Action Localness, Genericity and Invariants in STRIPS

Norman Y. Foo and **Abhaya Nayak** and **Maurice Pagnucco**
Knowledge Systems Group
Department of Artificial Intelligence
School of Computer Science and Engineering
The University of New South Wales
NSW 2052 Australia

Pavlos Peppas
Knowledge Systems Group
School of MPCE
Macquarie University
NSW 2109, Australia

Yan Zhang
Knowledge Systems Group
Department of Computing
University of Western Sydney, Nepean
Kingswood, NSW 2747, Australia

Abstract

The STRIPS specification for actions, despite criticisms of its shortcomings, remains popular in implementations because of its simplicity and intuitive appeal. Lifschitz showed in his logical account of STRIPS that states can safely be represented as sets of ground atoms, and this is our starting point. Our aim is to provide logical foundations for design heuristics and procedural implementations of STRIPS. In particular, we formalize the notions of 'localness' in pre- and post-conditions of actions. This justifies and explicates the intuitions that underlie the diagrammatic reasoning typically used to design such conditions. A formal account is also given for 'generic' actions that have parameters. We conclude with a method for extracting candidate action invariants that distinguishes between two kinds of inertia implied by the action specifications.

1 Introduction

Among the earliest proposals for specifying actions is STRIPS, due to Fikes and Nilsson [Fikes and Nilsson 71]. STRIPS has the great merit of simplicity and ease of implementation. As an example and a quick review of it, here is how the action of picking up block A in a blocks world may be specified:

action: pick-up(A)
pre-condition: on(A,X), clear(A)
post-condition:
  add-list: inhand(A)
  delete-list: on(A,X)

In a Prolog setting, this is easily implemented by the rule: *pick-up(A) :- on(A,X), clear(A), retract(on(A,X)), assert(inhand(A))* assuming that the state of the blocks-world is represented as a collection of Prolog facts. An immediate observation is that both the specification and the implementation describe a *generic* action that has A as a parameter, and that X is implicitly existentially quantified. Even this example however is not as general as STRIPS can be, for it is possible to have first-order formulas in the pre-condition and post-condition. However, unless restrictions are observed, the inclusion of such formulas, as pointed out by Lifschitz [Lifschitz 87] can cause difficulties. Moreover, until that paper appeared, there was no proper logical semantics for STRIPS. Further, constraints or invariants (like "no block can be both inhand and on something") are observed either (a) by specification or programming of the add-list and delete-list to ensure non-violation, or (b) by Prolog rules of the form: *impossible :- inhand(A), on(A,X)* where *not(impossible)* is used as a guard in the body of action rules.

A logical semantics of STRIPS should build upon the insights of Lifschitz and also give an account of several other characteristics. Lifschitz showed that if states of the world are represented as sets of ground atoms, then his semantics of STRIPS is safe, and we will use this as a starting point because we are principally interested in understanding the efficacy of Prolog-like implementations that are based on such state representation. Designers of rules for such programs often draw pictures like that shown in figures 1 and 2 to reason about pre- and post-conditions, and to maintain constraints implicitly. The most striking characteristic of such pictures is the identification of 'localness' and 'genericity'. Localness has to do with the fact that an informal 'window' (see the dashed enclosures in the figures) is typically drawn around the objects relevant to the action instance. Genericity is the notion that each window to which an action can apply looks like any other similar window except for a change in parameters. Another characteristic, noted by Lifschitz [Lifschitz 87] is that the constraints like the *impossible* rule above are often maintained implicitly by clever design of the add- and delete-lists. In this paper we address these and related characteristics to provide a formal explication for why they work.

The organization of the paper is as follows. In section 2 we recapitulate a number of standard definitions, introduce a formal counterpart of the 'window' idea and show how it accounts for pre-condition 'localness'. We also provide the formal equivalent of 'genericity' for actions. In section 3 we examine state change due to actions and show that post-condition 'windows' have the same boundary as the pre-condition ones. We also identify two kinds of inertial assumptions that appear to be implicitly used by designers of action specifications. Section 4 considers intended design semantics in the add- and delete-lists, and shows how candidate action invariants can be derived from them depending on two kinds of inertial assumptions. Concluding the paper is section 5 which discusses what we are currently pursuing and what has yet to be done.

