Planning in Description Logics: Deduction versus Satisfiability Testing

Liviu Badea
AI Research Lab
Research Institute for Informatics
8-10 Averescu Blvd., Bucharest, Romania
e-mail: badea@ici.ro

Abstract

Description Logics (DLs) are formalisms for taxonomic reasoning about structured knowledge. Adding the transitive closure of roles to DLs also enables them to represent and reason about actions and plans. The present paper explores several essentially different encodings of planning in Description Logics. We argue that DLs represent an ideal framework for analysing and comparing these approaches. Thus, we have identified two essentially different *deductive* encodings (a "causal" and a "symmetric" one), as well as a *satisfiability*-based approach.

While the causal encoding is more appropriate for reasoning about precondition-triggered causal events, the symmetric encoding is more amenable to reasoning about possible outcomes of courses of actions without actually executing them (while allowing both progression and regression).

In the deductive approaches, the existence of a plan corresponds to an inconsistency proof rather than to a model of some formula. Viewing planning as satisfiability testing addresses this problem by reducing planning to model construction.

1 Introduction

Description Logics (DLs) [11, 3, 20] are formalisms for taxonomic reasoning about structured knowledge. Like their predecessors (semantic networks and frame languages), DLs have been used mainly for representing and reasoning about the domain knowledge of a given problem, usually in the framework of a hybrid architecture.

In-depth theoretical investigations carried out in the last decade [20] have uncovered an almost complete picture of the expressive power and computational complexity of a wide range of Description Logics[1] and provided a firm starting point for considering various extensions. Such extensions were mainly motivated by the limitations of existing DLs in representing various types of knowledge such as modalities and epistemic operators [5, 7], higher-order constructs [9], non-monotonic features [4, 19], Horn rules [25] and many others.

[1]defined in terms of the constructs used.

Description Logics with the transitive closure of roles [2, 30] have also been proposed as a unifying formalism for various class-based representation languages [15] as well as for representing tense [29, 31], epistemic operators, actions and plans [18, 1, 7].

Some of these approaches rely on Schild's correspondence [30] between expressive description logics with the transitive closure of roles and propositional dynamic logic (PDL). Given that PDL has been conceived as a formal approach to reasoning about actions and dynamically evolving systems (such as programs), it may be surprising that so little research has been carried out towards representing planning in description logics.[2]

However, representing and reasoning about actions and planning in DLs is very important for modeling dynamically evolving DL knowledge bases at the conceptual level (as opposed to using an ordinary DL in a hybrid architecture, where one is not able to reason about actions *in* the DL, which is therefore incomplete). Combining such an approach with epistemic operators [8] may enable the design of DL-based intelligent agents.

The main goal of this paper is to present an in-depth analysis of the various approaches to encoding actions and planning in Description Logics. This issue is not entirely straight-forward, since – contrary to a first impression – there are several essentially different ways of encoding actions and planning problems in DLs. For example, we can encode planning either as deduction or as satisfiability testing. Viewed as a deduction problem, we have identified two essentially different encodings of planning: a *"causal"* and a *"symmetrical"* one.

Being asymmetrical (non-reversible), the causal encoding is more appropriate for reasoning about precondition-triggered causal events (even non-deterministic ones). However, it does not allow for a straight-forward approach to goal regression.

[2]We are considering description logics rather than plain PDL for encoding actions for two important reasons. First, description logics may provide additional constructs useful for integrating a theory of action in a more extensive KR framework. Second, in DLs it is possible to impose constraints on specific state instances (using assertional axioms). This is not possible in PDL.

The symmetrical approach, on the other hand, is more amenable to reasoning about possible outcomes of courses of actions without actually executing them. The symmetrical (reversible) form of this representation allows both progressive (forward) and regressive (backward) reasoning.

The above-mentioned deductive approaches to planning could be used together in a realistic setting in which causal external events (even non-deterministic ones) as well as actions under the control of intelligent agents coexist.

Planning viewed as deduction has its own problems in the framework of Description Logics because the existence of a plan amounts to proving the *validity* of a certain DL formula. But since the validity of a formula is usually reduced in DLs to the inconsistency of the negated formula, we reduce planning to proving inconsistency. This may seem somehow counter-intuitive, since we might have expected that a plan would correspond to a DL model of some formula rather than to a proof that no such model exists. Viewing planning as satisfiability testing (in the spirit of [24]) addresses this problem by reducing planning to model construction. A small disadvantage of this approach might be the requirement of a completely specified initial state, but any incompletely specified state can be easily completed.

A tableaux-based algorithm for checking consistency in a DL with the transitive closure of roles [8] has been developed and used for testing the SAT-based approach.

2 The $\mathcal{ALC}^\star$ Description Logic

Description logics are hybrid systems which separate the described knowledge in two distinct categories: *terminological* and *assertional* knowledge. The terminological knowledge is generic (intensional) and refers to classes of objects and their relationships, while the assertional knowledge is extensional as it describes particular instances (individuals) of these classes. Unless concept reification is allowed [9], these two levels are completely disjoint since a given object cannot be at the same time a concept *and* an instance.

Description logics further distinguish between two kinds of terminological knowledge, namely concepts and roles. *Concepts* are essentially unary predicates interpreted as sets of individuals, while *roles* represent binary predicates interpreted as binary relations between individuals.

In the following, we consider the smallest description logic able to express actions and conditional plans, namely the regular closure $\mathcal{ALC}^\star$ of Schmidt-Schauß and Smolka's $\mathcal{ALC}$ language [32] extended with identities $id(C)$. Compared with other description logics, $\mathcal{ALC}^\star$ is quite expressive, since it allows the internalization of general (possibly cyclic) concept definitions by means of the transitive closure of roles.

The following concept and role constructors are available in $\mathcal{ALC}^\star$:

$$C ::= CN \mid \top \mid \bot \mid C_1 \wedge C_2 \mid C_1 \vee C_2 \mid \neg C \mid \langle R \rangle C \mid [R]C$$

$$R ::= RN \mid id(C) \mid R^- \mid R_1 \vee R_2 \mid R_1 \circ R_2 \mid R^*$$

where CN, RN are concept and role names respectively, $\langle R \rangle C$ are existential restrictions (sometimes written as $\exists R.C$), while $[R]C$ are value restrictions (written also as $\forall R.C$). Role union ($R_1 \vee R_2$), composition ($R_1 \circ R_2$) and reflexive-transitive closure (R^*) allow for regular role expressions, whereas the identity role construct $id(C)$ is useful for representing conditional plans. Role inverses (R^-) are needed for goal regression.

