Minimal Semantics for Action Specifications in First-order Dynamic Logic

Jan Broersen*       Roel Wieringa*

Abstract

In this paper we investigate minimal semantics for First Order Dynamic Logic formulas. The goal is to be able to write action specifications in a declarative pre/post-condition style. The declarative specification of actions comes with some well known problems: the frame problem, the qualification problem and the ramification problem. We incorporate the assumptions that are inherent to both the frame and qualification problem into the semantics of Dynamic Logic by defining orderings over Dynamic Logic models. These orderings allow us to identify for each declarative Dynamic Logic action specification a unique intended model. This unique model represents the system that must be associated with the specification given the prefential semantics that is defined by the orderings.

1 Introduction

We investigate the use of Dynamic Logic [11] [12] for the writing of system specifications. DL is designed to reason about arbitrary programs/actions (in the propositional case) or programs built up from specific atomic actions like assignments (in the first order case). Dynamic Logic is a multi modal logic in which each action is accompanied by its own modal operators. We focus on specific DL-formulas that are used to specify conditional postconditions, guards and static constraints for actions. Actions that are susceptible to specification by these means, can be database transactions or system transactions of any other kind, as long as they are taken to be atomic. The only requirement we impose on (atomic) actions is that they are terminating processes of which the intermediary states are not observable. Actions can be nondeterministic. That is, an action may result in one of several possible next states.

Declarative action specifications with the help of pre and postconditions always comes with the frame problem [5] and the qualification problem [9]. These are very common themes in AI, but we focus on these problems in the more confined context of system specification. Because of this, our work is of interest for fields closely related to system specification such as reachability analysis and code generation.

Our aim is to incorporate the frame assumption and the qualification assumption into an intuitive semantics for dynamic logic formulas. Furthermore, the semantics should also deal with these assumptions in the presence of static constraints, either interpreted

---

1Faculty of Mathematics and Computer Science, Vrije Universiteit, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands, email: broersenjroelw@cs.vu.nl. Supported by VU-USF as part of the SINS project and supported by the Esprit Working group Aspire, contract nr. 22704
as derivation rules or as limitations on the state space. See [4] [3] for more details on these problems. We accomplish this by defining orderings over Kripke structures. The preferential semantics associated with these orderings gives us an exact unique meaning (model) for each specification.

The structure of the paper is as follows. In Section 2 we introduce a variant of Dynamic Logic (DL) and define a semantics for specifications in DL. In Section 3, we discuss the frame, qualification and ramification problems in more detail and indicate their interconnections. We propose several preferential semantics for DL specifications to deal with these problems. Section 4 compares our approach with other approaches. Section 5 concludes the paper and lists some topics for further research.

2 Dynamic Logic

Dynamic logic (DL) is a logic to reason about terminating programs. It relates assertions about composite programs to assertions about its parts and vice versa.

We refer to atomic actions by the meta-variables $a, b, \ldots$. The modal formulas $\langle a \rangle \phi$, where $a$ is an atomic action, mean that there is a possible occurrence of $a$ after which $\phi$ holds. The standard DL-semantics of (atomic) actions as relations over states is given in the next Section. We also consider compound actions; $\alpha; \beta$, $\alpha \cup \beta$ and $\alpha^*$ represent sequential composition, choice and iteration, respectively. We use the metavariable $\alpha$ to refer to arbitrary actions. The semantics of these constructs is of importance when proving properties of specifications. They are however not used for the writing of specifications.

2.1 The DL language

Definition 1 The language is built with the following elements:

- **Punctuation symbols:**
  The brackets ‘)’, ‘(’ and the comma ‘,’.

- **Variables:**
  A countable set $V$ of variable symbols. The associated meta-variables are $x, y, \ldots$.

- **Predicates:**
  A countable set $P$ of predicate symbols, including one distinguished predicate denoted by $\ldots$. The associated meta-variables are $P, Q, \ldots$.

- **Functions:**
  A countable set $F$ of function symbols.

- **Propositional connectives:**
  The set of symbols $\{\neg, \vee\}$

- **Quantifiers:**
  The set of symbols $\{\exists\}$.

- **Atomic actions**
  A countable set $AA$ of atomic action symbols. The associated meta-variables are $a, b, \ldots$.

- **Action connectives:**
  The set of symbols $\{\cup, ^*\}$
**Combined connectives:**
the set of operation symbols \{(), ?\}

The intended use of the logic is the writing of system specifications. A specific specification uses only a finite subset of the language. A specifier has to give this subset in a signature. He will also be asked to provide arities for predicate and function symbols. The arity of a predicate or function prescribes a length for the term lists concatenated to it.

**Definition 2** A **signature** \( \Sigma \) is thus defined as a specific combination of finite subsets of the predicate, function and atomic action symbols: \( \Sigma = (P, F, AA) \).

A signature contains the symbols that are to be given an interpretation relative to a specification. All other symbols are given logical interpretations.

**Definition 3** An **atomic formula** \((\text{atom})\) over a signature \( \Sigma \) is defined as:

\[
\begin{align*}
term &::= \text{variable} \mid \text{constant} \mid \text{function}^n \ (\text{termlist}_n) \\
costant &::= \text{function}^0 \\
termlist_1 &::= \text{term} \\
termlist_n &::= \text{termlist}_{n-1}, \text{term} \\
atom &::= \text{predicate}^n \ (\text{termlist}_n)
\end{align*}
\]

An **action** \((\alpha)\) over a signature \( \Sigma \) is defined as:

\[
\alpha ::= \text{atomic action} \mid \alpha \cup \beta \mid \beta \mid \alpha^* \mid \phi?
\]

A **well formed formula** \((\phi)\) over a signature \( \Sigma \) is defined as:

\[
\phi ::= \text{atom} \mid \top \mid \bot \mid \neg \phi \mid \phi \lor \psi \\
\forall \ \text{variable} \phi \mid \langle \alpha \rangle \phi
\]

We use \(\phi, \psi, \chi, \ldots\) as meta variables over well formed formulas. We abbreviate \(\neg(\neg \phi \lor \neg \psi)\) to \(\phi \land \psi\), \(\neg \phi \lor \psi\) to \(\phi \rightarrow \psi\), \((\phi \rightarrow \psi) \land (\phi \rightarrow \psi)\) to \(\phi \equiv \psi\), \(\neg(\alpha)\neg \phi\) to \([\alpha] \phi\) and \(\neg \exists x \neg \phi(x)\) to \(\forall x \phi(x)\).

**Definition 4** A **specification** \(\text{Spec} = (\Sigma, \Phi)\) is a pair consisting of a signature \( \Sigma \) and a finite set \( \Phi \) of well formed formulas over \( \Sigma \).

### 2.2 The semantics of DL

We want to model the effects or implications of actions as accessibility relations over states. So the differences between states should only constitute in what can be affected by actions. In general actions can affect interpretations of predicates and functions. A simplifying, but justified assumption is that each state of the 'system under specification' is identified with a possible interpretation of the predicate and function symbols. This means we rule out two possibilities with respect to the standard notion of a (Kripke) model in Modal Logic.
The first is that we do not consider different states with equal interpretations within one model. Considering different states with equal interpretations does not comply with the intuition that the states represent system states. All state information is present in the interpretation of the predicate and function symbols. If info about how a state is reached should be known to the system, then this must be encoded in the interpretation of predicate or function symbols. This agrees with our intuition that a state of our model represents a system state.

The second difference with standard Kripke models is that we do not consider different interpretations of predicates in the same state. Usually in Modal logics, the notion of 'frame' is defined. A frame represents the accessibility relations between possible states independent of the interpretation of symbols within these states. So different interpretations of predicates in the same state can be considered and the notion of 'validity on a frame' can be defined. Properties of frames reflect properties of the structure of the state (in our case system) that the formulas are about. However, these properties can also be defined as properties of the accessibility relation of each (Kripke) model, which means that we don’t really need the notion of 'frame'.