2 States, Actions and Windows

States of the world are describable formally in many ways, but a popular one is to represent them as collections of ground atoms. This is a particularly felicitous choice for databases, and for implementations in languages like Prolog. In this paper we accept this choice as it will permit us to focus on the issues that really matter without being distracted by technicalities that may attend alternative choices. It has the virtue of simplicity and is sanctioned by earlier investigations by Lifschitz. To this end therefore, we will adopt a first-order language with equality as our descriptive medium. Section 2.1 recalls the logic and model theory needed.

Actions are state transformers. In the STRIPS tradition, for an action to apply in a state its pre-condition must be satisfied by that state. Actions that are specified with parameters or variables have instances that are the result of instantiating or substituting constant names of objects in the state into the action pre-condition. If successfully applied, the post-conditions of the action are actuated, i.e., the add-list and delete-list operations are performed on the representation. If states are theories, then actions so modelled are theory change operators. Such operators have been studied extensively in the logics of action from many perspectives (an example is [Lin 95]), but it is not the purpose of this paper to introduce yet another. Rather, we wish to examine a well-known system that has wide practical acceptance to see if its design heuristics, particularly those with a diagrammatic flavor such as exposited by Barwise [Barwise 95], can be given formal justifications.

2.1 Basic Definitions and Constructions

Since ground atoms are the basis of state descriptions, in order to evaluate formulas with respect to states we need to know when an atom does not hold. The simplest assumption to achieve this is to use the version of the *closed world assumption* [Reiter 78] in the absence of constraints.

Definition 1 *If Δ is a set of ground atoms of $\mathcal{L}$, then the closed world assumption applied to Δ is CWA$(\Delta) = \Delta \cup \{\neg L | L \notin \Delta$ and L is a ground atom$\}$.*

This is actually a simplification of the original definition which involves state constraints, but here none are assumed to be (explicitly) available. A potential problem with this assumption is inconsistency when the description is *indefinite* [Zhang and Foo 96], e.g. there is a constraint of the form $p \vee q$ where p and q are atoms, and neither of them appear in the state description. The indefiniteness of actions in terms of their pre- or post-conditions is known to be problematic [Zhang and Foo 96]. For pre-conditions it is possible to avoid it by refining the action ontology to "split" the indefinite action into definite ones, but for post-conditions no such easy way is known. To circumvent this for the purpose of focusing on the main ideas, we shall therefore assume that all descriptions for states as well as for actions are *definite*. Then, using the closed world assumption, all states are *ground complete*, i.e., for each ground atom, either it or its negation holds in a state.

Formally, complete states are *structures* in model theory and logic, so we shall use the terms *state* and *structure* interchangeably. The next three definitions are paraphrased from [Enderton 72; Chang and Keisler 73]. (As it gets tedious to make distinctions between predicate (constant) symbols and the relations (domain elements) they denote, we will not observe these distinctions later.)

Definition 2 *Suppose $\mathcal{L}$ is a first-order language with equality. An $\mathcal{L}$-structure $\mathcal{U}$ is a triple $\langle \mathcal{A}, \mathcal{R}, \mathcal{C} \rangle$ where $\mathcal{A}$ is a non-empty set called the domain of $\mathcal{U}$, $\mathcal{R}$ is a map that assigns relations of the appropriate arity on $\mathcal{A}$ to predicate symbols in $\mathcal{L}$, and $\mathcal{C}$ is a map that assigns elements of $\mathcal{A}$ to constant symbols in $\mathcal{L}$.*