The semantics of the above constructors in the interpretation $\mathcal{I}$[3] is given by the following conditions:

$$\begin{array}{rcl}
\top^\mathcal{I} & = & \mathcal{D}^\mathcal{I} \\
\bot^\mathcal{I} & = & \emptyset \\
(C_1 \wedge C_2)^\mathcal{I} & = & C_1^\mathcal{I} \cap C_2^\mathcal{I} \\
(C_1 \vee C_2)^\mathcal{I} & = & C_1^\mathcal{I} \cup C_2^\mathcal{I} \\
(\neg C)^\mathcal{I} & = & \mathcal{D}^\mathcal{I} \setminus C^\mathcal{I} \\
(\langle R \rangle C)^\mathcal{I} & = & \{s \in \mathcal{D}^\mathcal{I} \mid \exists s' \in \mathcal{D}^\mathcal{I}.(s,s') \in R^\mathcal{I} \wedge s' \in C^\mathcal{I}\} \\
([R]C)^\mathcal{I} & = & \{s \in \mathcal{D}^\mathcal{I} \mid \forall s' \in \mathcal{D}^\mathcal{I}.(s,s') \in R^\mathcal{I} \rightarrow s' \in C^\mathcal{I}\} \\
id(C)^\mathcal{I} & = & \{(s,s) \mid s \in C^\mathcal{I}\} \\
(R^-)^\mathcal{I} & = & \{(s',s) \mid (s,s') \in R^\mathcal{I}\} \\
(R_1 \vee R_2)^\mathcal{I} & = & R_1^\mathcal{I} \cup R_2^\mathcal{I} \\
(R_1 \circ R_2)^\mathcal{I} & = & R_1^\mathcal{I} \circ R_2^\mathcal{I} \\
(R^*)^\mathcal{I} & = & \bigcup_{n \geq 0} (R^\mathcal{I})^n
\end{array}$$

Recall that the transitive closure of roles is not expressible in first-order logic (it requires at least fixpoint logics). However it is essential not only for encoding general terminological axioms, but also for our encodings of planning in $\mathcal{ALC}^\star$.

In order to represent the symmetric encoding, we will need a more expressive DL, namely one that provides explicit fixpoint constructors. The $\mathcal{ALC}^\mu$ language [28, 13] is strictly more expressive than $\mathcal{ALC}^\star$ and provides the following additional concept constructors:

$$C ::= \mu X.C \mid \nu X.C \mid X$$

where X is a "fixpoint variable" which can occur only in the scope of the least/greatest fixpoint constructors $\mu X.C$ and $\nu X.C$ respectively. And although $\mathcal{ALC}^\mu$ admits no role constructors (besides role inverses), the $\mathcal{ALC}^\star$ role constructors (occurring in existential or value restrictions) can be expressed by means of fixpoints. For instance,

$$\begin{array}{rcl}
\langle R^* \rangle C & = & \mu X.(X \vee \langle R \rangle C) \\
[R^*]C & = & \nu X.(X \wedge [R]C).
\end{array}$$

The terminological knowledge base (also called TBox) consists of general concept implications of the form $C_1 \rightarrow$

[3] which associates a subset $C^\mathcal{I} \subseteq \mathcal{D}^\mathcal{I}$ of the interpretation domain $\mathcal{D}^\mathcal{I}$ to each concept C and a binary relation $R^\mathcal{I} \subseteq \mathcal{D}^\mathcal{I} \times \mathcal{D}^\mathcal{I}$ to each role R.

C_2,[4] as well as validity axioms C (expressing the validity of the concept term C). Their semantic interpretation is $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ and $C^{\mathcal{I}} = \mathcal{D}^{\mathcal{I}}$ respectively. (Of course, the implication $C_1 \rightarrow C_2$ can be reduced to the validity axiom $\neg C_1 \vee C_2$. On the other hand, validity axioms C can be internalized in $\mathcal{ALC}^{\star}$ using role terms of the form $[\mathcal{R}^*]C$ where $\mathcal{R}$ is the disjunction of all role names occurring in the knowledge base [2, 30].)

The assertional knowledge base (also called ABox) consists of assertional axioms of the form

$s : C$	(concept instance assertions)
$(s, s') : R$	(role tuple assertions)

which are interpreted semantically as $s^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(s^{\mathcal{I}}, s'^{\mathcal{I}}) \in R^{\mathcal{I}}$ respectively.

An interpretation satisfying the terminological and assertional axioms of a knowledge base (KB) is called a *model* of the KB. A KB is called *consistent* iff it admits a model and inconsistent otherwise.

A concept C is called *satisfiable* w.r.t. a given KB iff it admits a non-void extension $C^{\mathcal{I}}$ in a model $\mathcal{I}$ of the KB. C is *valid* in a KB whenever $C^{\mathcal{I}} = \mathcal{D}^{\mathcal{I}}$ in all models $\mathcal{I}$ of the KB. C is valid iff its negation $\neg C$ is unsatisfiable.

Testing satisfiability (and therefore also validity) in $\mathcal{ALC}^{\star}$ as well as $\mathcal{ALC}^{\mu}$ is decidable, more precisely EXPTIME-complete [21, 13].

3 Encoding actions and planning in Description Logics

As we have mentioned in the introduction, Description Logics with the transitive closure of roles like $\mathcal{ALC}^{\star}$ can be used not only for representing taxonomic domain knowledge, but also for describing actions and plans. This can be achieved by regarding a DL role A as an action which transforms states S from (the extension of) the role's domain into states S' from (the extension of) its range: $(S, S') \in A^{\mathcal{I}}$. Thus, the value restriction $[A]C$ can be interpreted as the necessary precondition for action A to achieve the effect C.

Conditions (fluents) from our theory of action will be represented in a DL by concepts, while actions will be encoded as role names. Of course, (possibly conditional) plans can be represented as complex role terms, the role constructors $\vee, \circ$ and $*$ being interpreted as control structures (nondeterministic choice, sequence and nondeterministic iteration respectively). The identity role constructor $id(C)$ can be interpreted as a "test", which can be used for expressing the usual structured control primitives *if*, *while* and *repeat*:

$$\begin{aligned} \textit{if } C \textit{ then } A_1 \textit{ else } A_2 &= id(C) \circ A_1 \vee id(\neg C) \circ A_2 \\ \textit{while } C \textit{ do } A &= (id(C) \circ A)^* \circ id(\neg C) \\ \textit{repeat } A \textit{ until } C &= A \circ (id(\neg C) \circ A)^* \circ id(C) \end{aligned}$$

In the following, we will deal with propositional STRIPS actions A described in terms of the following three condition sets (containing only *non-negated* atomic fluents):

- preconditions Pre(A) (the conditions necessary for executing A)
- positive effects Add(A) (the fluents added by A's execution)
- negative effects Del(A) (the fluents deleted/falsified by A's execution).