So we do not discriminate between states and interpretations of predicates and functions. Therefore we don’t need an interpretation function to interpret predicates in states, we just identify states with interpretations of the predicate and function symbols.

**Definition 5** A possible state, or system state over an arbitrary domain $D$ is a tuple $(I_P, I_f)$. $I_P$ and $I_f$ are functions that interpret predicate and function symbols of a signature $\Sigma = (P, F, AA)$ over the domain $D$.

- $I_P$ is a function $P \to 2^{D^n}$, equivalently $I_P(P^n) \subseteq D^n$
- $I_f$ is a function $F \to (D^n \to D)$, equivalently $I_f(f^n)$ is a function $D^n \to D$

$I_P$ is an interpretation of the predicate symbols and $I_f$ is an interpretation of the function symbols. Possible states are referred to as $s, s', \ldots$. In a specification $(\Sigma, \Phi)$, the formulas in $\Phi$ are interpreted over a special kind of Kripke structures.

Variables are not updated and thus play no role in the definition of a state. A variable in a formula is a symbol that is used to quantify over domain elements or that represents an arbitrary domain element. Our variables are not state-carrying symbols.

**Definition 6** Given a signature $\Sigma = (P, F, AA)$, a structure $S = (D, S, I_{AA})$ is defined as follows:

- $D$ is an arbitrary domain
- $S$ is a nonempty set of possible states over $D$
- $I_{AA}$ is a total function $AA \to 2^{S \times S}$, equivalently $I_{AA}(a) \subseteq S \times S$

The definition states that atomic actions get their meaning from the differences in interpretation of Function and Predicate symbols. We do not want to interpret (atomic) actions over anything else than these differences. There is one other possible source for differences of interpretations between states: free variables. (Bound variables of course can never form the basis of differences of interpretation between states; their only role is to define the interpretation of quantified formulas, given the interpretation of function and predicate symbols.) We do not want to interpret actions over pairs of states that differ in their interpretation of free variables; actions can only influence the interpretation
predicate and function symbols. Therefore we impose that the assignment of free variables to domain elements is common to all states. This implies that we need a Domain that is common to all states, because if we allow domain elements to vanish when going from one state to the other, the assignment of free variables can only be ”saved” by assigning variables of which the previously assigned value has vanished, a new value. But this means that the interpretation of the free variables changes from state to state, which is not what we want. (A monotonically growing domain is no solution, because cycles are not excluded.) This is completely different from the situation in DL [11]. In DL actions (programs) are interpreted to modify values (bindings) of free variables. In DL it is the variables that are interpreted different in different states, while the interpretation of predicates, functions and constants stays the same. Note however that function symbols of arity 0 can be used as variables in the sense of traditional DL.

Variables and terms are interpreted in a standard way, as follows from the next definitions.

**Definition 7** Given a structure \( S = (D, S, I_{AA}) \) and a set of variables \( V \), an assignment \( I_V \) is a function \( V \rightarrow D \), assigning a domain element to each variable in \( V \).

**Definition 8** An interpretation \( I_t \) of a term \( t \) in a possible state \( (I_P, I_f) \) of a structure \( S \), given an interpretation of variables \( I_V \) is defined as follows.

- \( I_t(t) = I_V(t) \) in case \( t \) is a variable
- \( I_t(t) = I_f(f^n)(I_t(t_1), \ldots, I_t(t_n)) \) in case \( t \) has the form \( f^n(t_1, \ldots, t_n) \)

The next definition gives the interpretation of the action part of DL-formulas and of complete DL-formulas. These two can not be separated, because the definition of the action test \( \phi \) as a binary relation over states uses the notion of validity of a formula in a state, and the definition of validity of a formula after a compound transition \( \alpha \) uses the interpretation of compound actions as binary relations over states.

**Definition 9** Given a signature \( \Sigma = (P, F, AA) \), a structure \( S = (D, S, I_{AA}) \), with \( S = (I_P, I_f) \) and an assignment \( I_V \), the interpretation of an action \( \alpha \) denoted as \( I_A(\alpha) \) and validity of a wff \( \phi \) in a state \( s \) of the structure \( S \), denoted by \( S, s, I_V \models \phi \) are defined as:

- \( I_A(\alpha) \) \( \equiv \) \( I_{AA}(\alpha) \) if \( \alpha \in AA \)
- \( I_A(\alpha \cup \beta) \) \( \equiv \) \( I_A(\alpha) \cup I_A(\beta) \)
- \( I_A(\alpha; \beta) \) \( \equiv \) \( I_A(\alpha) \circ I_A(\beta) \)
- \( I_A(\alpha^*) \) \( \equiv \) \( (I_A(\alpha))^* \)
- \( I_A(\phi?) \) \( \equiv \) \{ \( (s, s) \mid S, s, I_V \models \phi \} \)
- \( S, s, I_V \models \bot \) \( \equiv \) never
- \( S, s, I_V \models t_1 = t_2 \) \( \equiv \) \( I_t(t_1) \) returns the same domain element as \( I_t(t_2) \)
- \( S, s, I_V \models P_i^n(t_1, \ldots, t_n) \) \( \equiv \) \( (I_t(t_1), \ldots, I_t(t_n)) \in I_P(P_i^n) \)
- \( S, s, I_V \models \phi \lor \psi \) \( \equiv \) \( S, s, I_V \models \phi \) or \( S, s, I_V \models \psi \)
- \( S, s, I_V \models \neg \phi \) \( \equiv \) not \( S, s, I_V \models \phi \)
- \( S, s, I_V \models \exists x \phi(x) \) \( \equiv \) for some \( d \in D \) holds \( S, s, I_V \{ x \mapsto d \} \models \phi(x) \)
- \( S, s, I_V \models \langle \alpha \rangle \phi \) \( \equiv \) for some \( s' \in S \) holds \( (s, s') \in I_A(\alpha) \) and \( S, s', I_V \models \phi \)
- \( S, s, I_V \models \top \) \( \equiv \) always

We see that \( \phi? \) more or less produces its own interpretation as a binary relation between states. This is possible because its interpretation really does not depend on the
accessibility properties of the structure, its interpretation only depends on the validity of \( \phi \) in a particular state.

A formula is S-valid if it is valid in all states of a structure S. In that case, we say the structure satisfies the formula and that the structure is a model of the formula. A formula is valid if it is S-valid for every structure S. Note that this is not entirely conventional. In literature, usually the word ’structure’ is reserved for frames, consisting only of states and an accessibility relation, without a valuation. In our definitions states are identical with interpretations. What we call a structure is in literature on modal logic usually called a model. We prefer to use the word model for structures that satisfy a set of formulas.

3 Action specification

We investigate the use of PDL for the writing of specifications for atomic actions. We focus on two aspects: the effect and the possible occurrence of actions. For the specification of the effect of an action \( a \) we can use formulas called ”conditional postcondition formulas”.

- **Conditional postcondition formulas**: \( \phi \rightarrow [a] \psi \) We say that \( \phi \) is a sufficient precondition of \( a \) with respect to the postcondition \( \psi \). We denote sets of these formulas by \( \Phi_{post} \).

The possible occurrence of actions can be controlled with ”guard formulas”.

- **Guard formulas**: \( \langle a \rangle \top \rightarrow \chi \) We call \( \chi \) a guard of \( a \), equivalently, a necessary precondition for the possible occurrence of \( a \). We denote sets of these formulas by \( \Phi_{guard} \).

Guard formulas actually can be seen as a special case of conditional postcondition formulas. This is however of no importance. What is important is that we can express guards (necessary preconditions for actions) in PDL. Furthermore it is important to investigate guards independently because they are strongly interconnected with the qualification problem.

We also want to investigate the influence of ”static constraints” (formulas without modalities) on the effect and possible occurrence of actions.

- **Static constraints**: \( \theta \) These are non-modal assertions that must be obeyed under any circumstance. We denote sets of these formulas by \( \Phi_{IC} \).