Definition 3 *If $\mathcal{U}_1$ and $\mathcal{U}_2$ are $\mathcal{L}$ structures with domains $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively such that $\mathcal{A}_1 \subseteq \mathcal{A}_2$, and the relation and constant assignment maps of $\mathcal{U}_1$ are that of $\mathcal{U}_2$ restricted to $\mathcal{A}_1$, then $\mathcal{U}_1$ is a substructure of $\mathcal{U}_2$, and conversely $\mathcal{U}_2$ is an extension of $\mathcal{U}_1$.*

Definition 4 *Two $\mathcal{L}$ structures $\mathcal{U}_1$ and $\mathcal{U}_2$ are elementarily equivalent, written $\mathcal{U}_1 \equiv \mathcal{U}_2$, if for every sentence $\phi \in \mathcal{L}$, $\mathcal{U}_1 \models \phi$ iff $\mathcal{U}_2 \models \phi$.*

Two elementarily equivalent structures satisfy exactly the same set of sentences, and are therefore indistinguishable by sentences. To look ahead a little, our interest in substructures arises from the intuition that they are the 'windows' within pictures of states to which actions are locally applied. The construction of these windows is determined by the arguments or parameters of the action pre-condition, and the objects of relevance are accumulated via a closure operation. This closure has the effect of accruing, to the constants that instantiate a pre-condition, other constants that may be needed to evaluate predicates in the pre-condition. For instance, if $\exists X\, P(X, b)$ is a component of a pre-condition and

b is a constant instantiated into it as an action parameter, we would need to accrue all constants c in the (domain of the) structure for which $P(c, b)$ holds in order to evaluate the component. This motivates the closure definition below.

Definition 5 *Given a set S of constants in $\mathcal{U}$, the expansion of S by n-ary predicate P in $\mathcal{U}$, denoted $closure(\mathcal{U}, S, P)$, is $\bigcup\{\{a_1, \ldots, a_n\} \mid$ some a_i is in S, and $\mathcal{U} \models P(a_1, \ldots, a_n)\}$. The notation $closure(\mathcal{U}, S, P_1, \ldots, P_k)$ has the obvious generalization to several predicates.*

In fact, this closure is a special case of a more general closure which is suggested by the notation $closure(\mathcal{U}, S, P, k)$ where k is the level of transitivity of P used in the expansion of S, e.g., if $a \in S$ and P is binary with $P(a, b)$ and $P(b, c)$ holding in $\mathcal{U}$ (while neither b nor c is in S), both b and c will be in $closure(\mathcal{U}, S, P, 2)$. Our closure is for $k = 1$, and we confine attention to this case simply because the key ideas in this paper are thereby not obscured by technical detail, and moreover for most practical applications $k = 1$ suffices. The generalization to arbitrary k is conceptually easy. The construction of the closure implies the following lemma.

Lemma 1 *Assume the notation as in definition 5. If $\mathcal{U} \models \exists Y Q(Y, X_1, \ldots, X_k, a_1, \ldots, a_m)$ then for some b in $closure(\mathcal{U}, \{a_1, \ldots, a_m\}, Q)\mathcal{U} \models Q(b, X_1, \ldots, X_k, a_1, \ldots, a_m)$. (As the converse is trivial, this is really an iff.)*

This lemma is the basis for an induction that establishes the corresponding lemma in which Q above is replaced by general formulas called *D-formulas* in definition 6 below, where D is $\{a_1, \ldots, a_m\}$ in the present context.

2.2 Blocks World Example

The blocks-world structure (state) illustrated by figures 1 and 2 will motivate much of the later development. All of figure 1 shows a world W in which $\mathcal{L}$ has predicate symbol $on(\cdot, \cdot)$, constant symbols $\{a, b, c, d, e, table\}$. Letting the constant symbols name their domain elements and the predicate symbol name its relation, the theory T_2 (subtheory of T_1, to be introduced shortly) of the structure is axiomatizable by the set $\{on(a, b), on(c, d), on(e, table), on(b, table), on(d, table)\}$. The dashed part of figure 1 shows a substructure of T_2 with the theory T_1 axiomatizable by $\{on(a, b), on(b, table), on(e, table)\}$. Already, if one accepts that the dashed part is a sub-structure of the entire structure, from well-known *preservation theorems* [Chang and Keisler 73] we can conclude the following: the universal sentence $\forall x \neg on(x, a)$ holds in T_2, so by the fact above, it also holds in T_1; the (trivially) existential sentence $on(a, b)$ holds in T_1, so it also holds in T_2. However, it will turn out that these conclusions from standard model theory are special cases of the main theorem of this section. This example is much simplified, for in practice we may need more predicates than shown, e.g., *clear, block* etc. By the CWA the negative ground literals are implied and not shown. As a result, the formula $\forall X \neg on(X, a)$ holds in