The following relationships between the above condition-sets are assumed: $Pre(A) \cap Add(A) = \emptyset$ and $Del(A) \subseteq Pre(A)$.

For example, the simple blocks-world action `A = move-X-Y-Z` (which moves the block `X` from `Y` onto `Z`) admits the following STRIPS description:

```
Pre(A) = {on-X-Y, clear-X, clear-Z}
Add(A) = {on-X-Z, clear-Y}
Del(A) = {on-X-Y, clear-Z}.
```

As we have already mentioned, there are several alternative approaches to encoding and reasoning about actions and plans in $\mathcal{ALC}^{\star}$. The two main categories of approaches are the *deductive* and the *satisfiability-based* ones. We start by discussing the deductive approaches.

3.1 Deductive planning in Description Logics

We have identified two essentially different encodings of planning as deduction: a *causal* (asymmetrical) one and a *symmetrical* one.

The causal (asymmetric) encoding

The causal encoding amounts to enforcing the existence of an action execution A whenever the preconditions $Pre(A)$ of A are verified:

$$[\mathsf{Eff}_{DED-CAUS}] \quad Pre(A) \rightarrow \langle A \rangle Add(A)$$

(where condition sets appearing in logical formulae are interpreted conjunctively).

The semantical interpretation of the above axiom[5]

$$holds(Pre(A), S) \rightarrow \exists S'.do(A, S, S') \wedge holds(Add(A), S')$$

shows that all actions A executable in state S (whose preconditions are satisfied in S) are actually executed in S, leading to (separate) successor states S'. The causal approach therefore encodes the entire search space (with all possible action executions from a given state) in its models.

Besides the explicit effects of action A, described by axiom $[\mathsf{Eff}_{DED-CAUS}]$, it is necessary to describe the persistence of the conditions (fluents) left unmodified by A. This is achieved by means of frame axioms of the form[6]

---

[4] where C_1 and C_2 can be arbitrary concept terms. All usual concept definitions, including cyclic and multiple definitions, are expressible using such general implications.

[5] We write $holds(C, S)$ instead of $S \in C^{\mathcal{I}}$ and $do(A, S, S')$ instead of $(S, S') \in A^{\mathcal{I}}$ in order to emphasize the fact that the interpretations of DL formulae are essentially situation calculus formulae.

[6] Since a given action typically affects only a small number of conditions, we will have to write $\mathcal{O}(\mathcal{A} \cdot \mathcal{C})$ such frame

$$[\mathsf{Fr}_{DED}] \quad C \rightarrow [A]C$$
$$\textit{for all} \quad C \in Conditions - (Del(A) \cup Add(A)).$$

Note that since we are in a deductive setting it is not necessary to explicitly mention the deleted effects in the consequent of the above axiom. In other words, a stronger version like $Pre(A) \rightarrow \langle A \rangle (Add(A) \wedge \neg Del(A))$ is not needed as long as the frame axioms do not allow the persistence of deleted effects. Similarly, a stronger version like $Pre(A) \rightarrow \langle A \rangle \top \wedge [A]Add(A)$ is also unnecessary for deductive planning.

A *planning problem* is usually specified by providing a (possibly incomplete) initial state described by the concept *Initial* (a conjunction of the concept names representing the conditions initially true) and a final (goal) state *Final*. For example, we can represent the Sussman anomaly problem in the blocks world as

```
Initial = on-c-a ∧ on-a-table ∧ on-b-table
        ∧ clear-c ∧ clear-b
Final = on-a-b ∧ on-b-c.
```

The most straight-forward approach to such a problem would be to reduce it to proving a theorem of the form

$$Initial \rightarrow \langle ?Plan \rangle Final$$

involving a meta-variable $?Plan$. Unfortunately, most description logic theorem provers do not allow for role variables (especially those with powerful role constructors, like $\mathcal{ALC}^\star$), so the simple approach above is not directly feasible.

If we knew the role term representing the plan $Plan = A_{i_1} \circ A_{i_2} \circ \ldots \circ A_{i_n}$, then the validity of the formula

$$Initial \rightarrow \langle Plan \rangle Final \qquad (1)$$

would be equivalent with the validity of the plan.

However, since we do not know $Plan$, we need to try proving (1) for all possible action sequences $Plan$. Unfortunately, this cannot be done effectively, since there are infinitely many such action sequences and therefore infinitely many theorems to try proving. Therefore, we will consider reducing the problem to proving a *single* formula containing a disjunction of all possible action sequences:

$$\begin{aligned} Initial \rightarrow\ & Final \vee \qquad (2) \\ & \langle A_1 \rangle Final \vee \langle A_2 \rangle Final \vee \ldots \vee \\ & \langle A_1 \circ A_1 \rangle Final \vee \langle A_1 \circ A_2 \rangle Final \vee \ldots \vee \\ & \langle A_2 \circ A_1 \rangle Final \vee \ldots . \end{aligned}$$

Since a disjunction of existential restrictions can be rewritten as an existential restriction $\langle R_1 \rangle q \vee \langle R_2 \rangle q = \langle R_1 \vee R_2 \rangle q$, we can reduce (1) to

$$[\mathsf{Plan}_{DED-CAUS}] \quad Initial \rightarrow \langle Any^* \rangle Final$$

where $Any = A_1 \vee A_2 \vee \ldots \vee A_k$ is the disjunction of all atomic actions occurring in the problem (the "repertory of actions") [6, 16]. Note that the role term Any^* plays the role of the meta-variable $?Plan$.

The relationship between (1) and (2) is subtle and requires some explanations. In general, a proof of (2) does not entail the existence of a proof of (1) for some $Plan$ (although the reverse is true) because (2) requires that for each state S verifying *Initial* we find a sequence of actions $Plan$ such that $\langle Plan \rangle Final$ holds – but $Plan$ need not be the same for all such states S!

The most straight-forward solution to this problem (pursued for example in [16][7]) would be to require complete state specifications (that do not allow for essentially different states S) and to make sure that the axioms constrain the successor states to be also completely specified. This amounts roughly to combining the axioms from our deductive (causal and symmetric) *and* SAT-based approaches. The problem with this approach lies in the large number of axioms employed which may significantly slow down a theorem prover, especially because reasoning with complete state specifications may be at a too fine-grained level, i.e. very close to "blind search" in the much too big space of complete state descriptions.