The formulas \( \phi, \psi, \theta \) and \( \chi \) contain no modal constructs; they are just propositional logic formulas. To simplify the setting we assume that these are the only formulas a specifier uses when stating a specification. This follows the syntactic restrictions of LCM [7]. It is motivated by the fact that we simply don’t need intricately nested modalities to specify actions declaratively.

The problem we are going to address in the following sections is that generally the intended semantics of a specification made with this type of formulas does not coincide with the standard DL-semantics.
The standard PDL-interpretation of postcondition formulas, is that when $\phi_i$ is true, action $a$, if present, leads to a situation where $\psi_i$ is true. The formulas describe the effect of an action. However, defining the effect of an action by means of a postcondition always causes the frame problem \cite{5}. This problem states that when specifying a postcondition we only want to specify conditions that have changed. We do not want, and often are not able, to specify all the conditions that do not change as the result of an action. So we want to make the frame assumption that everything that has not been specified to change has not changed. However, this is not reflected by the semantics, as defined in \ref{2}.

In the following we will define an ordering over DL-structures that reflects the intention to interpret DL-specifications under the assumption that actions do not change interpretations of function and predicate symbols if this is not explicitly expressed by the specification. We want the ordering to compare models on the property of 'access to closest possible states'. We first define a notion of distance between states.

**Definition 10** Given a structure $S = (D, S, I_{AA})$ and two states $s = (I_P, I_f)$ and $s' = (I'_P, I'_f)$ in $s$. The difference $\text{Diff}(s, s')$ between them is defined as the set of predicate instances and function values in which the states differ ($N(V)$ is the cardinality of a set $V$):

$$
\text{Diff}(s, s') = \left\{ \left\{ P^n_i, (d_1, \ldots, d_n) \right\} \mid (d_1, \ldots, d_n) \in I_P(P^n_i) \text{ and } (d_1, \ldots, d_n) \not\in I'_P(P^n_i) \right\} \cup 
\left\{ \left\{ P^n_i, (d_1, \ldots, d_n) \right\} \mid (d_1, \ldots, d_n) \not\in I_P(P^n_i) \text{ and } (d_1, \ldots, d_n) \in I'_P(P^n_i) \right\} \cup 
\left\{ \left\{ f^n_i, (d_1, \ldots, d_n) \right\} \mid I_f(f^n_i)(d_1, \ldots, d_n) \not= I'_f(f^n_i)(d_1, \ldots, d_n) \right\}
$$

We want to use this measure of distance between states in the comparison between different DL-structures. We define an ordering over Kripke structures that is connected to the definition of difference between states in such a way that structures where actions lead to 'closer' states are lower in the ordering. We will now first give the two slightly different orderings, before we explain what they are really about.

**Definition 11** Given a signature $\Sigma = (P, F, AA)$ and two structures $S = (D, S, I_{AA})$ and $S' = (D, S', I'_{AA})$,

$$S' \sqsubseteq_{mc} S \iff
S' = S
\text{ and}
\forall a \in AA, \forall s \in S, \exists s' \in S, (s, s') \in I'_{AA}(a) \Leftrightarrow \exists s'' \in S, (s, s'') \in I_{AA}(a)
\text{ and}
\forall a \in AA, \forall s \in S, \forall s' \in S, ((s, s') \in I'_{AA}(a) \Rightarrow \exists s'' \in S, ((s, s'') \in I_{AA}(a)
\text{ and}
\wedge N(\text{Diff}(s, s')) \leq N(\text{Diff}(s, s''))\right)$$

$\sqsubseteq_{mc}$ is a pre-order on structures, because $\sqsubseteq_{mc}$ can easily seen to be transitive and reflexive. MC stands for minimal cardinality. A structure is called an MC-model of $\Phi$ if it is a $\sqsubseteq_{mc}$-minimal model of $\Phi$. MC-models determine the minimal cardinality interpretation (semantics) of a specification $\langle \Sigma, \Phi \rangle$. 

7
Definition 12 Given a signature $\Sigma = (P, F, AA)$ and two structures $S = (D, S, I_{AA})$ and $S' = (D, S', I'_{AA})$, 

$$S' \sqsubseteq_{ms} S \iff$$

$$S' = S$$

and

$$\forall a \in AA, \forall s \in S, (\exists s' \in S, (s, s') \in I_{AA}(a) \Rightarrow \exists s'' \in S, (s, s'') \in I'_{AA}(a))$$

and

$$\forall a \in AA, \forall s \in S, \forall s' \in S, ((s, s') \in I'_{AA}(a) \Rightarrow \exists s'' \in S, ((s, s'') \in I_{AA}(a))$$

$$\wedge (\text{Diff}(s, s') \subseteq \text{Diff}(s, s'') \lor \text{Diff}(s, s') \supseteq \text{Diff}(s, s''))$$

$\sqsubseteq_{ms}$ is a pre-order on structures, because $\sqsubseteq_{ms}$ can easily seen to be transitive and reflexive. MS stands for minimal subset. A structure is called an MS-model of $\Phi$ if it is a $\sqsubseteq_{ms}$-minimal model of $\Phi$. MS-models determine the minimal subset interpretation (semantics) of a specification $(\Sigma, \Phi)$.

The first requirement in both $\sqsubseteq_{mc}$ and $\sqsubseteq_{ms}$ ordering is that structures can only be compared if they are based on the same set of states. In figure 1, where three models are compared on minimal change, this is reflected by the fact that all models contain the same set of black dots. This is actually not a necessary condition for the definition of the semantics, but it helps to make the definition more understandable. Models with exactly the same state transitions but differing in states that 'stand alone', can not be compared. If we would drop this condition such states would be comparable, but one could never be prefered above the other. The question what states are actually in the intended model of a specification is dealt with by a second ordering, to be defined in 3.2.

The second requirement in both orderings forces that structures can only be compared if each transition from a state in one of the models actually corresponds to a transition from the same state in the other model, that is comparable under the minimal cardinality respectively the minimal subset criterion. For two transitions to be comparable under the minimal cardinality criterion, it is sufficient to demand that they leave from the same state; the comparison is just made on the number of changes that both transitions bring about. For two transitions to be comparable under the minimal subset criterion, the changes of the first must be a subset of the changes of the second or the other way around. In figure 1 this is represented by the fact that if an arrow leaves from some state in model there is also an arrow from this states in other models. (The transitions may lead to different states. This is actually where we want to compare structures on; we want to prefer structures where transitions lead to closer states.) This is an intuitive criterion because we don’t want this ordering to deal with the possible occurrence of transitions (actions); this ordering should compare structures purely on the 'length' of transitions. (This observation is also made by Brass and Lipock [2] [14].)

The last requirement deals with this 'length' of transitions. In words it says: if $S' \sqsubseteq_{mc} S$ then for all transitions $(s, s')$ in $S'$ there is a transition $(s, s'')$ in $S$ that is 'longer'. In yet other words: if $S' \sqsubseteq_{mc} S$ then for corresponding transitions in corresponding states in the compared structures, the 'longest' transition in $S'$ is always less or equal to the 'longest' transition in $S$.(To see why the ordering is not a partial order: when the 'longest' transitions from the same states in both structures are equal, still both structures can differ in transitions that are 'shorter’ then these 'longest’ transitions.) In figure 1 the distance between dots represents the difference between states. Clearly the minimal one
is the one for which the property holds. The reason why this looks rather complicated is that we allow non-deterministic actions; we have to compare structures in which actions from one state can lead to several other states. In 3.4 we will see that for deterministic models orderings can be defined much simpler.

The difference between the semantics generated by both orderings will become more clear when we look at an example in 3.5 and in the following section where we study minimal models for several specification classes. But we already can say that the minimal subset semantics is weaker than the minimal cardinality semantics, as is expressed by the following proposition.

