Modal Logics for AI Planning[*]

J. Dix, J. Posegga, P.H. Schmitt

Universität Karlsruhe,
Institut für Logik, Komplexität und Deduktionssysteme
Postfach 6980, 7500 Karlsruhe, FRG
email: {dix|posegga|pschmitt}@germany.csnet

November 13, 1989

Abstract

This paper presents a survey on current research in modal and dynamic logic in the theory of AI planning techniques together with a first assessment of the role these concepts and techniques may play in the overall process of automated plan formation.

First, the use of modal logics for modelling planning problems is motivated. After showing the computational problems –besides the well known frame problem– of the situation calculus by an example, it is argued that a modal framework for modelling planning problems offers several advantages compared with classical planning approaches. Then, the basic ideas of modal logics are presented, and two existing approaches to planning using a modal framework are reviewed. At the end, an outlook towards efficient implementations of modal logics is given.

1 The Very Idea

The first attempts to use formal logic for automated plan generation, like Green's situation calculus, or Kowalski's extensions [Kow79] were completely based on first order predicate calculus and plans were extracted directly from automated deductions. The major advantage of these approaches is the use of a uniform framework with clear semantics. However, straightforward implementations of this approach did not achieve the efficiency to be useful for practical applications. An applicable implementation of a logic approach, as in STRIPS, had to employ more procedurally oriented inference techniques making semantic considerations much more difficult [Lif87].

This situation has not changed significantly up to now. Although research in deduction systems has led to considerable advances, the problem to implement a deduction system for first-order predicate calculus, that can generally solve "real-world" problems has still to be regarded as unsolved. However, classical first-order predicate calculus is just one of many logics that are known and have been investigated for some decades. It must be regarded as a general-purpose tool that is more or less well suited to different applications. Some other candidates in formal logic seem to offer more advantages while still preserving semantic clarity.

The basic motivation to use modal logic for planning tasks comes from the observation that the so-called Kripke structures (see [HC68] for details), which constitute the model theoretic semantics for modal logics exactly parallels the set-up in planning tasks: Kripke structures

[*]A slightly shorter version of this paper appeared in *Proc. First International Conference on Expert Planning Systems*, Brighton, UK, 1990.

are built up by a collection of "worlds" and an accessibility relation between these "worlds". Worlds may be interpreted as descriptions of all relevant parameters and facts of a particular situation during a plan formation process, while a world $\mathcal{W}\prime$ is accessible form world $\mathcal{W}$, if there is a single action or a sequence of actions that transforms the state of affairs described by $\mathcal{W}$ into the state described by $\mathcal{W}\prime$. The various properties of the accessibility relation (reflexivity, transitivity,...) define the actual modal logic that is used. Compared with situation-calculus-like axiomatizations in classical logic, these worlds in a modal framework correspond to the state parameter in formulas. So, modal logic nicely structures the domain.

But not only the state space may be modelled and structured as the Kripke universe of possible worlds and its accessibility relation, also the structure of partial plans itself fits into this framework [Cha87]. Viewed from some already obtained partial plan, where the sequencing of sub tasks is not completely specified, the accessibility relation points to those refinements that are still possible on the basis of the constraints already accumulated.

Dynamic logics can be regarded as an extension of 'classical' modal logics replacing one fixed accessibility relation on worlds by many user definable accessibility relations. Each available action in a planning environment could define a different modal logic by providing its own accessibility relation with certain properties. Actions like 'put' or 'move' for instance, can usually be undone, so an accessibility relation $\mathcal{R}_{move}$ will be symmetric. Actions like painting, on the other hand, are hard to undo, which implies asymmetry for $\mathcal{R}_{paint}$, but also transitivity, which will not be given for $\mathcal{R}_{move}$.

Besides its formal rigor the proposed logics offer some advantages compared with classical logic:

- Adequacy for the knowledge to be expressed.
- A rich body of theoretical results and techniques.
- Techniques for proving formulas which are well suited for the exploitation of heuristics (e.g. tableau systems as described in [Sch87]).
- Features for modelling phenomena like the generating or deleting of objects by Kripke frames with variant domains.