What we would like to achieve is to be able to reason with incomplete state specifications (for example by propagating only "weakest preconditions" and/or "strongest effects" instead of complete state information).

As shown above, incomplete state specifications give rise to situations in which a proof of (2) may construct a different $Plan$ for each completion (state) S verifying the incomplete initial state specification *Initial*. This ensures the existence of such a plan $Plan_S$ for each state S, but a given $Plan_S$ may not be applicable in all states

axioms. Their number can be reduced to $\mathcal{O}(\mathcal{C})$ by grouping the actions $A, A', A'', \ldots$ that leave C unaffected:

$$C \rightarrow [A \vee A' \vee A'' \vee \ldots]C.$$

Conditions is the (finite) set of atomic conditions occurring in the problem, $\mathcal{C} = |Conditions|$ and $\mathcal{A}$ the number of atomic actions.

[7] De Giacomo and Lenzerini do not explicitly state that the initial state should be completely specified. However, their approach of reducing planning to proving the validity of $Initial \rightarrow \langle Any^* \rangle Final$ fails in the case of incompletely specified initial states due to their allowing actions with negated preconditions.

For example, consider $Initial = p$, $Final = q$ and an action a with $Pre(a) = \{\neg q\}$, $Add(a) = \{q\}$, $Del(a) = \{\neg q\}$, described by means of the following axioms

$$\begin{aligned} \neg q &\rightarrow \langle a \rangle \top \\ \langle a \rangle \top &\rightarrow \neg q \\ & [a]q. \end{aligned}$$

Initial is incompletely specified since the value of q is not mentioned. Therefore, two possibilities arise: either q is true in *Initial* (case in which the empty plan $Plan' = id$ is the only solution), or $\neg q$ holds in *Initial* (case in which $Plan'' = a$ is the only solution), so there exists no "global" plan. But the formula $Initial \rightarrow \langle Any^* \rangle Final$ (i.e. $p \rightarrow \langle a^* \rangle q$) is nevertheless provable using the above axioms, showing that the approach in [16] fails in this case.

S' verifying the incomplete specification $Initial$. On the other hand, the planning problem amounts to finding a plan that is guaranteed to work no matter what state we are in.[8]

Thus it may seem that it is impossible to reduce planning to proving a DL formula, so as to take advantage of an existing DL theorem prover. Therefore, it may seem we need to use a syntactical plan generation approach (like in [27]) by writing a specialized planning algorithm on top of a Description Logic (or Dynamic Logic) theorem prover. However, writing such a specialized planning algorithm external to the DL is somewhat inappropriate in a KR formalism like Description Logics, where we would like to be able to impose various constraints on the plan.

Fortunately, we can avoid this by showing that, although (1) and (2) are not equivalent in the general case, we can nevertheless recover a "global" plan (i.e. a solution to (1)) from a proof of (2). In order to do this, we shall single out a state S whose plan $Plan_S$ constructed according to (2) is also applicable to all the other states S'. The state S with this property is the completion of the (incomplete) initial state specification $Initial$ (obtained by conjoining to $Initial$ a negated literal $\neg C$ for each action precondition C not specified in $Initial$).

Due to our assumption that the precondition lists of actions contain only positive literals[9] and by additionally disallowing constraints (terminological axioms) involving preconditions of actions, the negated literals in state descriptions do not influence the executability of actions. (Note also that in the deductive settings, negated conditions are *not* propagated by frame axioms.) Therefore, the plan $Plan_S$ for the completed state S will be applicable in all other states as well and will be a "global" plan. In our setting, (1) and (2) are therefore equivalent and we can safely reduce the planning problem to finding a proof for (2).

The planning problem has thus been reduced to proving the $\mathcal{ALC}^{\star}$ theorem $[\mathsf{Plan}_{DED-CAUS}]$. But proving the validity of such a formula is usually reduced in DLs to proving the inconsistency of its negation:

$$[\neg\mathsf{Plan}_{DED-CAUS}] \quad Initial \wedge [Any^{*}]\neg Final.$$

Drawing an analogy with the answer-set of a logic programming query, we should be able to modify a DL theorem prover so that it returns a "falsifying interpretation" $\mathcal{I}$ for each inconsistent query $[\neg\mathsf{Plan}_{DED-CAUS}]$. This interpretation would be constructed while trying to build a model of the formula $[\neg\mathsf{Plan}_{DED-CAUS}]$. Whenever a plan exists, the latter formula is inconsistent due to a clash involving the goal condition $Final$ and the plan can be reconstructed from the (inconsistent) interpretation $\mathcal{I}$ built so far.

[8] "Conditional" plans like $Plan_S$ may be interesting in their own right, but we do not explore this issue further.

[9] If an action had a negated literal $\neg C$ as a precondition, we could replace it by the precondition C' and define $C' = \neg C$ as an axiom in the DL.

Note that unlike many planning systems which do not have a sound and complete stopping criterion[10], the above approach to planning provides a decidable, sound and complete planning algorithm. This is especially important for proving that no plan exists.

The above reduction of plan construction to an inconsistency proof may seem somehow counter-intuitive in DLs, since we might have expected that a plan would correspond to a *model* of some formula rather than to a proof that no such model exists. This viewpoint will be pursued in the satisfiability-based encoding presented below.

The causal encoding presented above is more appropriate for reasoning about precondition-triggered causal events of the environment (as opposed to actions under the full control of agents – which may or may not choose to execute them, even if the preconditions are satisfied). It is also able to represent non-deterministic causal events (events with multiple possible outcomes), as described in more detail in section 4.1. But since causal events are not necessarily reversible, the causal encoding is asymmetrical in a certain sense, and it does not allow a straight-forward representation of goal regression (i.e. reasoning backward from the goals $Final$). Reasoning in the causal encoding is therefore limited to progression (forward reasoning from the initial state), which may be inefficient (but it is the only type of reasoning possible when dealing with such precondition-triggered causal events).

The symmetrical encoding

The symmetrical encoding deals with representing the reasoning about possible outcomes of courses of action without actually executing the actions. More precisely, we shall write axioms saying that whenever the preconditions $Pre(A)$ of action A are verified and A is executed, the positive effects of A must hold in the successor state:

$$[\mathsf{Eff}_{DED-SYM}] \quad Pre(A) \rightarrow [A]Add(A).$$

This can be seen more easily in the semantic interpretation:

$$holds(Pre(A), S) \wedge do(A, S, S') \rightarrow holds(Add(A), S').$$

Similarly with the causal setting, we do not need to explicitly mention the deleted effects $\neg Del(A)$ in the consequent of the above axiom (because we are in a deductive setting).