**Proposition 1** An MC-model of $\Phi$ is also an MS-model of $\Phi$.

**Proof** Assume $M$ is an MC-model but not an MS-model of some set of formulas $\Phi$. If $M$ is not an MS-model, there has to be a model $M'$ with a transition $a$ from a state $s$ to a state $s'$ that is comparable to a transition $a$ in $M$ from $s$ to a state $s''$ and that brings about a subset of the changes brought about by the transition in $M$. Surely this model is also comparable to $M$ in the minimal cardinality ordering, and because the transition $a$ in $M'$ changes a subset of the corresponding transition in $M$, there are also less changes. This means that $M'$ is also below $M$ in the minimal cardinality ordering, which leads to a contradiction.

3.1.1 Minimal models for different specification classes

We will now investigate properties of minimal models for several classes of specification formulas. The properties we study are ”existence of minimal models”, ”equivalence of the minimal cardinality and minimal subset criteria” and ”determinism of minimal models”.

We start of with the most general class we consider (We use the convention that a variable $v$ ranges from 1 to $v'$).

**Definition 13** the description of CLASS I formulas:

$$\Phi_{\text{post}} : \quad \phi \rightarrow \{a\}(P_{11} \land \ldots \land P_{1v'}) \lor \ldots \lor (P_{p'1} \land \ldots \land P_{p'v'})$$

$$\Phi_{\text{guard}} : \quad \langle a \rangle \top \rightarrow \chi$$

$$\Phi_{\text{IC}} : \quad (C_{11} \land \ldots \land C_{1v'}) \lor \ldots \lor (C_{p'1} \land \ldots \land C_{p'v'})$$
with the $P_{ij}$ and $C_{ij}$ positive or negated atomic formulas (literals), with no restrictions on the quantification of variables whatsoever.

An important property of minimal models for this most general class we consider is that if there is a model with a non-empty accessibility relation, then there is a minimal model with a non-empty accessibility relation.

**Proposition 2** Let $\Phi$ be a set of formulas of CLASS I that has a model $M = (D, S, I_{AA})$ for which $I_{AA}(a) \neq \emptyset$ for all $a$ in the formulas. Then there is an MS-model $M_{MS} = (D, S, I_{AA}')$ of $\Phi$ for which $I_{AA'}(a) \neq \emptyset$ for all $a$ in the formulas and an MC-model $M_{MC} = (D, S, I_{AA}'')$ of $\Phi$ for which $I_{AA''}(a) \neq \emptyset$ for all $a$ in the formulas.

**Proof**

We prove this property with the help of formula transformations that preserve the property that minimal models exist. First we define a shorthand for a set of formulas of CLASS I:

$$\phi^i \rightarrow [a](\forall \land P_{pq}^i)$$

$$\langle a \rangle \top \rightarrow \chi$$

$$\forall \land C_{rs}^k$$

The superscript indices represent that there may be several formulas of that form in the set. (We use only one guard, because we look at $\chi$ as the concatenation of all necessary preconditions for $a$.) The modal formulas in the set only refer to the action $a$. This is no restriction, because modal formulas not containing $a$ clearly can not prevent the existence of a minimal model with $I_{AA}'(a) \neq \emptyset$ given the info that there is a model with $I_{AA}(a) \neq \emptyset$.

Now assume that this set has no minimal model. Then the set of formulas

$$\phi^i \rightarrow [a](\forall \land P_{pq}^i \land \forall \land C_{rs}^{1} \land \ldots \land \forall \land C_{rs}^{k'})$$

$$\langle a \rangle \top \rightarrow (\chi \land \forall \land C_{rs}^{1} \land \ldots \land \forall \land C_{rs}^{k'})$$

$$\forall \land C_{rs}^k$$

also has no minimal model. This is trivial because this set obviously is logically equivalent to the former, since we only added the constraints as necessary preconditions and as postconditions. This was already true for the former set of formulas, because constraints hold in all states.

But then the set of formulas

$$\phi^i \rightarrow [a](\forall \land P_{pq}^i \land \forall \land C_{rs}^{1} \land \ldots \land \forall \land C_{rs}^{k'})$$

$$\langle a \rangle \top \rightarrow (\chi \land \forall \land C_{rs}^{1} \land \ldots \land \forall \land C_{rs}^{k'})$$

also has no minimal model. If it had, this minimal model already is or can be easily transformed into a minimal model of the former set of formulas. The only way in which a minimal model for this set can not be a minimal model of the former set is a different interpretation of predicate and function symbols in states where no $a$ enters or leaves (isolated states). This means that every minimal model for this set corresponds to at least one minimal model of the former set, namely a model where the interpretation of propositions in isolated states is changed as to comply with the constraints. (note that this is always possible, because we started off with the assumption that there is a transition $a$, which can not be the case if the constraints contradict each other)

Now we rearrange the formulas to the following set
\( \phi^i \rightarrow [a](\bigvee Q^i_{uv}) \)
\( (a) \top \rightarrow \chi' \)

Both \( \bigvee Q^i_{uv} \) and \( \chi' \) cannot be equivalent to \( \perp \), because then the postconditions or the preconditions for \( a \) in the original set of formulas would be inconsistent with the constraints, which contradicts the assumption we started off with. (Note that at this point we have eliminated the constraints out of the problem.)

But now it follows that there is some \( u \) for which the set
\( \phi^i \rightarrow [a](\bigwedge Q^i_{uv}) \)
\( (a) \top \rightarrow \chi' \)

has no minimal model, because if for all \( u \) the corresponding sets of this form would have a minimal model we could compare this finite set of models in one of the orderings and arrive at a model that has to be minimal for the former set of formulas.

But now we are at the point that it is not all difficult to construct a minimal model from the formulas we have, given there is a model containing a transition \( a \). This means we will arrive at a contradiction, which proofs the proposition. The construction focuses on each transition \( a \) from a state \( s \) to a state \( s' \) in the model.

- Make all \( s' \)-interpretations of function symbols equal to their \( s \)-interpretation (for the moment we assume we do not have equality in the language, otherwise this would become even more complex).

- Make all \( s' \)-interpretations of predicate symbols that are not in the literals \( Q^i_{uv} \) equal to their \( s \)-interpretation.

- Leave all \( s' \)-interpretations of predicate symbols that are in literals \( Q^i_{uv} \), whose \( s' \)-interpretation is equal to their \( s \)-interpretation, as they are.

- Now look at the interpretation of predicate symbols that are in literals among \( Q^i_{uv} \) whose \( s' \)-interpretation is not equal to their \( s \)-interpretation, and that contain a variable that is existentially quantified. Force a minimal interpretation of the existential quantifier by adapting the interpretation of the predicate symbols in such a way that there is precisely one assignment of the existentially quantified variable that causes the literal to be true.

All of these transformations do not make the model invalid and the result is a model that is minimal.

For this type of specifications MS-models and MC-models do not coincide and are not deterministic, as can be shown by a simple propositional example.

**Example 1** Consider the specification:
\[
\begin{align*}
[dial\ number](get\ connection \lor get\ busy\ tone) & \rightarrow costs\ money \\
get\ connection & \rightarrow \neg costs\ money \\
(dial\ number)\top & \rightarrow \neg costs\ money \land \neg get\ connection \land \neg get\ busy\ tone
\end{align*}
\]

Under the minimal subset criterion both the state where \( get\ connection \) holds and the state where \( get\ busy\ tone \) holds is reachable. Under the minimal cardinality criterion only the state where \( get\ busy\ tone \) holds is reachable.
We now turn our attention to quantification. Existential quantification is a source of non-determinism, as will be clear from the second class we study.

**Definition 14** the description of CLASS II formulas:

\[
\Phi_{\text{post}} : \phi \rightarrow [a] L_1 \land L_2 \land \ldots \land L_k \\
\Phi_{\text{guard}} : \langle a \rangle \top \rightarrow \chi \\
\Phi_{\text{IC}} : (M_1 \land M_2 \land \ldots \land M_l) \lor (N_1 \land N_2 \land \ldots \land N_m)
\]

with the \( L_h, M_i \) and \( N_j \) positive or negated atomic formulas (literals), with no restrictions on the quantification of variables whatsoever.