The papers already published on this topic and our experience with the use of dynamic logic in program verification [HRS88] supports our belief that modal logic and its extension to dynamic logic offers a framework well suited to formally specify planning tasks, intermediate steps and the final plan.

Furthermore recent advances in the theory and design of modal logic inference engines, comparable to the introduction of the resolution rule in place of the level saturation technique for classical theorem provers, give rise to the hope of increased practical applicability [Ohl88]. There is actually evidence that planning problems modelled by means of possible words can be handled more efficiently than in first order predicate calculus in a situation-calculus like style [Gin87].

The following section will review two classical planning paradigms, namely situation calculus and STRIPS, and show their problems which are typical for current approaches to planning. section 3 gives an idea how modal logics work, and section 4 shows how they can be employed for planning. Techniques for efficient implementation of modal logics are treated in section 5.

2 Classical Planning Paradigms

2.1 Situation Calculus

The situation calculus goes back to McCarthy and Hayes [MH69]. The basic idea is to use standard first order logic to express and solve a planning problem, whereas the state of the world (a so-called *situation*) is denoted by values of terms. Each predicate has an additional argument that denotes the situation in which it is valid. An action with n arguments is represented by a $n+1$-ary function, where the $n+1^{th}$ argument denotes the state in which the action is executed. The value of the function is again a state[1].

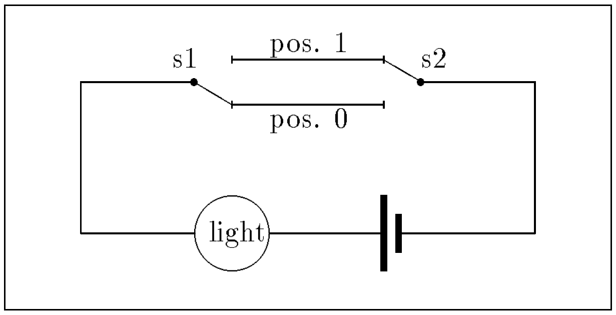

Figure 1: Change-over switches

Suppose we want to model two change-over switches that can be used to switch a light bulb on and off (see figure 1). An axiomatization for this rather trivial example is shown in figure 2 (Uppercase letters indicate variables).

```
A1)  unequal(0,1).
A2)  unequal(1,0).
A3)  unequal(s2,s1).
A4)  unequal(s1,s2).
A5) pos(s1,0,i).
A6) pos(s2,1,i).
A7) pos(s1,X,Z) & pos(s2,X,Z) --> light(on,Z).
A8) pos(s1,X,Z) & pos(s2,Y,Z) & unequal(X,Y) --> light(off,Z).
A9) pos(X,0,Z) --> pos(X,1,sw(X,Z)).
A10) pos(X,1,Z) --> pos(X,0,sw(X,Z)).
A11) pos(X,Y,Z) & unequal(X,X1) --> pos(X,Y,sw(X1,Z)).
```

Figure 2: Axiomatization of Change-over switches

The axioms translate as follows: `A1` to `A4` define the appearing constants to be not equal. `A5` and `A6` say that switch `s1` is in position `0` at situation `i`, and `s2` is in position `1` at situation `i`. Axiom `A7` says: if the two switches `s1` and `s2` are both in position `X` in some situation `Z`, then

[1]Other variants of this calculus use a special predicate 'T' to denote that a term describing some property in some state is valid, and a function 'result' to map a function denoting an action and a state to some new state. This simplifies the definition of frame axioms, but the difference is irrelevant for our purposes.

the light is on in this situation. Axiom A8 defines the light two be off if they are in different positions. Axioms A9 and A10 describe what it means to switch some switch X in some situation Z. The frame axiom A11 says that switching a switch does not affect any other switch.

If we are interested in a plan for switching the light on, we have to find a proof of the formula light(on,S). The resulting substitution for the variable S (in this case for instance sw(s1,i)) would represent the plan we are looking for.