The frame axioms $[\mathsf{Fr}_{DED}]$ are identical to the ones used in the causal setting.

Finally, the validity of a plan $Plan = A_{i_1} \circ A_{i_2} \circ \ldots \circ A_{i_n}$ is equivalent to proving the theorem $Initial \rightarrow [Plan]Final$. However, since we do not know $Plan$, we need to prove a formula containing a disjunction of all possible action sequences[11]:

$$Initial \quad \rightarrow \quad Final \vee \tag{3}$$

[10] They usually set an ad-hoc bound on the length of the plan.

[11] Similar considerations as in the case of the causal setting apply here.

$$[A_1]Final \vee [A_2]Final \vee \ldots \vee$$
$$[A_1 \circ A_1]Final \vee [A_1 \circ A_2]Final \vee \ldots \vee$$
$$[A_2 \circ A_1]Final \vee \ldots.$$

But unfortunately, the disjunction of value restrictions cannot be rewritten as a single value restriction[12], so we *cannot* reduce (3) to a formula like $Initial \to [Any^*]Final$ (which would be the analog of $[\mathsf{Plan}_{DED-CAUS}]$). In fact, formula (3) cannot be encoded in $\mathcal{ALC}^\star$ (or PDL) and not even in *repeat*-PDL. In order to represent (3), we need the full expressive power of the μ-calculus, i.e. $\mathcal{ALC}^\mu$ (which provides general fixpoint constructors):

$[\mathsf{Plan}_{DED-SYM}]$
$$Initial \to \mu X.(Final \vee [A_1]X \vee \ldots \vee [A_k]X).$$

The validity of $[\mathsf{Plan}_{DED-SYM}]$ is equivalent with the inconsistency of

$[\neg\mathsf{Plan}_{DED-SYM}]$
$$Initial \wedge \nu X.(\neg Final \wedge \langle A_1\rangle X \wedge \ldots \wedge \langle A_k\rangle X).$$

Using a result of Niwinski (mentioned in [34]) saying that the formula $\nu X.(\langle A_1\rangle X \wedge \langle A_2\rangle X)$ is not expressible in *repeat*-PDL, we conclude that neither $[\neg\mathsf{Plan}_{DED-SYM}]$ nor $[\mathsf{Plan}_{DED-SYM}]$ can be expressed in $\mathcal{ALC}^\star$ (not even in its ω-regular extension). Strangely enough, the symmetric encoding requires more expressive power than does the causal encoding. However, reasoning in $\mathcal{ALC}^\mu$ is just as hard/easy as reasoning in $\mathcal{ALC}^\star$ (both are EXPTIME-complete).

Regression The above encoding of planning seems to be more appropriate for progression (i.e. reasoning forward from the initial state and looking for a sequence of actions leading to the goal state). The following results show however that the above axioms can be rewritten in an equivalent form that is more appropriate for regression (backward reasoning from the final state by recursively replacing goals with action subgoals until they are satisfied in the initial state). This shows the intrinsic precondition-effect *symmetry* of the approach.

Proposition 1 *The following axioms are equivalent (1)* $p \to [a]q$ *(2)* $\langle a^-\rangle p \to q$ *and (3)* $\neg q \to [a^-]\neg p$.

Proof. Since (2) and (3) are contra-positives, we need to prove only the equivalence (1) $\iff$ (3). Indeed, $p \to [a]q$ is interpreted as[13]:
$\forall S.p(S) \to \forall S'.\,(a(S,S') \to q(S'))$
$\forall S.\forall S'.\neg p(S) \vee \neg a(S,S') \vee q(S')$
$\forall S'.\forall S.\neg q(S') \to (a^-(S',S) \to \neg p(S))$
i.e. $\neg q \to [a^-]\neg p$ $\square$

The "regressive" forms of the effect and frame axioms are therefore:

[12]Note that $[R_1]q \vee [R_2]q \neq [R_1 \vee R_2]q = [R_1]q \wedge [R_2]q$.

[13]for brevity, we write $p(S)$ instead of $holds(p,S)$ and $a(S,S')$ instead of $do(A,S,S')$.

$[\mathsf{Eff}^-{}_{DED-SYM}]$	$\langle A^-\rangle Pre(A) \to Add(A)$
or equivalently	$\neg Add(A) \to [A^-]\neg Pre(A)$
$[\mathsf{Fr}^-{}_{DED}]$	$\langle A^-\rangle C \to C$
or equivalently	$\neg C \to [A^-]\neg C.$

3.2 Planning as testing satisfiability in $\mathcal{ALC}^\star$

Viewing planning as satisfiability testing amounts to regarding a plan as a model of some formula rather than as a proof that no such model exists (as in the deductive approaches). Planning is thus reduced to model construction, in the spirit of [24]. But unlike Kautz and Selman, who reduce linear-time planning to propositional satisfiability, our approach reduces planning to $\mathcal{ALC}^\star$ satisfiability. A model corresponds thus to a Kripke structure rather than just a propositional truth assignment (as in [24]). Since $\mathcal{ALC}^\star$ provides the transitive closure of roles, we do not need to use (like [24]) iterative deepening over fixed-length planning problems. We additionally ensure the completeness of the termination check (our algorithms always terminate and in case they do so without finding a plan, then it is guaranteed that no such plan exists).

The effect and frame axioms used in the deductive approaches are correct and complete w.r.t. deduction, but they are not strong enough to rule out anomalous models. For example, they admit models in which actions are executed despite the fact that their preconditions are not satisfied. Such models can be avoided by using axioms of the form

$[\mathsf{Pre}_{SAT}]$	$\langle A\rangle\top \to Pre(A)$
or equivalently	$[A^-]Pre(A).$

For precondition-triggered causal events, we impose the executability axioms:

$[\mathsf{Exec}_{SAT}]$ $\quad Pre(A) \to \langle A\rangle\top.$

The following axiom rules out models in which actions are executed but their effects do not hold:

$[\mathsf{Eff}_{SAT}]$ $\quad [A]\mathit{Eff}(A)$

where $\mathit{Eff}(A) = Add(A) \wedge \neg Del(A)$ are the effects of action A[14]. Note that in the deductive setting, only the positive effects $Add(A)$ had to be enforced in the successor states of A. Even if these states would have been *consistent* with $Del(A)$, this would not have been sufficient for executing some other action whose preconditions are in $Del(A)$. $Del(A)$ should have been valid in those states and not just consistent with them.