CLASS II is a subset of CLASS I. The difference is that postcondition formulas are made determinate (contain no disjunctive information) and constraints are limited to a much more restricted form. The following theorem holds for specifications of this type:

**Proposition 3** For formulas of CLASS II the minimal subset and minimal cardinality semantics coincide.

**Proof**

Assume we have a set of formulas

\[
\phi^i \rightarrow [a]((\land P^i_q) \land (((\land C^1_s) \lor (\land D^1_t)) \land \ldots \land ((\land C^k_s) \lor (\land D^k_t)))) \\
\langle a \rangle \top \rightarrow \chi \land (((\land C^1_s) \lor (\land D^1_t)) \land \ldots \land ((\land C^k_s) \lor (\land D^k_t))) \\
(\land C^1_s) \lor (\land D^1_t)
\]

And under preservation of minimality properties to

\[
\phi^i \rightarrow [a]((\land P^i_q) \land (((\land C^1_s) \lor (\land D^1_t)) \land \ldots \land ((\land C^k_s) \lor (\land D^k_t)))) \\
\langle a \rangle \top \rightarrow \chi \land (((\land C^1_s) \lor (\land D^1_t)) \land \ldots \land ((\land C^k_s) \lor (\land D^k_t)))
\]

Now focus on transitions \( a \) from a state \( s \) to a state \( s' \) in the model. In \( s \) each \((\land C^k_s) \lor (\land D^k_t)\) is satisfied. Minimal interpretation strives to make the \( s' \)-interpretation of the literals in this formula equal to their \( s \)-interpretation. There is one formula component that possibly forbids this: \( \land P^i_q \). If this formula part forces a different \( s' \)-interpretation of literals in \((\land C^k_s) \lor (\land D^k_t)\), it must contradict either \((\land C^k_s)\) or \((\land D^k_t)\). It is actually rather easy to check whether such a contradiction is present; just check for opposite literals. It can not contradict both parts, because then, from the beginning, there can not have been a transition at all. But then, under minimal interpretation, we can just eliminate the part of \((\land C^k_s) \lor (\land D^k_t)\) that contradicts \( \land P^i_q \) and conjoin the other part to the postcondition. If both parts of \((\land C^k_s) \lor (\land D^k_t)\) do not contradict \( \land P^i_q \), just eliminate \((\land C^k_s) \lor (\land D^k_t)\) completely from the postcondition, because minimal interpretation assures that this formula holds in \( s' \). This means we get a set of formulas
with the same minimality properties as the formulas we started with. Now in the MS-model all s'-interpretations of function and predicate symbols that do not appear in the \(Q^i_q\) obviously must be equal to their s-interpretation. The s'-interpretation of function and predicate symbols that appear in the \(Q^i_q\) and that do not have existentially quantified variables as an argument, or that do not depend on some variable at all, must be equal to their s-interpretation for minimality reasons (Note that equal interpretation of the literals \(Q^i_q\) in both s and s' does not assure equal interpretation of predicate and function symbols in these literals. this is because a different interpretation of a function symbol \(f\) can be compensated by a different interpretation of a predicate symbol \(P\) in such a way that the interpretation of the literal \(P(f(t))\) stays the same. Only a minimal interpretation can assure that the interpretation of predicate and function symbols in these literals is equal.) Function and predicate symbols that do appear in the \(Q^i_q\) and that depend on a variable that is existentially quantified, are interpreted in such a way that there is precisely one assignment of the variable to a domain element, that causes the literal to hold. From a model with more than one existing assignment that causes the literal to hold, we could easily construct a more minimal one under the subset ordering by adapting the interpretation of function and predicate symbols in such a way that only one of these assignments would cause the literal to hold. But then the model \(M\) is also minimal under the cardinality ordering. We already have proven that the MC-models are a subset of the MS-models. Now, for other MS-minimal models of the formulas the assignment of existentially quantified variables is possibly to other domain elements, but for every model there is precisely one such an assignment. Each assignment brings about the same number of changes to interpretations of function and predicate symbols and so for each of these models also the number of changes is equal.

Although the postcondition in the postcondition formulas of specifications of CLASS II are completely determinate, minimal models for these specifications are not deterministic. We will first define deterministic models and then give an example why minimal models for this class are not deterministic.

**Definition 15** A structure \(S = (D, S, I_{AA})\) is deterministic if for each state \(s \in S\) and for each \(a\) there is maximally one state \(s'\) such that \((s, s') \in I_{AA}(a)\).

**Example 2** Consider the specification:

\[
[\text{Shoot } \_ \_ \text{gun}] \exists \text{something, Hit(something)}
\]

Obviously if the domain for the Predicate Hit has several "objects", there are many possible transitions possible from a certain state. More of these transitions can be present in minimal models (the minimality criteria do not minimize nondeterminism!) This reflects the fact that there are many things that are possibly hit. In the following we will see that the existential quantification is the only source for non-determinism in this specification.
The last class we consider here precisely describes the class of formulas for which minimal models are deterministic. This is an important class. Since the minimal (intended) models are deterministic for these formulas, the only intention a specifier can have when providing formulas of this form, is to specify a deterministic system. Interpreting the formulas under a non-minimal semantics would allow for non-deterministic interpretation of the formulas, which is not what is intended.

**Definition 16** the description of CLASS III formulas:

\[ \Phi_{\text{post}} : \phi \rightarrow [a] \bigwedge L_1 \land L_2 \land \ldots \land L_k \]

\[ \Phi_{\text{guard}} : (a) \top \rightarrow \chi \]

\[ \Phi_{\text{IC}} : (M_1 \land M_2 \land \ldots \land M_l) \lor (N_1 \land N_2 \land \ldots \land N_m) \]

with the \( L_k, M_i \) and \( N_j \) positive or negated atomic formulas (literals), and with the restriction on the quantification that variables in \( L_k, M_i \) and \( N_j \) are all universally quantified.

This class is a subset of both CLASS I and CLASS II. The difference with CLASS 2 is that we allow only universal quantification. We prove the property that for formulas of CLASS III, both MS-models and MC-models are deterministic.

**Proposition 4** For a specification Spec build with formulas from CLASS III, MS-models and MC-models are deterministic.

**Proof** Under preservation of minimal models we transform (as in the former proof) a set of formulas of CLASS III to

\[ \phi^* \rightarrow [a](\bigwedge Q_q^i) \]

\[ (a) \top \rightarrow \chi' \]

Now assume that there is an MS-model \( M \) of this set that is not deterministic. Then we have that \( \exists a \in AA, \exists s \in S, \exists s' \in S, \exists s'' \in S, (s, s') \in I_{AA}(a) \land (s, s'') \in I_{AA}(a) \) Now let \( \{Q_q^i\} \) be the set of all atoms that appear in the heads of determinate postcondition formulas, for which \( S, s, I_V \models \phi^* \). Because \( S \) is a model and we have only universal quantifications in the heads of formulas, the atoms in \( \{Q_q^i\} \) have the same valuation in both \( s'' \) and all \( s' \). This means that \( s'' \) differs from \( s \) in the same atoms among \( \{Q_q^i\} \), as \( s' \) does. This means that \( s'' \) and \( s' \) differ in at least one atom, say \( A_d \) not among the atoms in \( \{Q_q^i\} \). Without loss of generality we assume that \( s' \) is the state where \( A_d \) has an interpretation different from its interpretation in \( s \) and \( s'' \) is the state where \( A_d \) has an interpretation equal to its interpretation in \( s \). Now we can construct the model \( S' \) that is equal to \( S \) except for that the transition \( (s, s') \) is left out. Obviously \( S' \) is below \( S \) in the \( \subseteq_{ms} \)-ordering, because the transition \( s, s' \) that changes more then \( s, s'' \) is left out. But this contradicts that \( S \) is an MS-model.

The proof for MC-models is analogous.

The former property does not hold if we allow Horn-clauses as constraints, as follows from the next example.
Example 3

\[ C(j) \rightarrow [a]A(j) \]
\[ \langle a \rangle \top \rightarrow \neg A(j) \land \neg B(j) \land C(j) \]
\[ A(j) \land C(j) \rightarrow B(j) \]

The state \{A(j), B(j), C(j)\} has two possible follow-up states \{A(j), B(j), C(j)\} and \{A(j), B(j), C(j)\}. We could of course add the distinction between base and other predicates to work around this.

