Action Hierarchies in Description Logics

Thorsten Liebig and **Dietmar Rösner**
Otto-von-Guericke-Universität Magdeburg
Institut für Informations- und Kommunikationssysteme
P.O. Box 41 20, D–39016 Magdeburg, Germany
{tliebig|roesner}@iik.cs.uni-magdeburg.de

Abstract

Our project aims at the automatic generation of multilingual text for product maintenance and documentation from a structured knowledge representation formalized by means of plans in a description logic. These plans consist of one or more actions or other plans. This language independent representation is also designed to support the qualitative simulation of the documentation steps and to support the generation of multilingual text by offering action descriptions at different levels of detail for verbalization. This requires a hierarchical organization of performable action descriptions capable to define actions by inheriting already existing actions. Prerequisite for inheritance of pre- and postconditions of actions in a KL-ONE system is the definition of a proper subsumption relation for this logical expressions, conform with the subsumption for ordinary KL-ONE constructs. We propose a framework for extending the LOOM system with syntax and semantics for representing and reasoning with action hierarchies.

1 Introduction

The TECHDOC project[1] [Stede and Rösner, 1994] aims to automatically generating at least parts of multilingual technical documentation from a language independent representation of form and content. The generation is based on language and product specific knowledge, as well as on general technical domain knowledge, represented in a description logic.

This process has to cope with key requirements on technical documentation, namely conformity between product and documentation, a consistent terminology and the tailoring to different addressees and purposes.

The underlying knowledge structure of instruction and maintenance texts is formalized by means of *plans*. These plans consist of one or more *actions* or other plans. Actions are single steps of a maintenance or instruction plan, building a reasonable unit. They are not further decomposable. Typically these actions can be expressed in one single sentence.

Our representation approach for actions is designed to provide the *qualitative simulation* of the instruction steps and to support the *generation of multilingual text* by offering action descriptions at different levels of detail for verbalization. This requires the declarative representation of performable action descriptions in a hierarchical way.

We are using the hybrid KL-ONE–like [Brachman and Schmolze, 1985] knowledge representation system LOOM [MacGregor and Bates, 1987] for generating natural-language with the sentence generator PENMAN [Mann, 1983] as well as for representing knowledge about the technical products [Liebig and Rösner, 1996]. Our aim is to find a theoretical framework for representing actions in description logics in a hierarchical way, conform with the representation of ordinary KL-ONE constructs (e. g. concepts).

In the following section we describe two different approaches for representing actions. Based on their analysis we then postulate the representation requirements suitable for our task. In section 2.4 we present an extended framework of description logics which is able to provide representation and reasoning facilities for our kind of action descriptions.

2 Action Descriptions

There are many different approaches for representing actions in the field of natural language processing and traditional planning. Due to their varying requirements they have different advantages and disadvantages.

Natural Language Processing There exist hierarchically organized action descriptions in systems for natural language generation (e. g. PENMAN Upper Model [Bateman *et al.*, 1990]). These descriptions focus mainly on the verb as the relevant object for classification. An

[1] The TECHDOC project is in progress at the FAW Ulm (Research Institute for Applied Knowledge Processing, Ulm, Germany) since mid 1991. Most of the work is now continued at the "Knowledge–Based Systems and Document Processing Group" at the University of Magdeburg, Germany.

advantage of hierarchical action descriptions in a classification system is the automated inheritance of properties. For instance the roles *actor* and *actee* are inherited by all action descriptions modelled as descendants of the concept *action* in the PENMAN Upper Modell. Action descriptions of this kind are well suited for natural language processing, but they are not sufficient for planning or simulated execution.

Planning Many action descriptions in AI are related to the field of planning. Actions are therefore interpreted as operators, mapping one world description onto another. In general they are based on first order predicate logic and characterized by *pre-* and *postconditions*. The precondition is a set of conditions which necessarily have to hold in order to execute the action. The set of facts which holds after the action has been performed is called *postcondition*. As a result of their operational description, actions can be executed for planning or simulation purposes. But up to now there does not exist a satisfying approach for defining actions by refining an existing action. From the taxonomical point of view this leads to action descriptions without any hierarchical organization.