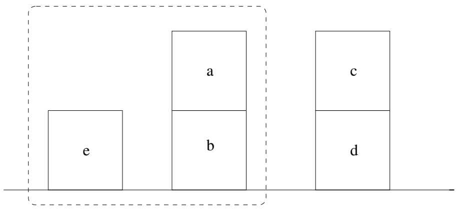

Figure 1: A pre-condition window

this structure. Observe that this is the intended meaning of the predicate $clear(a)$ in practice, so $clear(\cdot)$ is *eliminable* in a language with the $on(\cdot, \cdot)$ predicate.

2.3 What Is A Window?

We are now ready to formalize the informal notion of a 'window' for an action instance. Let the pre-condition of action A be $\phi(X_1, \ldots, X_n)$. If A applies in state $\mathcal{U}$ with substitutions $a_1, \ldots, a_n$ for $X_1, \ldots, X_n$ respectively, construct a sub-structure $\mathcal{W}$ of $\mathcal{U}$ by letting the domain of $\mathcal{W}$ be $closure(\mathcal{U}, \{a_1, \ldots, a_n\}, P_1, \ldots, P_k)$ where $P_1, \ldots, P_k$ are the predicates that occur in ϕ. In order to emphasize the origins of $\mathcal{W}$, it is convenient to also write it as $window(\phi[a_1, \ldots, a_n], \mathcal{U})$ or alternatively by $window(A[a_1, \ldots, a_n], \mathcal{U})$, being careful to remember that the domain of this structure includes constants beyond $a_1, \ldots, a_n$ by virtue of closure.

Example 1 *Suppose the pre-condition of action $move(A, B)$ in the blocks world is $\forall X \neg on(X, A) \wedge \forall X \neg on(X, B) \wedge \exists Y on(A, Y) \exists Z \wedge on(B, Z)$ and it is applied to the structure $\mathcal{U}$ in figure 1 with the substitutions $A/a, B/e$, then the substructure $\mathcal{W}$ will have closure domain $\{a, e, b, table\}$, and relations restricted to these.*

This example explains the motivation for the name *window*. In figure 1 the objects in the dashed line box constitute the domain of $\mathcal{W}$. This window of the action $move(a, e)$ encloses all the objects that could possibly be relevant in a test of the applicability of this action. The extent of the window is determined here by the fact that its pre-condition is a $\{a, e\}$-formula, as specified in the definition below.

Definition 6 *Given a set D of domain constants in $\mathcal{L}$, the D-formulas are Boolean combinations of (i) ground atoms (ii) existential (partly) grounded atoms and (iii) negations of existential (partly) grounded atoms. In all instances each atom must have at least one argument that is from D.*

It appears that most practical pre-conditions of actions have as instances *D-formulas* where D is the set of substituted constants in the particular instance of the action. For example, in the instance of the action $move(a, e)$, the pre-condition $\forall X \neg on(X, a) \wedge \forall X \neg on(X, e) \wedge \exists Y on(a, Y) \wedge \exists Z on(b, Z)$ is a $\{a, e\}$-formula. Note that this definition effectively enforces 1-transitivity in P. A 2-transitive formula would have (for a binary P) a subformula instance of the form $P(a, X) \wedge P(X, Y)$—or some such permutation thereof—where a is in D, and unless Y is instantiated to some b in D, this subformula violates the definition of a D formula since $P(X,Y)$ does not have a D constant. However, if indeed Y is some such b, the subformula reads $P(a,X) \wedge P(X,b)$ and is 1-transitive.