3.2 The qualification problem in DL specifications

The interpretation of guard formulas is that action \( a \) can only take place when \( \chi_j \) is true. Thus \( \chi_j \) is a necessary precondition for the possible occurrence of \( a \). So guard formulas control the possible occurrence of actions. The problem associated with the possible occurrence of actions is the qualification problem. This problem states that it is not possible to foresee all necessary preconditions for the success of an action [9]. This means that a specifier actually is never able to give a sufficient precondition for an action. Of course he may want to give a sufficient precondition anyway, but he then must face the fact that he may end up with an inconsistent specification because the action he gave a sufficient precondition for may violate constraints or may have contradictory postcondition axioms associated to it. A possible solution to this problem is to weaken sufficient preconditions by specifying them as defaults [2] [14]. We take the perspective that a specifier should only be allowed to give necessary preconditions (guards) and that the assumption that actions occur unless this contradicts guards (or static constraints, or conflicting postcondition axioms) is somehow implemented in the semantics. The standard PDL-semantics does not provide this. It is easily seen that there are interpretations of action specification axioms in which actions do not occur even if the guards are true. In particular the structure where the accessibility relation between states is completely empty always satisfies a set of action specification axioms.

Guards are necessary preconditions, so they do not 'force' transitions. In the following we define an interpretation that tends to interpret the guards specified for an action \( a \) as sufficient, provided that the possible occurrence of \( a \) is compatible with the constraints and postconditions in the postcondition axioms. We accomplish this by formalizing the qualification assumption in the notion of maximal reachability.

Definition 17 Given a signature \( \Sigma = (P, F, AA) \) and two structures \( S' = (D, S', \mathcal{I}_{AA}) \) and \( S = (D, S, \mathcal{I}_{AA}) \)

\[ S' \sqsubseteq_{mr} S \text{ iff } \]
\[ S' \subseteq S \]
\[ \text{and} \]
\[ \text{for all } a \in AA, \mathcal{I}_{AA}^{'}(a) \subseteq \mathcal{I}_{AA}(a) \]

\( \sqsubseteq_{mr} \) is a partial order on structures, because \( \sqsubseteq_{mr} \) can easily seen to be transitive, reflexive and anti-symmetric. The ordering just 'prefers' as much states and transitions over them as possible, thus implementing the notion of 'maximal reachability'. In Figure 2 this is reflected by the fact that all transitions and states in lower models are also in the model at the top.
Figure 2: Comparing models on maximal reachability

**Proposition 5** Let $\Phi$ be a set of formulas for which there is a model. Then there is a $\sqsubseteq_{mr}$-maximal model.

**Proof** It is sufficient to prove that the $\sqsubseteq_{mr}$-ordering on models forms a complete lattice. For this we have to prove that each subset of the models satisfying $\Phi$ has an upperbound and underbound that is itself a model. We can easily define an upperbound and underbound. Let $(D, S, T_{AA}^i)$ represent a set of models for $\Phi$. Then we define the model that is an upperbound by $S = (D, S, T_{AA})$ with $S = \bigcup S^i$ and $T_{AA}$ such that $\forall a \in AA, T_{AA}(a) = \bigcup T_{AA}^i(a)$ and the model that is an underbound as $S' = (D, S', T_{AA}')$ with $S' = \bigcap S'$ and $T_{AA}'$ such that $\forall a \in AA, T_{AA}'(a) = \bigcap T_{AA}^i(a)$. It is not difficult to see that these models will satisfy $\Phi$.

Preferring $\sqsubseteq_{mr}$-maximal structures leads to yet another interpretation of formulas. Again, this interpretation results in non-monotonic entailment. An example of this is provided by the set $\Phi = \{[a]A(p)\}$. Under interpretation over $\sqsubseteq_{mr}$-maximal structures, $\langle a \rangle A(p)$ is entailed by $\Phi$. However, this is no longer true for $\Phi \cup [a] \neg A(p)$.

### 3.3 The interpretation of static constraints

In the foregoing we did not give special attention to the interpretation of static constraints; they were just interpreted as modality free dynamic logic formulas. A standard logic interpretation of constraints gives rise to what is called ”ramifications”. To see this we repeat one of our former examples.

**Example 4** Consider the specification:

\[
\begin{align*}
[dial\ number](get\ connection \lor get\ busy\ tone)) \\
get(connection) &\rightarrow costs\ money \\
\langle dial\ number \rangle \top &\rightarrow \neg costs\ money \land \neg get\ connection \land \neg get\ busy\ tone
\end{align*}
\]

When we discussed this example we interpreted the constraint $get(connection) \rightarrow costs(money)$ as giving rise to the ramification $costs(money)$ in cases that $get(connection)$ was made true. We call this kind of interpretation of constraints the **ramification semantics** for constraints. We also want to consider another interpretation: the constraint semantics.
Definition 18: Given a specification \((\Sigma, \Phi_{\text{trans}})\) with \(\Phi_{\text{trans}} = \Phi_{\text{post}} \cup \Phi_{\text{guard}} \cup \Phi_{\text{IC}}\), the constraint semantics is defined by the \(\sqsubseteq_{\text{mr}}\)-maximal model of the set MSIC/MCIC, with MSIC/MCIC the set of MS/MC-models of \(\Phi_{\text{post}} \cup \Phi_{\text{guard}}\) that in addition satisfy \(\Phi_{\text{IC}}\).

The constraint semantics does not give rise to derived effects (states), it merely 'cuts out' access to states in which the constraints are not satisfied. (See also [15]) We accomplish this by first looking at minimal models of specifications without taking the constraints into account. Transitions in these minimal models possibly lead to states that do not comply with constraints. By applying the criterion that models also have to comply with static constraints in a second step, we lose these transitions all together.

Constraint and ramification semantics can of course be combined by splitting the sets of non-modal axioms into two sets: one set to be interpreted by the constraint semantics and one by the ramification semantics. The formulas to be interpreted by the ramification semantics are called derivation rules. Spruit et al. [17] work this out for Propositional Dynamic Database Logic (PDDL) with Horn Clauses as derivation rules.

3.4 Min-Max-models

For the interpretation of specifications under both frame and qualification assumption we have to combine orderings. We do this by applying them one after the other. What, if any, is the right (intuitive) order in which to do this? The answer can be found by looking at what the orderings are supposed to represent. The minimal change orderings concern the effect of actions independent from whether they occur or not. Maximal reachability deals with possible occurrence. It is intuitive then to apply minimal change first. The minimal change ordering determines what action we actually mean by 'determining' their effects. After that we can 'talk' about the possible occurrence of actions. This motivates the following definition.

Definition 19: Given a specification \((\Sigma, \Phi)\), a Min-Max-model is defined as a \(\sqsubseteq_{\text{mr}}\)-maximal element of the set of MS/MC-models of \(\Phi\).

The next example shows that defining this in reverse order does not provide the semantics we are looking for.

We take a signature with \(\mathcal{P} = \{A, B\}\), and \(\mathcal{A} = \{a\}\), and the following set of formulas: \(\Phi = \{[a]A, \langle a \rangle T \rightarrow \neg A \land \neg B\}\).

The left picture shows the \(\sqsubseteq_{\text{mr}}\)-maximal structure that satisfies \(\Phi\). This structure is not MS-satisfying. This means it is not useful to apply maximality and minimality in this order. There are only two MS-models of the example formulas, the structure with no access to other states at all and the one shown in the right picture. Clearly the one in the picture is \(\sqsubseteq_{\text{mr}}\)-maximal. We show in 3.5 that this construction also works in case there are ramifications.