This approach has the nice property of having well-defined semantics, which facilitates proving completeness and soundness. Additionally, it provides the full expressivness and flexibility of first-order logic. Unfortunately, it is hopelessly inefficient. This will be shown in the following.

It is a key issue in planning to determine the state of the world after executing some (maybe only partially known) sequence of actions. Assume, for instance, we are interested in the status of the light after switching some arbitrary switches two times. An instantiation for X the theorem light(X,sw(X1,(sw(X2,i)))) will answer the question. The following shows a refutation for the above axiomatization (A2 - A11) and the negated theorem (A12). The clauses appear in conjunctive normal form, whereas negation is denoted by "-" and disjunction by '|'. Axiom A7 and A1 do not appear, because they are not needed for the proof. $ans is a so-called answer literal that is used to collect the substitution we are interested in. The proof shown in figure 3 was done with the otter [McC88] theorem prover.

```
---------------- PROOF ----------------
2 unequal(1,0).
3 unequal(s2,s1).
4 unequal(s1,s2).
5 pos(s1,0,i).
6 pos(s2,1,i).
8 -pos(s1,X,Z) | -pos(s2,Y,Z) | -unequal(X,Y)
               | light(off,Z).
9 -pos(X,0,Z) | pos(X,1,sw(X,Z)).
10 -pos(X,1,Z) | pos(X,0,sw(X,Z)).
11 -pos(X,Y,Z) | -unequal(X,X1) | pos(X,Y,sw(X1,Z)).
12 -light(X,sw(X2,sw(X1,i))) | $ans(X).
15 (10,6) pos(s2,0,sw(s2,i)).
28 (11,15,3) pos(s2,0,sw(s1,sw(s2,i))).
32 (11,5,4) pos(s1,0,sw(s2,i)).
37 (32,9) pos(s1,1,sw(s1,sw(s2,i))).
161 (8,37,28,2) light(off,sw(s1,sw(s2,i))).
168 (161,12) $ans(off).

-------------- statistics -------------
clauses input                12
clauses generated           156
CPU time                      2.52 sec.
                ...
```

Figure 3: Sample Proof

The refutation was done with Hyperresolution as inference rule. For this example no special equality handling is needed (which is in general surely not the case) because only very restricted form of equality is sufficient. Thus, on would expect that the axiomatization can be handled quite efficiently.

The prover needs about three seconds to find a proof and generates 156 resolvents. However, if we try to model longer sequences of switching actions, we soon get into trouble, as the following table shows. The performance figures time_{otter} and gen.res._{otter} refer to the otter prover on a Sun 3 with 10 MByte physical memory. time_{pttp} is the time the Prolog Technology Theorem Prover (PTTP, see [Sti88]) needed on a Symbolics Lisp-Machine with an equivalent axiomatization.

switches	$gen.res._{otter}$	$time_{otter}$	$time_{pttp}$
2	156	2.5	
4	197	2.7	0.8
6	447	14.1	5.7
8	1791	159.0	36.7
10	7167	2238.2	205.0
12	28671	34155.5	1124.0

Note that his surprising inefficiency is neither caused by a large amount of frame axioms (there is only one), nor by the theorem prover[2]. However it's run-time grows exponentially, as well.) The actual trouble is the fact that the domain the prover works in is completely unstructured: all formulas describing different states of the world are mixed up which leads to a combinatorial explosion of possible resolvents, most being useless for the proof. Solving serious planning tasks is obviously not possible in this way.

From a theoretical point of view, there are two ways to tackle this lack of efficiency: *restricting* the language so that it can be handled by less complex algorithms, or making the algorithm more efficient by decreasing the amount of inferences that can be done. As the first alternative seems to be unreasonable, because some domains can not even be axiomatized in first-order predicate calculus appropriately, we will focus on the second. What is needed is a formalism to guide inference. This is basically what was done in STRIPS.