The effect axiom in the symmetric deductive setting $[\mathsf{Eff}_{DED-SYM}]$ is weaker than its SAT counterpart $[\mathsf{Eff}_{SAT}]$ for two reasons:

[14]$\neg Del(A)$ represents the conjunction of the negated conditions from $Del(A)$.

- [Eff_{SAT}] explicitly enforces $\neg Del(A)$ in the successor states of A
- [$\mathsf{Eff}_{DED-SYM}$] constrains the successor states of A only if the current states verifies the preconditions $Pre(A)$.

[$\mathsf{Eff}_{DED-SYM}$] is too weak for the SAT setting. However, the following intermediate version

$$[\mathsf{Eff'}_{SAT}] \quad Pre(A) \to [A]Eff(A)$$

is equivalent with [Eff_{SAT}] when combined with [Pre_{SAT}]. This can be proved using the following result.

Proposition 2 *The three sets of axioms below are equivalent:*

(1) $\langle a\rangle\top \to p$
$p \to [a]q$

(2) $\langle a\rangle\top \to p$
$[a]q$

(3) $[a^-]p$
$[a]q$.

The frame axioms need to enforce the persistence not only of the positive literals (as in the deductive setting)

$$[\mathsf{Fr\text{-}pos}_{SAT}] \quad C \to [A]C \quad \textit{for} \quad C \in Conditions - (Del(A) \cup Add(A))$$

but also of the negative literals

$$[\mathsf{Fr\text{-}neg}_{SAT}] \quad \neg C \to [A]\neg C \quad \textit{for} \quad C \in Conditions - (Pre(A) \cup Add(A)).$$

The crucial difference w.r.t. the deductive approach consists in reducing the planning problem to testing the *satisfiability* of the formula

$$[\mathsf{Plan}_{SAT}] \quad Initial \wedge \langle Any^*\rangle Final$$

(or, equivalently, of its regressive variant

$$[\mathsf{Plan}^-{}_{SAT}] \quad Final \wedge \langle (Any^-)^*\rangle Initial.)$$

Therefore, a plan will be recovered from a *model* of the above formula. This requires practically no modification to an existing $\mathcal{ALC}^\star$ consistency testing algorithm since such algorithms work by constructing models. In our tests, we have used the $\mathcal{R}eg\mathcal{AL}$ system described in [8] for solving propositional STRIPS planning problems encoded as satisfiability testing.[15]

Note that the SAT-based approach requires a "completely specified" initial state, in which either C or $\neg C$ holds for each action precondition C[16]. If neither C nor $\neg C$ holds in state S, then there may exist anomalous models in which actions having C as a precondition are executed in S. Fortunately, a "completely specified" initial state entails "completely specified" intermediate states.

[15] An automated translation tool from STRIPS specifications to $\mathcal{ALC}^\star$ axioms has been implemented for this purpose. Then, the $\mathcal{ALC}^\star$ reasoning services are used for constructing a plan, i.e. a model of some formula.

[16] An incomplete initial state can be "completed" by adding a negated literal $\neg C$ for each unspecified action precondition C. This works since the condition sets $Pre(A)$ contain only positive literals.

4 Related work

4.1 The Frame Problem for Nondeterministic Actions

Craig Boutilier and Nir Friedman [10] try to solve the frame problem for non-deterministic actions in a monotonic setting, drawing inspiration from Reiter's explanation closure axioms developed for the deterministic setting [26].

They argue that some of the main intuitions underlying Reiter's solution must be abandoned in a nondeterministic setting due to the possible correlations among effects. They therefore use a much stronger *Process Logic* instead of the weaker Dynamic Logics (Process Logics are not only more expressive than Dynamic Logic, but they usually have a higher computational complexity – doubly exponential or worse, whereas Dynamic Logic is worst-case EXPTIME-complete). However we show in the following that the recourse to Process Logic is unnecessary, Dynamic Logic and $\mathcal{ALC}^\star$ being sufficient for their purposes.

Boutilier and Friedman deal with non-deterministic action specifications with multiple action clauses of the form

$$a \text{ causes } \rho^a_{11}||\ldots||\rho^a_{1k^a_1} \text{ when } D^a_1$$
$$\ldots$$
$$a \text{ causes } \rho^a_{n1}||\ldots||\rho^a_{nk^a_n} \text{ when } D^a_n$$

where the preconditions (discriminants) D^a_i are exhaustive $\bigvee_{i=1}^n D^a_i$ and pairwise exclusive $\neg(D^a_i \wedge D^a_j)$ for $i \neq j$. Each possible effect ρ^a_{ij} is a conjunction of literals. An action clause

$$a \text{ causes } \rho^a_{i1}||\ldots||\rho^a_{ik^a_i} \text{ when } D^a_1 \tag{4}$$

says essentially that if the current state s verifies the precondition D^a_i, then the action a is applicable in s with the possible effects ρ^a_{ij} for $j = 1,\ldots,k^a_i$ (if $k^a_i > 1$, the action is non-deterministic since there are several possible outcomes).

A different type of action clauses describing *necessary* effects of actions is also available:

$$a \text{ necessarily causes } \rho^a_{i,nec} \text{ when } D^a_i \tag{5}$$

Such an action theory (4,5) is interpreted in Dynamic Logic (and therefore also in $\mathcal{ALC}^\star$) as:

$$D^a_i \to \langle a\rangle\rho^a_{i1} \wedge \ldots \wedge \langle a\rangle\rho^a_{ik^a_i} \wedge [a]\rho^a_{i,nec} \tag{6}$$

However (6) deals only with the explicit effects of a. Frame axioms of the form

$$l \wedge \neg Poss(a,\neg l) \to [a]l \tag{7}$$

have to be added for all literals l, where $Poss(a, l)$ is the disjunction of all discriminants of action clauses having l in one of their possible effects lists

$$Poss(a,l) = \bigvee_{\substack{i \\ l \in \rho_{ij}^a \text{ for some } j}} D_i^a$$

In the case of deterministic action clauses, there is just one possible outcome in each action clause ($k_i^a = 1$), so $Poss(a, l)$ implies that l must be true after executing a and therefore Reiter's solution is applicable in a straightforward fashion in this case.