So, because of the above reasons, we shall *assume* the following:

Assumption 1 (D-formula Assumption) *If ϕ is a pre-condition of action $A[a_1,\ldots,a_n]$, then $\phi[a_1,\ldots,a_n]$ is a $\{a_1,\ldots,a_n\}$-formula.*

The formal reason for why windows work in practice is provided by the next theorem.

Theorem 1 (Pre-condition Conservative Extension) *Let $\phi[a_1,\ldots,a_n]$ be a pre-condition of an action instance to be applied to state $\mathcal{U}$. If α is any $\{a_1,\ldots,a_n\}$-formula, then $\mathcal{U} \models \alpha$ iff $window(\phi[a_1,\ldots,a_n],\mathcal{U}) \models \alpha$.*

Proof: We proceed by induction on the complexity of α. When α is a ground atom the theorem follows by definition of the closure. For the non-ground basis case it suffices to consider by example the case of $\exists P(a_1,X)$ as more general cases follow the same pattern. The theorem is in this case exactly lemma 1. The inductive steps in the proof of the equivalence are a consequence of the easily verified fact that D-formulas are closed under Boolean operations. □

In other words, the theory of $\mathcal{U}$ is a *conservative extension* [Chang and Keisler 73; Enderton 72] of that of $window(\phi[a_1,\ldots,a_n],\mathcal{U})$ for $\{a_1,\ldots,a_n\}$-formulas. This theorem is the essence of why the pictorial idea of a 'window' cut out of a picture of a state is intuitively appealing. We claim that it captures the notion of 'localness' in such pictures. The theorem states that an instantiated pre-condition is satisfied by the objects in the window if and only if it is satisfied by the entire picture.

3 State Change

Consider now figure 2 which intuitively represents the resultant state of the $move(a,e)$ action applied to the state in figure 1. Note that the window boundary is unchanged from figure 1. This is another characteristic of 'localness' that is captured. We now turn to a formal account of these observations based on the post-conditions of the action instance. Our first assumption is that all the objects and predicates of direct interest in the post-condition are already named in the pre-condition, i.e.,

Assumption 2
(Post-condition Assumption) *If $A[a_1,\ldots,a_n]$ is an action instance with pre-condition $\phi[a_1,\ldots,a_n]$, then all constants of this instance in the post-condition are among the $a_1,\ldots,a_n$, and all the predicates in the post-condition are among those of the pre-condition.*

The next assumption is the interpretation of what it means to add and delete atoms in the corresponding lists. Lifschitz [Lifschitz 87] proposes the very reasonable one that, for atomic add and delete lists, amounts to the:

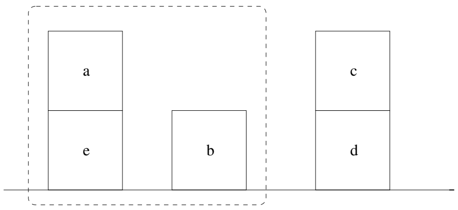

Figure 2: A post-condition window

Assumption 3 (Inertia Assumption) *Let action instance A transform state (structure) $\mathcal{U}$ to state (structure) $\mathcal{U}_1$. Then (i) $\mathcal{U} \models P$ implies $\mathcal{U}_1 \models P$ for any atom P not in the delete-list of A — this is the idea of* inertia or persistence by default*; (ii) if Q is an atom in the add-list, then $\mathcal{U}_1 \models Q$.*

In practice, the second condition seems to apply by default to atoms Q already known to be such that $\mathcal{U} \models \neg Q$. If the Inertia Assumption is understood to dominate pre-conditions to the extent that no atom that persists can be in the add-list (this is not required by the semantics of Lifschitz [Lifschitz 87]), and no atom that does not hold in $\mathcal{U}$ and $\mathcal{U}_1$ is in the delete-list, then we have:

Assumption 4 (Strong Inertia Assumption) *This is as in the Inertia Assumption, except that in (i) we add the qualifier, "and only for atoms P not in the delete-list", and in (ii) we strengthen the consequent to "$\mathcal{U}_1 \models Q$ and $\mathcal{U} \not\models Q$".*

For strong inertia then, nothing is in the add- or delete-list unless it has changed signs in the course of the action. Strong inertia was implicitly used by Reiter in [Reiter 91] in his formulas γ_Q^+ and γ_P^- which combined explanation closure (due to Schubert) and result closure (due to Pednault) in his solution to the frame problem. It is tempting to think that this should be the natural default for designers of specifications. While inertia is a reasonable guide to the design of action specifications in practice, we shall see an example in section 4.2 that strong inertia does not appear to be the intended designer semantics in some STRIPS specifications, i.e. that sometimes (weak) inertia has significance.

For non-atomic add- and delete-lists, we might as well go the whole hog and consider post-conditions which are formulas. Bearing in mind the cautionary statements from [Lifschitz 87], it is prudent to consider only such formulas that are *definite* so that they in effect define a set of ground literals, whose predicate names and constants are subject to the Post-condition Assumption above. Then we apply the Inertia Assumption as before.

The result is that the Post-condition Assumption coupled with these remarks implies:

Theorem 2 (Post-condition Conservative Extension) *If ϕ and ψ are the pre- and post-conditions of action instance A respectively, then with the notation above (i) subject to the Post-condition Assumption the domain of $window(\psi,\mathcal{U}_1)$ is a subset of the domain of $window(\phi[a_1,\ldots,a_n],\mathcal{U})$; (ii) in addition, subject to the Inertia Assumption, the Pre-condition Conservative Extension theorem also holds for*

$window(\psi, \mathcal{U}_1)$ *with respect to* $\mathcal{U}_1$.

Assertion (i) is trivial. Assertion (ii) follows from the Inertia Assumption and reasoning similar to the Pre-condition Conservative Extension theorem. Together, the two theorems confirm that the window construction captures the intuitive notion of 'localness'. The 'genericity' of actions is captured by the following observation (see [Enderton 72; Chang and Keisler 73] for the definition of isomorphism between structures) and the fact that isomorphic structures are elementarily equivalent (see definition 4).

Observation 1 (Isomorphism of Windows) *Any two instances of an applicable action has isomorphic pre-condition, and correspondingly isomorphic post-condition, windows.*

4 Action Invariants

Zhang and Foo [Zhang and Foo 96] have used persistence to derive action invariants from definite pre- and post-conditions. Since the Inertia Assumption is accepted in our formalization, one should expect that a similar derivation is possible here. If so, it would give a formal account of how STRIPS rule designers, for reasons of implementation efficiency, avoid having to specify the *impossible* predicates as action guards by clever selection of add- and delete-lists.

Definition 7 *An invariant of an action A is a formula β s.t., for all instances of A, if β holds in the state (structure) before A is applied, β will hold in the resulting state (structure).*

In the context of several actions (not merely different instances) constraints are simply the set of invariants common to all of them. So, it suffices to search for candidate invariants for each action in order to discover system constraints. In practice, designers of action specifications know many constraints, but they may not be aware of others that are implicit in the action specifications. Conversely, known constraints can be used to specify conditions for actions. In this section we indicate how invariants depend on action specifications.

The notions of pre- and post-condition windows introduced in the earlier sections provide the justification for localized reasoning about action invariants. While this section reports preliminary work toward the elimination of the *impossible* rules, the ideas proposed are prompted by the preceding formal development. We are currently investigating systematic algorithms to fulfill this work.

4.1 Ontological Primacy of Atoms

In many knowledge representation systems there is an implicit priority given to those facts of the world that can be expressed *positively* as ground atoms. This is a *linguistic* convention seldom breached, and no doubt there are strong psychological and cognitive reasons for it. Not being experts in these disciplines, we adduce evidence for what might be called a principle of the *ontological primacy of atoms* by recounting some facts that are widely known in our AI and KR community. First, databases (both deductive and otherwise) record atoms as the primary data. So do pure logic programs. Herbrand models are standard in KR semantics, and many of the characterization theorems of non-monotonic reasoning systems are expressed in terms of properties of such models. The bias toward positivity appears to be widespread even in formal approaches.