2.2 The STRIPS Approach

STRIPS-like systems use a less expressive formalism for describing situations, where disjunctions cannot be expressed. Actions are modelled as *preconditions*, *add-lists*, and *delete-lists*. The states of the world are regarded as conjunctions of logical literals and the add- and delete-lists say how the world will change if an action is executed, by giving a set of literals to be added to, or deleted from the conjunctions.

When we try to model the above example in 'pure' STRIPS, we are running into a problem: the 'switch' action `sw` was used for changing the state of a switch from either 0 to 1, or vice versa. This cannot be expressed directly, instead, we would have to introduce two actions: one for switching a switch from 1 to 0, and one for switching from 0 to 1. This means also that it is not possible to express the goal in the same way as above, which comes from the limited expressiveness of the STRIPS formalism compared with situation calculus.

STRIPS avoids the problems of situation calculus by structuring the domain and restricting the use of theorem proving. Theorem proving is only done within one situation in order to find out what actions are applicable, i.e. which actions' preconditions are entailed by a given situation. The prover never has to consider more that one situation and the actual finding of a plan is done by *search*. This is fairly efficient, unfortunately the *semantics* of STRIPS-like formalisms is rather unclear. The main reason for this is that it is in general not possible to include all formulas that may change after the execution of an action in its add- and delete-list (see [Lif87] for further details).

[2]The reason PTTP handles the problem considerably faster is chiefly that the axiomatization is a Horn set.

So, STRIPS 'guides' the search of the prover on a meta-level, which makes it hard to cope with formal properties. The next sections argue that a modal framework gives a chance to nicely structure the domain and preserve semantic clarity.

3 What are Modal Logics?

Propositional (resp. first-order) modal logics are extensions of classical propositional (resp. first-order) predicate logic by the box-operator '$\Box$'. '$\Box$' can be applied to any formula ϕ:'$\Box\phi$' has the meaning 'it is necessary that ϕ'. Its dual, the diamond-operator '$\Diamond$' ('it is possible that') is definable by setting $\Diamond = \neg\Box\neg$: note the strong ressemblance with the quantifiers '$\exists$' and '$\forall$'='$\neg\exists\neg$'. The language $\mathcal{L}_{modal}$ of modal predicate logic is essentially that of predicate logic augmented by '$\Box$', $\mathcal{L}_{modal} = \mathcal{L}_{pred} \cup \{\Box\}$, the set of terms remains the same and the formulas Fml_{modal} are defined inductively, as in classical logic, with the only additional rule: If 'ϕ' is a formula, so is '$\Box\phi$' (cf. [Fit83], [HC68]).

We now want to give a precise meaning to the operator '$\Box$' by considering an appropriate semantics and defining the notion of a first-order modal structure $\mathcal{K}$, also called *first-order Kripke structure*. Recall that in predicate logic, a $\mathcal{L}$-structure $\mathcal{A}$ consists of a non-empty set A (the *universe* of $\mathcal{A}$), where the quantifiers range over, and an interpretation $I^{\mathcal{A}}$ which maps the constant-, function- and relation-symbols of $\mathcal{L}$ to elements of, functions and relations over A: $\mathcal{A} = (\mathcal{A}, \mathcal{I}^{\mathcal{A}})$.

A *Kripke* $\mathcal{L}_{modal}$*-structure* $\mathcal{K}$ is a pair $\mathcal{K} = (P_{worlds}, R^{\mathcal{K}})$ with:

- $P_{worlds} = \{\mathcal{W}_j : j \in J\}$ is a set of ordinary first-order $\mathcal{L}$-structures $\mathcal{W}_j = (W_j, I^{\mathcal{W}_j})$, called the 'possible worlds of $\mathcal{K}$' (they are usually written with lowercase letters w_j).
- $R^{\mathcal{K}}$ is a binary relation on P_{worlds}: '$w_1 R^{\mathcal{K}} w_2$' meaning that world w_2 is accessible from world w_1 in $\mathcal{K}$.

