{L}}{\bullet}$$

where the $\Delta$'s below the arrows indicate the occurrence of an abnormality with the respective fluent. Unfortunately, the following anomalous model is also admitted when we minimise change (to see this observe that there are as many $\Delta$'s in the anomalous model as in the intended one, however, they occur at different times with different fluents):

$$\overset{A,\overline{L}}{\bullet} \xrightarrow[\Delta L]{Lo} \overset{A,L}{\bullet} \xrightarrow[\Delta\overline{L}]{Wa} \overset{A,\overline{L}}{\bullet} \xrightarrow{Sh} \overset{A,\overline{L}}{\bullet}$$

This second, anomalous model is clearly counter-intuitive. There is no justification for the gun becoming unloaded during the wait action. In our framework we can show that the anomalous model is rejected. This result confirms that, in our approach, the inertial mapping will generally feature among the simplest transformations. In particular, the $Wait$ action, having been specified as not doing anything (thus corresponding to $E(Wait)$ admitting all possible worlds), receives as its intended interpretation the simplest program which does nothing—the empty program. More generally we have:

**Proposition 2.** *Let $w$ be a world and 'a' an action such that $w \in E(a)$, then the intended mapping is always the inertial mapping. That is, $Res(a, w) = \{w\}$.*

*Proof.* From Proposition 1, $K(v|w)$ gets its least value for $v = w$. Since $w \in E(a)$, $\{K(v|w) : v \in E(a)\}$ is minimised when $v = w$, yielding $Res(a, w) = \{w\}$. □

Carrying over the arguments from the switches example, the load action receives the program that sets the only bit associated with the *Loaded* fluent and the *Shoot* action gets the program which checks if the *Alive* bit is set and resets it.

The program that performs the composite sequence of actions, consisting of *Load*, $Wait$ and *Shoot* actions, we take to be the composition of these programs. This composite program, associated with the composite action, clearly yields only the intended model above and not the anomalous model. In particular, we cannot trade a change during the $Wait$ action with a change during the *Shoot* action, as takes place in the anomalous model under minimisation of change.

Unfortunately, though it appears the formalism presented has a number of desirable properties, and generally adheres to our commonsense intuitions, it also suffers a number of obvious deficiencies which suggest the framework, in its present incarnation, is not satisfactory. One such deficiency regards the program/data distinction. Our solution of having separate program and data tapes appears too simplistic. In particular, we have the following.

**Proposition 3.** *The complexity of a world transformation is bounded by a fixed constant that depends on the universal machine $U$ used in (2).*

*Proof.* Let $u$ be a program which runs on $U$ that ignores the world tape and interprets the data tape as a program ($u$ encodes a universal program). We can proceed as follows: we encode $u$ on our program tape, supplying, on the data tape, a program $p$ such that $u(p) = v$, for any $v \in E(a)$. Now $U(u, w) = v$, so, by (2), $K(v|w) \leq |u|$. □

This is clearly a severe limitation. What it shows, in particular, is that we have been too simplistic in determining the roles that program and data can play in our formalism. More specifically, what might be required is to place a stronger restriction on what constitutes valid data.

## 6 Conclusion

The aim of this paper was to provide a formalism for commonsense reasoning about action which appeals to Occam's razor as its guiding principle, generalising the commonsense law of inertia.

We argued that we can identify commonsense with simplicity which we went on to formalise using Kolmogorov complexity. Subsequently, a formalism that identifies the intended interpretations of an action as the simplest transformations that satisfy the direct effects of an action is provided. We showed that it is possible to characterise commonsense intuitions regarding minimal change in this framework, and showed that we can solve the Yale Shooting Problem when minimal change breaks down.

Ultimately, we argued, the present framework still faces significant limitations which render it preliminary. A number of such issues (for example, the problems with the program/data distinction) are currently under investigation.

One of the main motivations behind our work has been to furnish a framework with which we can analyse such aspects of commonsense reasoning as causal reasoning, as identified by McCain & Turner [1], Lin [9], Thielscher [10], Sandewall [11], among others. In this respect, this paper is an attempt to lay the groundwork for such an analysis. In particular, just as the 2nd Law of Thermodynamics identifies a direction of time and hence causation, so we hope that analogous information theoretic arguments may allow us to give a formal characterisation of commonsense notions of causation. The hope is that the framework proposed will supply a natural platform through which to address these concerns.

## References

1. McCain, N., Turner, H.: A causal theory of ramifications and qualifications. In Mellish, C., ed.: Proceedings of the 14th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, San Francisco (1995) pp1978–1984
2. Pagnucco, M., Peppas, P.: Causality and minimal change demystified. In Nebel, B., ed.: Proceedings of the 17th International Joint Conference on Artificial Intelligence. Volume 1., Seattle, Washington, Morgan Kaufmann (2001) pp125–130
3. Hanks, S., McDermott, D.: Nonmonotonic logic and temporal projection. Artificial Intelligence **33** (1987) pp379–412
4. McCarthy, J., Hayes, P.: Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B., Michie, D., eds.: Machine Intelligence 4. Edinburgh University Press (1969) pp463–502
5. Shanahan, M.: Solving the frame problem. MIT Press, Cambridge, Mass. (1997)
6. McCarthy, J.: Circumscription — a form of nonmonotonic reasoning. Artificial Intelligence **13** (1980) pp27–39
7. Li, M., Vitnayi, P.: An introduction to Kolmogorov complexity and its applications. 2nd edn. Springer-Verlag, New York (1997)
8. Solomonoff, R.: A formal theory of inductive inference. Part I. Information and Control **7** (1964) pp1–22
9. Lin, F.: Embracing causality in specifying the indirect effects of actions. In Mellish, C., ed.: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, San Francisco (1995) pp1985–1991
10. Thielscher, M.: Ramification and causality. Artificial Intelligence **89** (1997) pp317–364
11. Sandewall, E.: Transition cascade semantics and first assessments results for ramification, preliminary report. Technical Report R-96-19, Department of CIS, Linköping University, Sweden (1996)