We therefore advance the following hypothesis, and then formalize it in the next definition.

Assumption 5 (Positive Literal Bias) *In an action specification the add-list of atoms are determinants of the delete-list of atoms in the presence of constraints.*

Definition 8 *Let C be a set of formulas that express the constraints of the system. A set Γ of atoms determines another set Λ if $C \cup \Gamma \models \Lambda$.*

The Positive Literal Bias hypothesis then translates to this formal proposal. Let A be an action that is applied to state $\mathcal{U}$ with resultant state $\mathcal{U}_1$, and with add-list Γ of atoms, delete-list Λ of atoms. Let C be (possibly unknown) system constraints. Then $C \cup \Gamma \models \neg \bigvee \Lambda$, which is equivalent to $C \models \bigwedge \Gamma \supset \neg \bigvee \Lambda$. The notation $\bigvee \Lambda$ means the disjunction of the atoms in the set Λ. This means that $\bigwedge \Gamma \supset \neg \bigvee \Lambda$ is a candidate invariant for the action A if its constants are generalized. This generalization is possible because of observation 1 on isomorphisms of windows.

Our proposal is in the spirit of Explanation Closure attributed to Schubert [Reiter 91]. Indeed the assertion $C \cup \Gamma \models \neg \bigvee \Lambda$ has the structure of an abduction of the negative literals in terms of the positive ones with respect to C.

4.2 Generating Candidate Invariants

Let us now apply the above proposal to a few examples. Several caveats will be revealed in the process, which is to be thought of as a procedure for generating plausible guesses at invariants which will be correct if the Positive Literal Bias hypothesis holds along with some weak requirements.

For the $move(a, e)$ action, the add-list is $on(a, e)$ and the delete-list is $on(a, b)$, so by the proposal we have the candidate invariant $on(a, e) \supset \neg on(a, b)$, which generalizes (using unique names) to $(X \neq Y) \supset (on(A, X) \supset \neg on(A, Y))$ which is just a fancy way of saying that no block is in different places, i.e., the Prolog rule: *impossible :- on(A,X), on(A,Y), X* $\neq$ *Y* used as a guard for checking the move action. However, the generalization implies that every block has been moved before, since the generalization relies on an isomorphism of result windows and these are the effects of the action. This motivates the following definition.

Definition 9 *A ground atom P is actualised by an action A if it was false in some initial window satisfying the pre-condition of A but true in the resulting window.*

By earlier remarks, we can replace 'window' by 'state', and equivalently say P satisfies the post-condition of A. Then the proposed candidate invariant is plausible for every A-actualised atom. The *soundness* of the proposed invariant is not in question for the following reasons. Let $P_1, \ldots, P_n$ be the add-list of an instance of action A, and $Q_1, \ldots, Q_m$ be the delete-list. Then in the resultant state it is the case that $\bigwedge_{i=1}^{i=n} P_i \wedge \bigwedge_{i=1}^{i=m} \neg Q_i$ and this implies the candidate invariant in our proposal, which is $\bigwedge_{i=1}^{i=n} P_i \supset \bigwedge_{i=1}^{i=m} \neg Q_i$. In fact, if we assume strong inertia as in assumption 4, then all of $P_1, \ldots, P_n$ would have been false in the initial state, so the conjunctive formula $\bigwedge_{i=1}^{i=n} P_i \wedge \bigwedge_{i=1}^{i=m} \neg Q_i$ is false in it, and cannot be an invariant, while the weaker formula in question holds in it. To confirm that this is actually an invariant in our formalization an induction is necessary, but in [Zhang and Foo 96] a persistence logic has been used to verify it.

One final remark: the *clear* predicate used in some implementations has not been used in the blocks-world example here as it is eliminable. For technical reasons our proposal requires non-eliminable predicates.