Finally, a relation $(\mathcal{K}, w) \models \phi$ between Kripke structures with a distinguished world $w \in P_{worlds}$ and formulas ϕ of $\mathcal{L}_{modal}$ can be defined in a straightforward way by induction on the complexity on ϕ (the only interesting case being formulas involving the box-operator $\Box$):

- If ϕ is not of the form $\phi = \Box\psi$, then proceed as usual (by induction).
- If ϕ has the form $\phi = \Box\psi$, we define:

$$(\mathcal{K}, w) \models \Box\psi \quad \underline{:\text{iff}} \quad \begin{array}{l} \text{for all worlds } w\prime \\ \text{accessible from w } (w R^{\mathcal{K}} w\prime) \\ \text{we have: } (\mathcal{K}, w\prime) \models \psi. \end{array} \tag{1}$$

The last step is to define a Kripke structure $\mathcal{K}$ to be a model of a sentence ϕ, if ϕ is valid in all worlds of $\mathcal{K}$: $\mathcal{K} \models \phi$ $\underline{:\text{iff}}$ for all $w \in P_{worlds}$: $(\mathcal{K}, w) \models \phi$.

Various special modal logics can be defined by imposing certain conditions on the accessiblility relation R (*symmetry, reflexivity, transitivity, seriality*) or on the universes of the worlds (e.g. *monotony*: "if $\mathcal{W}_1 R^{\mathcal{K}} \mathcal{W}_2$, then $W_1 \subseteq W_2$", or the *constant domain assumption*: "$\forall j \; W_j = W_0$"). In many cases, such conditions are reflected by axioms formalizable in $\mathcal{L}_{modal}$ itself, so that sound and complete *inference procedures* can be stated.

To summarize, the universe of a Kripke structure $\mathcal{K}$ is not only a set of individuals, but a set of *worlds* (or *situations*), that are themselves sets of objects together with interpretations. This gives us help for structuring the *real world*, like, e.g. in many-sorted logics. But more importantly, we also have a mechanism (the $\Box$-operator) at hand, that allows us, to switch from some worlds to others (for example when an action or a planning step occurs) and to formulate axioms about.

4 Modal Logic Approaches to Planning

This section gives an idea how formalisms based on modal logic frameworks can be employed to avoid the problems of STRIPS and the inefficiency of the situation calculus: they provide a formalism for employing theoretically well founded deductive techniques in structured domains.

We will first describe a system for reasoning about actions proposed by Ginsberg and Smith. Their approach is not inherently based on a concrete modal logic, but gives an idea how to model possible worlds by describing the access from one world to another.

The rest of the section reviews an approach entirely based on modal logic, namely Henry Kautz' work on planning by dynamic logic.

4.1 Ginsberg/Smith's Possible Worlds

Ginsberg and Smith [GS88] describe a state of the world by logical formulas, and the resulting state of affairs after executing an action is computed by evaluating consistency properties an these formulas.

Three sets are used to describe a planning set-up: first, a set D of *domain constraints* saying what is inherently true in the domain and can never be changed by an action. These domain constraints are usually conditions like 'one thing can only at one place at a time'. Second, a set S describing the initial state of affairs. Finally, a set of actions, whereas each action $a \in A$ is given as its set of preconditions $P(a)$, and effects $C(a)$.

If an action a is executed in some situation S, the *potential* subsequent states of the world $T(S, a)$ are defined as:

$$T(S,a) = \{t | t = D \cup C(a) \cup M, M \subseteq S \text{ and } t \not\models \mathit{false}\} \tag{2}$$

This means, a potential world is a consistent set containing the effects of an action, and the formulas describing the preceding state S. Additionally, the domain constraints have to hold.

The set of *possible* worlds $\Pi(S, a)$ is then computed from $T(S, a)$ by considering some partial ordering "$\preceq$": $\Pi(S, a) = \{t | t \in T(S, a)$ and t is $\preceq$-maximal$\}$.

Thus, possible worlds are these potential worlds that are maximal in some sense. The crucial point is actually defining such an appropriate partial ordering "$\preceq$". Ideally, it should select exactly the world we intuitively have in mind as the result of executing an action. This will of course be nearly impossible in practice for real-world domains: usually the ordering will be to weak and select a set of possible worlds. This is a similar phenomenon as "multiple extensions" in default reasoning.