More precisely, the effect axioms (6) reduce, in the deterministic setting, to

$$D_i^a \to \langle a \rangle \rho_i^a \tag{8}$$

(assuming no "necessary" effects). Considering only those axioms for which $l \in \rho_i^a$ (for a given literal l), we have that

$$\langle a \rangle \rho_i^a \to \langle a \rangle l. \tag{9}$$

(8) and (9) entail

$$Poss(a,l) \to \langle a \rangle l. \tag{10}$$

The condition that a is deterministic amounts to

$$\langle a \rangle l \to [a] l \tag{11}$$

which can be combined with the completeness (exhaustiveness) condition[17]

$$\langle a \rangle true \tag{12}$$

to give

$$\langle a \rangle l \leftrightarrow [a] l. \tag{13}$$

(7), (10) and (13) entail the "$\leftarrow$" direction of the Reiter successor state axiom

$$\langle a \rangle l \leftrightarrow Poss(a,l) \vee (l \wedge \neg Poss(a, \neg l)). \tag{14}$$

The "$\to$"direction can be obtained from the "$\leftarrow$" direction for the negated literal $\neg l$

$$\langle a \rangle \neg l \leftarrow Poss(a, \neg l) \vee (\neg l \wedge \neg Poss(a,l))$$

which is equivalent with

$$[a] l \to \neg Poss(a, \neg l) \wedge (l \vee Poss(a,l))$$

i.e

$$[a] l \to (Poss(a,l) \wedge \neg Poss(a, \neg l)) \vee (l \wedge \neg Poss(a, \neg l)). \tag{15}$$

Since the effects of an action cannot be contradictory, we have that $\neg (Poss(a,l) \wedge Poss(a, \neg l))$, which entails

$$Poss(a,l) \wedge \neg Poss(a, \neg l) \leftrightarrow Poss(a,l). \tag{16}$$

Finally, (13),(15) and (16) entail the "$\to$" direction of (14).

[17] This condition imposed by Boutilier and Friedman is not justified in the general case: not every action a is applicable in every state – there may be conditions under which a is *not* applicable.

As can be easily seen from the above considerations, Reiter's solution (14) is not applicable in the case of nondeterministic actions, mainly because the frame axioms (7) are too weak (i.e. in a certain sense incomplete): they do not say anything about the persistence or nonpersistence of the literal l in states verifying $Poss(a, \neg l)$. Such states, however, do not necessarily lead to states verifying $\neg l$, they just might do so (since only some of the possible outcomes of action a lead to $\neg l$). Therefore we have to describe what happens to the literal l in all possible outcomes of a. For this purpose, Boutilier and Friedman use a Process Logic. However this may lead to important complexity blowups as well as to more complex and less understandable encodings. We argue that such a recourse to Process Logic is unnecessary: Dynamic Logic and $\mathcal{ALC}^\star$ would have been sufficient for their purposes.

The main problem with writing frame axioms for nondeterministic action clauses (4) (encoded as (6)) is that we have to describe the persistence of some literal l w.r.t. action a in a way that discriminates between the several possible outcomes $\langle a \rangle \rho_{i1}^a, \ldots, \langle a \rangle \rho_{ik_i^a}^a$ of the nondeterministic action. However, this discrimination is not possible in Boutilier and Friedman's approach because they use a formula like $[a]l$ to express the persistence of l (the formula $[a]l$ leads to the persistence of l in *all* possible outcomes $\langle a \rangle \rho_{i1}^a, \ldots, \langle a \rangle \rho_{ik_i^a}^a$). (Boutilier and Friedman use the preconditions D_i^a to discriminate between the possible successor states in case of *deterministic* actions.) In order to be able to discriminate between the possible outcomes of a nondeterministic action clause (4):

$$a \text{ causes } \rho_{i1}^a || \ldots || \rho_{ik_i^a}^a \text{ when } D_i^a$$

we shall encode each of the k_i^a different possible outcomes ρ_{ij}^a as the result of a different action name a_j

$$D_i^a \to \langle a_1 \rangle \rho_{i1}^a \wedge \ldots \wedge \langle a_{k_i^a} \rangle \rho_{ik_i^a}^a \wedge [a] \rho_{i,nec}^a \tag{17}$$

(compare (17) with Boutilier and Friedman's encoding (6)) and use the disjunction $a = a_1 \vee \ldots \vee a_{k_i^a}$ in queries and necessary effect axioms involving a.

The crucial benefit of this encoding relies in the possibility of writing frame axioms that discriminate between the possible outcomes $\langle a_j \rangle \rho_{ij}^a$ of action a:

$$l \wedge \neg Poss(a_j, \neg l) \to [a_j] l \tag{18}$$

given that (17) is equivalent with the conjunction of axioms of the form

$$D_i^a \to \langle a_j \rangle \rho_{ij}^a \wedge [a] \rho_{i,nec}^a$$

corresponding to the deterministic action clauses:

$$a_j \text{ causes } \rho_{ij}^a \text{ when } D_i^a.$$

Note that the frame axioms (18) can be rewritten as

$$l \wedge \neg D_i^a \to [a_j] l \quad \textit{if } \neg l \in \rho_{ij}^a \cup \rho_{i,nec}^a$$

$$l \to [a_j] l \quad \textit{if } \neg l \notin \rho_{ij}^a \cup \rho_{i,nec}^a$$

or, in our representation:

$$l \wedge \neg Pre(a) \rightarrow [a_j]l \quad for\ l \in Del(a_j)$$

$$l \rightarrow [a_j]l \quad for\ l \notin Del(a_j)$$

or even

$$l \rightarrow [a_j]l \quad for\ l \in Del(a_j) - Pre(a)$$

$$l \rightarrow [a_j]l \quad for\ l \notin Del(a_j)$$

i.e.

$$l \rightarrow [a_j]l \quad for\ l \in Conditions - (Del(a_j) \cap Pre(a))$$

which coincide with our representation of the frame axioms (because $Del(a_j) \subseteq Pre(a)$ in our representation).

The above considerations therefore show that Boutilier and Friedman's nondeterministic action clauses can be encoded in our causal (asymmetric) deductive approach in an even simpler fashion than they originally did (in $\mathcal{ALC}^\star$ rather than a more sophisticated Process Logic).

4.2 Deductive planning using dynamic logic

Dynamic logic has been used in the past to encode reasoning about actions and plans [27, 23], but a syntactical planning algorithm implemented on top of a Dynamic Logic theorem prover was usually employed. In the present paper we reduce planning to reasoning *within* a Description Logic, by using exclusively the DL reasoning services (without any additional external algorithms).

In [33], planning was reduced to proving theorems like $Initial \rightarrow \langle ?Plan \rangle Final$ in a tactical theorem prover for First Order Dynamic Logic (KIV). There, strategies like progression and regression were implemented by means of tactics of the theorem prover, which may be a too low level approach to the problem.

We argue that these approaches are inappropriate for Description Logics for at least two important reasons. First, Description Logics do not allow for variables in formulae, so the goal formulae above are not expressible in DLs. Secondly, reasoning in First-Order Dynamic Logic (FODL) is highly undecidable [22], thereby rendering any FODL theorem prover incomplete in principle.