The MS-models of a set of formulas \(\Phi\) (we mean general formulas here) form a partially ordered set under the \(\sqsubseteq_{\text{mr}}\)-ordering. This set is not a lattice, as is shown by the following
example. Take the formula $\neg((a)A(c) \land (a)B(c))$ and the two MS-satisfying structures $\mathcal{S}$ with $I_{AA} = \{(a, \{\}, \{(A, \{d\})\})\}$ and $\mathcal{S}'$ with $I_{AA}' = \{(a, \{\}, \{(B, \{d\})\})\}$. There is no MS-satisfying structure that is an upper bound for both structures $\mathcal{S}$ and $\mathcal{S}'$. The structure $\mathcal{S}''$ with $I_{AA}'' = \{(a, \{\}, \{(A, \{d\})\}), (a, \{\}, \{(A, \{d\})\})\}$ is an upper bound under the $\subseteq_{\text{mr}}$-ordering, but is not MS-satisfying (not even satisfying). So in general there can be more Min-Max-models of a set of formulas $\Phi$. However, for specification formulas of the restricted form we defined, we can prove that the MS-models do form a complete lattice. Therefore, the following proposition holds.

**Proposition 6** Each specification $\Phi_{\text{trans}}$ has a unique Min-Max-model if it has a model.

**Proof** We must prove that the minimal models of a specification $\Phi_{\text{trans}}$ form a complete lattice under the $\subseteq_{\text{mr}}$-ordering. To prove this we define the lub of a set of MS-satisfying (or MC-satisfying) structures $(D, S', I_{AA})$ as $S = (D, S, I_{AA})$ with $S = \cup S'$ and $I_{AA}$ such that $\forall a \in AA, I_{AA}(a) = \cup I_{AA}(a)$ and the glb as $S' = (D, S', I_{AA}')$ with $S' = \cap S'$ and $I_{AA}'$ such that $\forall a \in AA, I_{AA}'(a) = \cap I_{AA}'(a)$. It is not difficult to see that both lub and glb are minimal models of $\Phi_{\text{trans}}$.

In Min-Max-models, the maximization in most cases causes severe non-determinism. To see this we once more look at the following example.

**Example 5** Consider the specification:

$$[\text{Shoot}_a \text{gun}] \exists \text{something}, \text{Hit(something)}$$

If the variable "something" ranges over an infinite domain of objects, the action $\text{Shoot}_a \text{gun}$ from a given state branches to an infinite number of other states.

For deterministic structures (models) a Min-Max model can be defined with the help of one, much simpler ordering. For deterministic models we define the $\subseteq_{\text{mm}}$-ordering.

**Definition 20** Given a signature $\Sigma = (P, F, AA)$ and two structures $S = (D, S, I_{AA})$ and $S' = (D, S', I_{AA}')$,

$$S' \subseteq_{\text{mm}} S \text{ iff }$$

$$S' \subseteq S$$

and

$$\forall a \in AA, \forall s \in S, (\exists s'' \in S, (s, s'') \in I_{AA}(a) \Rightarrow \exists s' \in S', ((s, s') \in I_{AA}'(a)$$

$$\land \text{Diff}(s, s') \subseteq \text{Diff}(s, s''))$$

It is not difficult to see that the above (partial) ordering combines the notion of maximal reachability and minimal change.

**Proposition 7** For specifications of CLASS III, the Min-Max-model coincides with the model that is minimal under the $\subseteq_{\text{mm}}$-ordering.

The proof of this is ommitted.
3.5 Example

As an example specification we take:

\[
\text{Gun} \text{loaded}\_\text{sharp} \rightarrow [\text{fire}\_\text{gun}] \text{Gun} \text{blown}\_\text{up} \lor \text{Bullet} \text{emitted}
\]

\[
\text{Gun} \text{loaded}\_\text{blank} \rightarrow [\text{fire}\_\text{gun}] \text{Gun} \text{blown}\_\text{up} \lor \text{Air} \text{and dust} \text{emitted}
\]

\[
\rightarrow [\text{fire}\_\text{gun}] \text{Big noise}
\]

\[
(fire\_gun) \top \rightarrow \neg \text{Gun} \text{blown}\_\text{up}
\]

\[
(fire\_gun) \top \rightarrow \text{Gun} \text{loaded}\_\text{sharp} \lor \text{Gun} \text{loaded}\_\text{blank}
\]

\[
\text{Bullet} \text{emitted} \rightarrow \text{Somebody is hit}
\]

\[
\neg (\text{Gun} \text{loaded}\_\text{sharp} \land \text{Gun} \text{loaded}\_\text{blank})
\]

For the interpretation of this (propositional) specification we have to choose between the several semantics we defined. First we look at the interpretation of constraints. There is only one action, fire\_gun. It is a nondeterministic action because some postcondition formulas contain disjunctions. One of the alternatives represented by the disjunction is Bullet\emitted. This atom is also present in one of the constraints. If we interpret this constraint under the "constraint semantics“, the condition "Somebody is hit" can actually never change to true as a result of the action fire\_gun. It can only be true after the action fire\_gun if it was already true in the state before fire\_gun took place. This means the most intuitive interpretation for this constraint is the ramification semantics. We think the ramification semantics is usually the best interpretation for constraints.

Now we look at which minimality criterion to apply. We have to choose between the minimal subset and the minimal cardinality criterion. The choice is not too difficult if we look at the postcondition Gun\blown\_up \lor Bullet\emitted. Given the choice that we interpret constraints under the ramification semantics, the truth of Bullet\emitted will imply the truth of Somebody is hit in resulting states. This means that the action fire\_gun usually 'has the choice’ between making one atom true (Gun\blown\_up) or two (Bullet\emitted and Somebody is hit). The minimal cardinality semantics will choose the first alternative which is really counterintuitive. The minimal subset semantics just makes no choice; both successor states are possible. This means that the minimal subset semantics is a more non-deterministic interpretation. We think the minimal subset semantics is always the more intuitive one.

We now show how the action fire\_gun can be qualified. The maximality criterion assures that in the Min-max-model of the specification, fire\_gun-transitions are actually there. But of course we can constrain the occurrence of transitions by adding extra formulas to the specification. Adding \{Gun\blown\_up\} or \{-Big noise\} would leave us with no transitions at all. Adding \{-Gun\loaded\_sharp\} would result in transitions that never make Somebody is hit true.

Finally we name some properties that are true in the Min-max-model (with the minimal subset / ramification criteria) of the specification and that are formulated using action connectives. The first one is:

\[
\text{Gun} \text{loaded}\_\text{blank} \rightarrow [\text{fire}\_\text{gun}] \text{Gun} \text{loaded}\_\text{blank}
\]

It says that if the gun was loaded with a blank it will stay loaded with blanks after a possibly infinite amount times of shooting. A second one is:

\[
[\text{Gun} \text{loaded}\_\text{sharp} \land \text{fire}\_\text{gun}] (\neg \text{Gun} \text{blown}\_\text{up} \rightarrow \text{Somebody is hit}).
\]

It says that if the gun is fired in a situation where it is loaded with a bullet, and as the result of it the gun will not blow up, then somebody is hit.
4 Comparison with other work

The frame problem and the associated problems of qualification and ramification are common themes in the AI literature on knowledge representation and reasoning about action and change [5].

Reiters approach to the frame problem [16] is to rewrite 'effect axioms' to 'successor state axioms'. Effect axioms say for each action which predicates change their value if the action is performed. Successor state axioms say for each predicate which actions change when performed. Reiters approach consists mainly of a change of focus from specific actions to specific predicates and a form of completion on them.

A difference between Reiters language and ours seems to be that in his language quantification over states and actions is possible. However, in our language quantification over states reachable by actions is done by modal operators. Unrestricted quantification over states would have to be done by adding the usual box and diamond of modal logic. Quantification over actions could be added in our language. Since the number of specified actions is usually finite this is no severe restriction.