Ginsberg and Smith propose set inclusion as a first idea for "$\preceq$". One can also think of more semantic oriented orderings, like comparing the underlying models according to some measurement, an idea taken from Winslett [Win88].

If we do require a unique subsequent world, the result $do(a, S)$ of executing an action a in a situation S can then be formalized as (3).

$$do(a,S) = \begin{cases} S & \text{if } \Pi(S,a) = \emptyset \text{ or } S \not\models P(a) \\ \bigcap_{\pi_i} \pi_i \in \Pi(S,a) & \text{otherwise} \end{cases} \tag{3}$$

As far as the authors know, the above methods have not yet been integrated in a planning system that is supposed to carry out 'serious' planning tasks. It seems therefor premature to predict their usefulness. Subsequent work of the authors is supposed to investigate this.

4.2 Henry Kautz' Dynamic Logic

Kautz's [Kau82] describes a first first-order dynamic logic framework for plan generation. Within dynamic logic a plan is regarded as a reachability-relation over possible worlds (in Kripke's sense). Let P be a plan specified as a sequence of actions, then $[P]\, q$ is true in some world $\mathcal{W}$, if q is true in every world $\mathcal{W}'$ reachable from $\mathcal{W}$ by executing P.

For the subsequent, assume $\bar{X}$ denotes a set of variables, and $p(\bar{X})$ means the variables $\bar{X}$ appear free in p. A planning set-up is then:

1. A set A of *actions* of the form
$$\forall \bar{X}\colon \mathit{pre}(\bar{U}) \rightarrow [\mathit{act}(\bar{V})]\, \mathit{con}(\bar{W})$$
whereas $\mathit{pre}(\bar{U})$ and $\mathit{con}(\bar{W})$ are non-modal quantifier-free formulas defining the preconditions and postconditions of an action $\mathit{act}(\bar{V})$, and $\bar{X} = \bar{U} \cup \bar{V} \cup \bar{W}$.
For example:
$\forall X\colon \mathit{block}(X)\ \&\ \mathit{clear}(X)\ \&\ \mathit{handempty} \rightarrow\ [\mathit{pick\text{–}up}(X)]\, \mathit{hold}(X)$.

2. A set D of first-order formulas axiomatizing the *constraints* of the problem domain.

3. A set F of *frame axioms* of the form $\forall \bar{X}\colon p \rightarrow [a]\, q$ with $a \in A$ and $D \vdash p \rightarrow q$, defining what an action leaves *unchanged*.

4. A set R of *plan restrictions* that define the requirements that the final plan has to fulfill. They have the form
$$\forall \bar{X}\colon r(\bar{U}) \rightarrow [u(\bar{V})]\, s(\bar{W})$$
$r(\bar{U})$ being $\exists$-first non-modal formula, $s(\bar{W})$ is an $\forall$-first non-modal formula, and $\bar{X} = \bar{U} \cup \bar{V} \cup \bar{W}$.
This formalizes conditions for a plan u to be a *solution* for a planning problem: if $r(\bar{U})$ is valid in the initial situation, then $s(\bar{W})$ must hold after executing u.

The kernel of a planning algorithm within this framework works by calculating strongest postconditions $\mathcal{SP}$s or weakest preconditions $\mathcal{WP}$s of actions and conditions, for planning forward or backward, respectively. Due to space requirements, we cannot describe the complete algorithm here. Very simplified, the principle is as follows:

Suppose $G = A \cup F \cup D$ and $P = <a_1, a_2, \ldots, a_n>$ is a nonempty plan with atomic actions a_i. P satisfies some restriction $r \rightarrow [a_1, a_2, \ldots, a_n]\, s \in R$ if, in case of

forward planning $G \vdash \mathcal{SP}(r, a_1) \rightarrow [a_2, \ldots, a_n]\, s$, and, in case of

backward planning $G \vdash r \rightarrow [a_1, \ldots, a_{n-1}]\ \mathcal{WP}(a_n, s)$.