Finally, strategies like regression and progression are implemented by means of low level tactics of the theorem prover, rather than at the conceptual level. Our approach addresses all the above problems successfully by using a general role like Any^* instead of role meta-variables, by providing sound and complete reasoning algorithms for solving the planning problem within the DL (and without any external algorithms) and finally by being able to encode strategies like progression and regression at the conceptual level.

4.3 The Robot-Tino Project

The Robot-Tino project at the University of Rome [17] represents a related approach to planning using DLs with a non-monotonic epistemic operator.[18]. The approach is incomplete w.r.t. the planning problem, but the incompleteness seems to be inessential, occurring only in artificial special cases. On the other hand, due to the increased expressivity of the (auto)epistemic operator, it is applicable in more general cases than our approach (constraints involving action preconditions are dealt with, albeit incompletely).

The main drawback seems to be the difficulty of reasoning in the non-monotonic logic associated to the (auto)epistemic operator. Since no such theorem prover has been implemented up to now, De Giacomo et al. use the procedural rules of CLASSIC, leading to a rather limited implementation.

Acknowledgments

Thanks are due to anonymous reviewers who pointed out the related work [17].

References

[1] Artale A., Franconi E. *A computational account for a description logic of time and action.* Proc. KR-94, 3-14.

[2] Baader F. *Augmenting concept languages by the transitive closure: An alternative to terminological cycles.* IJCAI-91, pp. 446-451, also DFKI RR-90-13.

[3] Baader F., Hollunder B. *KRIS: Knowledge Representation and Inference System – System Description.* DFKI TM-90-03.

[4] Baader F., Hollunder B. *Embedding Defaults into Terminological Knowledge Representation Formalisms.* Proc. KR-92, 306-317.

[5] Baader F., Laux A. *Terminological Logics with Modal Operators.* IJCAI-95, pp. 808-814.

[6] Badea Liviu. *A unitary theory and architecture for knowledge representation and reasoning* (in Romanian) PhD Thesis, November 1994.

[7] Badea Liviu. *A unified architecture for knowledge representation and reasoning based on terminological logics.* International Workshop on Description Logics, Roma 1995.

[8] Badea Liviu. *A unified architecture for knowledge representation based on description logics.* Proc. ECAI-96, 288-292.

[9] Badea Liviu. *Reifying Concepts in Description Logics.* Proc. IJCAI-97.

[10] Boutilier C., Friedman N. *Nondeterministic Actions and the Frame Problem.* AAAI Spring Symposium on Extending Theories of Action, Stanford, March 1995.

[11] Brachman R.J., Schmolze J.G. *An Overview of the KL-ONE Knowledge Representation System.* Cognitive Science 9 (2) 1985.

[12] Bylander T. *Complexity results for planning.* IJCAI-91, 274-279.

[18] The necessity of dealing with state completions in our encodings also seems to imply that a non-monotonic approach is needed in a more general setting.

[13] DE GIACOMO G., LENZERINI M. *Concept Language with Number Restrictions and Fixpoints, and its Relationship with the Mu-calculus.* Proc. ECAI-94, 411-415.

[14] DE GIACOMO G., LENZERINI M. *Boosting the correspondence between description logics and propositional dynamic logics.* Proc. AAAI-94, 205-212.

[15] DE GIACOMO G., LENZERINI M. *What's in an Aggregate: Foundations for Description Logics with Tuples and Sets.* IJCAI-95, pp. 801-807.

[16] DE GIACOMO G., LENZERINI M. *Enhanced Propositional Dynamic Logic for Reasoning about Concurrent Actions.* Proc. AAAI Spring Symposium, 1995.

[17] DE GIACOMO G., IOCCHI L., NARDI D., ROSATI R. *Moving a Robot: The KR&R Approach at Work.* Proc. KR'96, pp. 198-209.

[18] DEVANBU P.T., LITMAN D.J. *Plan-based terminological reasoning.* Proc. KR-91, 128-138.

[19] DONINI F.M., LENZERINI M., NARDI D., SCHAERF A., NUTT W. *Adding Epistemic Operators to Concept Languages.* Proceedings KR-92, Boston.

[20] DONINI F.M., LENZERINI M., NARDI D., SCHAERF A., NUTT W. *The Complexity of Concept Languages.* Proceedings KR-91, Boston.

[21] FISCHER M.J., LADNER R.E. *Propositional Dynamic Logic of Regular Programs.* Journal of Computer and System Science 18, pp.194-211, 1979.

[22] HAREL D. *Dynamic Logic.* In Gabbay D., Guenther F. (eds) *Handbook of Philosophical Logic*, Vol. 2, pp. 497-604, Reidel Dordrecht, 1984.

[23] KAUTZ H. *A first-order dynamic logic for planning.* Technical Report CSRG-144, University of Toronto, May 1982.

[24] KAUTZ H., SELMAN B. *Pushing the envelope: planning, propositional logic, and stochastic search.* Proc. AAAI-96.

[25] LEVY A., ROUSSET M.C. *CARIN: A Representation Language Combining Horn Rules and Description Logics.* Proc. ECAI-96, 323-327.

[26] REITER R. *The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression.* In V. Lifshitz (ed) Artificial Intelligence and Mathematical Theory of Computation (Papers in Honor of John McCarthy), pp. 359-380, 1991.

[27] ROSENSCHEIN S.J. *Plan synthesis: a logical approach.* Proc. IJCAI-81.

[28] SCHILD KLAUS. *Terminological Cycles and the Propositional μ-Calculus.* DFKI Research Report RR-93-18, 1993.

[29] SCHILD KLAUS. *Combining terminological logics with tense logic.* Proc. EPIA'93.

[30] SCHILD KLAUS. *A correspondence theory for terminological logics: preliminary report.* IJCAI-91.

[31] SCHMIEDEL A. *A temporal terminological logic.* AAAI-90, 640-645.

[32] SCHMIDT-SCHAUSS M., SMOLKA G. *Attributive concept descriptions with complements.* Artificial Intelligence 48 (1), pp. 1-26, 1991.

[33] STEPHAN W., BIUNDO S. *A New Logical Framework for Deductive Planning.* Proc. IJCAI-93, 32-38.

[34] STREETT R.S. *Fixpoints and Program Looping: Reductions from the Propositional Mu-Calculus into Propositional Dynamic Logic of Looping.* In R. Parikh (ed), *Proceedings of the Workshop on Logic of Programs,* Lecture Notes in Computer Science, Vol. 193, pp. 359-372, Springer Verlag, Berlin, 1985.