A more serious distinction between Reiter's specification formulas and ours concerns preconditions. Reiter's action preconditions are:

\[ \text{Poss}(a, s) \leftarrow \chi_{\text{suf}}(x_i, s) \]

while we have:

\[ \langle a \rangle \top \rightarrow \chi_{\text{nee}}(x_i) \]

The difference is that in Reiter's formula \( \chi \) is a sufficient condition for the possibility of action \( a \), while in our formula it is a necessary condition. So Reiter can actually never forbid actions from being possible in certain situations, he can only force them to be possible. We chose not to specify sufficient conditions, because this might cause problems in the presence of static constraints that possibly forbid transitions to go to certain states. This is exactly the problem Reiter e.a. run into [13].

Other formulas Reiter uses do correspond to ours. Reiters positive effect axioms have the form:

\[ \text{Poss}(a, s) \land \phi(y_i, s) \rightarrow R(y_i, \text{do}(a, s)) \]

Quantification is over actions \( a \) and states \( s \). These axioms are supposed to sum up all actions that make the predicate symbol \( R \) true (by choosing the right sufficient effect precondition \( \phi(y_i, s) \)). In our language this would correspond to a final set of formulas (one for each action) of the form:

\[ \phi(y_i) \rightarrow [a]R(y_i) \]

Note that we do not need a part \( \langle a \rangle \top \) in this formula, while Reiter needs a \( \text{Poss}(a, s) \). This is because the formula part \( [a]R(y_i) \) already incorporates that \( R(y_i) \) is evaluated only if \( a \) is possible. Reiters negative effect axioms have the form:

\[ \text{Poss}(a, s) \land \psi(y_i, s) \rightarrow \neg R(y_i, \text{do}(a, s)) \]

These can be translated into a finite set of our:
\[ \psi(y_i) \rightarrow [a] \neg R(y_i) \]

Now Reiters solution to the frame problem consists in defining a completion of the above formulas by replacing the positive and negative effect axioms by successor state axioms of the following form:

\[ \text{Poss}(a, s) \rightarrow (R(y_i, do(a, s)) \leftrightarrow (\phi_{\text{suf}}(y_i, s) \lor (R(y_i, s) \land \neg \psi_{\text{suf}}(y_i)))) \]

We will call this axiom a positive successor state axiom, because it is based on intuitions about changes that makes \( R \) to hold (a change from \( \neg R(y_i, s) \) to \( R(y_i, do(a, s)) \)). Reformulated in our language a positive successor state axiom corresponds to a finite set of formulas of the form:

\[ [a] R(y_i) \leftrightarrow \phi(y_i, s) \lor (R(y_i) \land \neg \psi(y_i, s)) \]

note that here we also do not need a part \( \langle a \rangle \top \). Because Reiter can quantify over actions he can do with one formula, in our language we get a finite set. The intuitive meaning of this axiom is that predicate \( R \) only holds in the state reached after performing \( a \) iff in the current state the sufficient effect precondition for \( R \) holds or if in the current state \( R \) already holds and the sufficient effect precondition for \( \neg R \) does not hold.

This approach has several limitations. First of all it should be mentioned that the completion can only be performed on postcondition formulas that are determinate (nu disjunctive postconditions). This means that this approach does not deal with nondeterministicly specified effects. Second, the completion is not possible in the presence of static constraints that possibly contradict the effect. Third, the completion is not ‘complete’, because the successor state axiom does not forbid changes from ‘\( R \) holds in state \( s \)’ to ‘\( \neg R \) holds in \( do(a, s) \)’. This is because the fact that the sufficient precondition \( \psi_{\text{suf}} \) does not hold, does not forbid \( \neg R \) to hold in \( do(a, s) \). This is clearly not intended. Another way to see this is that positive and negative effect axioms are not symmetrically dealt with. A symmetrical approach would also give us negative successor state axioms. We get these by focusing on the negative effect axiom and applying the same intuitions that were used for the construction of the positive successor state axioms.

\[ \text{Poss}(a, s) \rightarrow (\neg R(y_i, do(a, s)) \leftrightarrow (\psi(y_i, s) \lor (\neg R(y_i, s) \land \neg \phi(y_i)))) \]

It is not difficult to see that this formula is not logically equivalent with the former one, which means that the formula contains information not present in the positive successor state axiom. For an equal handling of changes of \( R \) from true to false and the other way around, this really should have been the case.

The obvious solution for both incompleteness and asymmetry would be to add both positive and negative successor state axioms. But in that case the part \( (\neg R(y_i) \land \neg \phi(y_i, s)) \) is superfluous, because the axioms are logically equivalent to the following pair of formulas:

\[
\begin{align*}
\text{Poss}(a, s) & \rightarrow (R(y_i, do(a, s)) \leftrightarrow (\phi(y_i, s))) \\
\text{Poss}(a, s) & \rightarrow (\neg R(y_i, do(a, s)) \leftrightarrow (\psi(y_i, s)))
\end{align*}
\]

We think the intuitions that Reiter tries to catch are best represented by the completion represented by these formulas. It is actually a very simple form of completion since the original formulas are logically equivalent with:
Our semantics provides a symmetric handling of positive and negative changes.

Borgida et al. [1] take the perspective of the designer of specification languages and discuss ways to state that “nothing else changes” by syntactic as well as semantic means. Our work can be regarded as introducing a richer semantics for the specification language to capture this.

Giunchiglia, Kartha and Lifschitz introduce the action language $\mathcal{AR}$ [10], which is an extension of the language $\mathcal{A}$ [8], introduced by Gelfond and Lifschitz. $\mathcal{AR}$ differs from $\mathcal{A}$ in that it also deals with ramifications.

There is a straightforward translation of most elements of the language $\mathcal{AR}$ into elements of (the propositional version of) our language. The translation is:

- $A$ causes $C$ if $P$ to $P \rightarrow [A]C$ (conditional postcondition)
- always $L \supset W$ to $L \rightarrow W$ (static constraint)
- impossible $G$ if $L$ to $\langle G \rangle \top \rightarrow \neg L$ (necessary precondition)
- $J$ initiates $L$ to $[J]L, \langle J \rangle \top \rightarrow \neg L$ (combination of a conditional postcondition and a necessary precondition)

Under this translation their (minimal) interpretation function $Res_D$ perfectly matches our Min-Max-models (ramification semantics for the constraints, and because they only consider determinate postconditions there is no difference between the minimal subset or minimal cardinality criterion). The difference between their language and ours is that they can express that an effect possibly occurs ($A$ possibly changes $F$ if $P$) which is not translatable into our language. On the other hand we can express that the effect of an action is a choice between two or more alterations ($[a][A \vee B]$), which is not translatable in their language. Interesting is that both constructs are claimed to represent a non-deterministic aspect of actions.

Winslett’s work on database update semantics [18] focuses on a model-oriented approach to updates, de-emphasizing the relation between the specification and the models. Instead, we base our semantics on the declarative semantics of a specification in DL, which allows us to reason about updates in the same language.

Brass and Lippek [2] [14] study action specification with the help of defaults. They also define orderings over modal interpretations. Frame and other assumptions are represented by formulas interpreted as defaults. This still puts the responsibility on the specifier to provide such formulas, which is not always desirable. Furthermore their models represent ’action traces’ and do not allow for non-determinism.

When restricting ourselves to finite sets of atomic actions and atomic formulas, some of the semantics presented here for explicit effect axioms are a convenient starting point for operationalization. In this case, the state space of the specified systems is finite, and we are able to construct it explicitly. This opens the door to the application of model checking techniques (like in [6]) on this state space to verify system properties.

5 Conclusions and future work

In this paper we defined several alternative semantics for action specification formulas under the frame assumption and the qualification assumption. Choosing one of them
removes the ambiguity introduced by implicit frame assumptions and implicit qualification assumptions. Procedures such as adding frame axioms or applying completion that are usually necessary to reveal the intended meaning of a specification, can be compared with the semantics we defined. Furthermore, we plan to compare existing procedures for scenario generation and reachability analysis with our semantics. We also plan to investigate ways to generate the intended model (Min-Max-model) from a given specification. For that we will have to limit ourselves to finite domains. We will have to find a suitable representation for models and an algorithm that connects this representation to specifications. Also the prospect of code generation will be investigated.

References