Based on this idea a planning procedure can be set up that proceeds recursively by substituting the left-hand side r of the restriction by $r' = \mathcal{SP}(r, a_1)$ (or s by $s' = \mathcal{WP}(a_n, s)$, respectively), until there is some r' with $G \vdash r' \rightarrow s$ (or $G \vdash r \rightarrow s'$).

Kautz main interests are of a theoretical nature not supported by practical experience. The algorithm he actually proposes uses bi-directional search and is proven to be correct. For reasons of computational tractability, he had to waive completeness in some cases, but this won't probably hurt very much from a practical point of view. The system itself has unfortunately not yet been implemented.

The major advantage of the approach is its well-founded theoretical basis. Additionally the structure of the domain is used to control the amount of deduction (basically computing $G \vdash r \rightarrow s$), which supports the hope that the necessary amount of frame axioms won't blow up the search space too much.

However, dynamic logic is an extremely rich language and requires costly algorithms to be handled. It was originally developed to cope with program verification tasks, which seems to require more complex formal specifications than planning itself. The authors would think that a less expressive language that can be handled easier might suffice for planning.

4.3 Other Approaches

Few planning systems that are based on a modal logic framework are actually implemented, mostly motivated by theoretic results, not by a concrete application or problem to be solved. One of the recent ones is the system of Schwind and Lafon [LS88], who use a temporal logic for resoning about actions. The inference engine itself is based on a tableau calculus.

A list of theoretical work on the subject should also include the so-called 'Assume' Logic developed by Fariñas del Cerro and Herzig [FnDCH88] whithin the context of formalizing database updates. Their logic of 'elementary changes' can also be used for modelling actions: a modal operator ASS is proposed, whereas a formula $ASS[p]q$ can be interpreted as "if p is added to the current state of the world, then q must be true". To avoid non-determinism and guarantee completeness they allow p to be atomic only, which is a strong restriction.

There are other approaches, e.g. in the area of belief revision, that are not directly based on modal logic but are interesting within this context. Space requirements prevent us from presenting them here, but the 'planning' and 'reasoning about actions' section of IJCAI-89 [Sri89] should give an idea of current research in this area.

5 How about Efficiency?

To construct efficient calculi for modal logics is a difficult problem. Most calculi described in the literature are very special tableau based procedures (see [Fit83]). Instead of defining a special calculus for every variant of a modal-logic, a much better task would be to use existing theorem provers by modifying them appropriately. A natural idea (going back to Tarski and McKinsey) is to translate from modal logic into predicate logic. Ohlbach [Ohl88] constructed a translation of Fml_{modal} into $Fml_{predicate}$ (preserving validity), such that resolution-based theorem provers could be used very efficiently for deciding the validity of such translated formulas. Special properties of the accessibility relation R are reflected by the use of appropriate unification algorithms.

The translation of $\phi \in Fml_{modal}$ into $\phi* \in Fml_{predicate}$ is a fairly delicate matter and cannot be briefly reviewed. We just sketch the method by an example (see [Ohl88]). The key idea is, to introduce a new two-place function "[,]" (this is simply the well-known *list-operator* and we write [a,b] as [ab]) and to augment every predicate- and function-symbol of ϕ by a new argument: this argument (the "world-path") contains the information necessary for allowing the modal operators to be dropped. In addition, the language is also augmented by new (skolem-) constants and (skolem-) functions. These, however, occur only in the new arguments inside of [].

Let ϕ be the formula $\Diamond\Diamond\forall x\ (\Diamond P(x) \wedge \Box Q(x))$.
The translated formula $\phi*$ is $\forall x\ (P([0abc(\mathrm{x})], x) \wedge \forall u Q([0abu], x))$.

This translation is easily motivated by requiring that $\Diamond$ (in the object language) is nothing else, than the $\exists$-operator (in the meta language). Thus, the constants $0, a, b$ and the function $c(x)$ can be seen as skolem functions for this $\exists$-operator.