4.3 Weak Inertia

In this rather preliminary subsection we show an example where strong inertia is not the intended semantics of a STRIPS specification.

Example 2 *Consider a duopoly law for the ownership of a television station. Current owners may sell to others, but at all times there must be two owners. Here is the STRIPS specification for the action buy(A,E) meaning informally that owner E is selling his share of the station to A.*

action: buy(A,E)
pre-condition: own(B,T), own(E,T), $A \neq E$
post-condition:
add-list: own(A,T)
delete-list: own(E,T)

Please note the formal similarity of this situation with that of *move(a,e)* that we discussed earlier. In any instance of $buy(a, e)$, the window of the action will contain in addition to $own(a, t)$ an instance $own(b, t)$ which will persist through the action but is not in the add-list. However, the candidate invariant using strong inertia is formally similar to that of the $move$ action in the blocks world, viz., $(A \neq B) \supset (own(A, T) \supset \neg own(B, T))$ which is too weak to prevent a monopoly.

Consider now how *weak inertia* may be used here. If we change the add-list to $own(A, T), own(B, T)$, a candidate invariant is $own(A, T) \wedge own(B, T) \wedge (E \neq A \wedge E \neq B \wedge A \neq B) \supset \neg own(E, T)$ which is intuitively what the specifier intended under the unique names assumption. Evidently, weak inertia has different designer semantics.

5 Conclusion

We have provided formalizations of the ideas of 'localness', 'genericity', and 'implicit constraints' in STRIPS-like specifications of actions. The first two are very faithful to the intuitions of designers, but the third is not quite settled as it appears that two different kinds of inertia are assumed. The first two formalizations have potential uses in diagrammatic reasoning, while the third suggests opportunities for identifying the psychology of action specifications with appropriate model classes for which inductive proofs of candidate invariants can be avoided. There is an increasing interest in diagrammatic reasoning [Barwise 95] in AI and we shall be extending the window semantics in this direction. We are currently pursuing the identification of dynamic system classes in the spirit of Sandewall [Sandewall 94] to see how invariants can be extracted completely from specifications without induction. There appear to be at least three notions of completeness for invariants that can be formalised.

References

[Barwise 95] Barwise, J. and Etchemendy, J., "Heterogeneous logic", in *Diagrammatic Reasoning: Cognitive and Computational Perspective*, eds. J. I. Glasgow, et al. MIT Press, 1995, 209-232.

[Chang and Keisler 73] Chang, C.C., and Keisler, H.J., *Model Theory*, North Holland, 1973.

[Enderton 72] Enderton, H., *A Mathematical Introduction to Logic*, Academic Press, 1972.

[Fikes and Nilsson 71] Fikes, R.E. and Nilsson, N.J., " STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", *Artificial Intelligence*, 2, 1971, 189-208.

[Lin 95] Lin, F., "Embracing Causality in Specifying the Indirect Effects of Actions", Proceedings of the 14th International Joint Conference in Artificial Intelligence, IJCAI-95, 1995.

[Lifschitz 87] Lifschitz, V., "On the Semantics of STRIPS", in *Reasoning about Actions and Plans*, ed. M. Georgeff and A. Lansky, Morgan Kaufmann, 1987, 1-9.

[Reiter 78] Reiter, R., "On Closed World Data Bases", in *Logic and Database*, ed., Gallaire, H. and Minker, J., Plenum Press, 1978, 55-76.

[Reiter 91] Reiter, R., "The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes)and a Completeness Result for Goal Regression", in *Artificial Intelligence and Mathematical Theory of Computation*, ed. V. Lifschitz, Academic Press, 1991, 359-379.

[Sandewall 94] Sandewall, E., *Features and Fluents; the Representation of Knowledge about Dynamical Systems, Vol. I*, Oxford University Press, 1994.

[Zhang and Foo 96] Zhang, Y. and Foo, N., "Deriving Invariants and Constraints from Action Theories", *Fundamenta Informaticae*, 30:23-41, 1996.