6 Concluding Remarks

We have tried to give a compact survey of the present state of logic-based automated plan generation with emphasis on the modal and dynamic logic approach. We may summarize

- the research so far is mainly theoretical,
- the theoretical foundations for a dynamic logic planning calculus are available in sufficient detail and rigour, though it is expected that modifications and additions have to be made

when implementing a usable system,

It is promising to take a next step in the development of logic-based automated planning systems: to implement an experimental system and put it to a test in a realistic application.

References

[Cha87] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(2):249–267, May 1987.

[Fit83] Melvin C. Fitting. *Proof Methods for Modal and Intuitionistic Logic*, volume 169 of *Synthese Library.* D. Reidel Publishing Co., Dordrecht, Holland, 1983.

[FnDCH88] L. Fariñas Del Cerro and A. Herzig. An Automated Modal Logic of Elementary Changes. In Mamdani, Smets, Dubois, and Prade, editors, *Non-standard Logics for Automated Reasoning*, chapter 2, pages 63–79. Academic Press, 1988.

[Gin87] Mathew L. Ginsberg. Possible wolds planning. In Michael P. Georgeff and Amy L. Lansky, editors, *Reasoning about Actions and Plans, Proceedings of the 1986 Workshop*, Los Altos, CA, 1987. Morgan Kaufmann.

[GS88] M. L. Ginsberg and D. E. Smith. Possible worlds planning I + II. *Artificial Intelligence*, 35, 1988.

[HC68] G.E. Hughes and M.J. Cresswell. *An Introduction to Modal Logics.* Methuen and Co, Ltd, London, UK, 1968.

[HRS88] Maritta Heisel, Wolfgang Reif, and Werner Stephan. Implementing verification strategies in the kiv-system. In Ewing Lusk and Ross Overbeek, editors, *Proceedings of the 9th Conference on Automated Deduction*, Argonne, Ill, May 1988. Springer Verlag.

[Kau82] Henry Kautz. A first-order dynamic logic for planning. Technical Report TR CSRG-144, Dept. of Computer Science, University of Toronto, Ontario, Canada, 1982.

[Kow79] Robert Kowalski. *Logic for Problem Solving.* North Holland, New York, N.Y., 1979.

[Lif87] Vladimir Lifschitz. On the semantics of STRIPS. In Michael P. Georgeff and Amy L. Lansky, editors, *Reasoning about Actions and Plans, Proceedings of the 1986 Workshop*, Los Altos, CA, 1987. Morgan Kaufmann.

[LS88] E. Lafon and C. B. Schwind. A theorem prover for action performance. In *Proceedings of the European Conference on Artificial Intelligence*, München, W. Germany, August 1988.

[McC88] William W. McCune. Otter users' guide. Argonne Nat'l Laboratories, Argonne, ILL, 1988.

[MH69] John McCarthy and Pat Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*. Edinburgh University Press, Edinburgh, Scotland, 1969.

[Ohl88] Hans Jürgen Ohlbach. A Resolution Calculus for Modal Logics. In Ewing Lusk and Ross Overbeek, editors, *Proceedings of the 9th Conference on Automated Deduction*, Argonne, Ill, May 1988. Springer Verlag.

[Sch87] Peter H. Schmitt. The THOT theorem prover. Technical Report TR 87.9.7, IBM Germany, Heidelberg Scientific Center, Heidelberg, W. Germany, 1987.

[Sri89] N. S. Sridharan, editor. *Eleventh International Joint Conference on Artificial Intelligence*, Dertoit, MI, August 1989.

[Sti88] Mark E. Stickel. A Prolog Technology Theorem Prover. In E. Lusk, R. Overbeek E. Lusk, and R. Overbeek, editors, *9th International Conference on Automated Deduction*, Argonne, Ill., May 1988. Springer-Verlag.

[Win88] Marianne Winslett. Theory Revision Semantics for Use in Reasoning about Actions. In *Proc. AAAI-88*, 1988.