2.1 Requirements and Approach

Already implemented systems have shown that it is possible to build action hierarchies in description logics according to classification criteria, like those used for natural language generation. But there does not exists a similar framework for the representation of performable actions, which supports the specialization of actions and the inheritance of pre- and postconditions.

However, our task demands for an action description which can be analysed by additional system components, e. g. for the generation of natural language, and can be executed for the simulation of action sequences. In order to support both, we propose a declarative action description represented in a description logic which

- is *executable* (operational),
- allows *action* definitions by *specialization*,
- supports the *inheritance of pre- and postconditions* similar to other concept-forming expressions and
- results in a *hierarchical organization* of action descriptions.

The specialization of actions is needed to define a particular action from a more general one. Figure 1 shows a part of a potential action hierarchy. There, the action `increase-temperature` is defined as a specialization of the action `change-temperature` and therefore inherits all properties (participant restrictions, pre- and postcondition, etc.) of its ancestor. `increase-temperature` is more specific in comparison with `change-temperature` because former action changes the temperature in a particular way. `change-temperature` is again a specialization of the more general action `change` in the abstraction hierarchy.

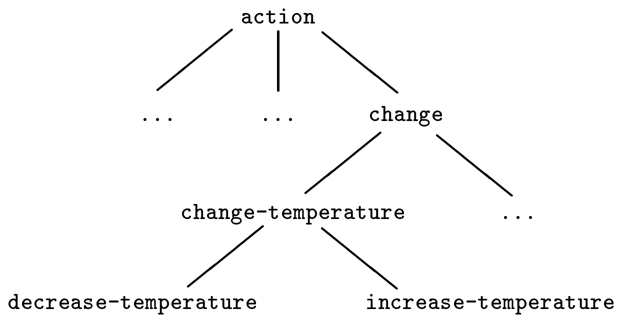

Figure 1: Part of a possible action hierarchy

An operational representation of actions enables to perform the *qualitativesimulation* of the instruction steps, which is required in order to check for completeness and applicability of a plan, or to derive hints and warnings automatically. Consider, for example, the consequences of the preparatory step 'warming up the engine' of a car for the maintenance process of changing oil: All parts and fluids directly connected to the engine become hot. This implicit knowledge can be inferred by the reasoning system because of the physical model of the represented product. This can result in automatically generated warnings whenever hot objects (e. g. the oil) will be involved in future instruction steps. This example illustrates that for those kinds of reasoning tasks, one will need an expressive domain representation based on a model of physical structuring as well as on dynamically changing qualities and status properties (see [Liebig and Rösner, 1996]).

In short, we propose a declarative action description which can be executed (for the qualitative simulation of the instruction steps) and which supports action specializations and inheritance of pre- and postconditions in order to enrich text generation with auxiliary product and context specific knowledge.

2.2 Benefits

Formalizing the previously proposed action representation in a description logic has the following advantages:

- Actions can be defined clear and efficient manner using inheritance mechanisms. The definition of an action consists mainly of its ancestor and the statements which characterize the intended refinement.
- Actions can be structured by their hierarchical ordering. More general actions are located near the root; the most specific actions are located at the leaves. This allows more efficient search mechanisms as in linear orderings.
- It is also conceivable to use different parts of the action description as filtering criteria for the process of plan creation. A documentation workbench could then support a technical editor by offering sets of

actions suitable for different documentation paragraphs (installation, handling, cleaning etc.).

- The inference services of the representation system support the modelling of actions by recognizing incoherent or semantically equivalent action descriptions.
- The language generation component can choose between different verbalizations by varying the degree of abstraction of the action under consideration. While processing the action `increase-temperature` from figure 1, the sentence planner can, for example, choose between a vague expression like "change temperature to" or more concrete "increase temperature to".

2.3 Hierarchical Action Descriptions

The representation of actions in inheritance systems has been partly addressed by several projects.

A system for the representation of actions and plans in a description logic is RAT (representation of actions using terminological logics) [Heinsohn *et al.*, 1992]. They distinguish between *atomic actions*, which are non-decomposable and characterized by their pre- and postconditions, and *plan schemata*, which represent sequences of actions with possible constraints on the involved objects. Pre- and postconditions of atomic actions can be described using a subset of the underlying description logic: conjunctions of feature restrictions, agreements, and disagreements. The most important service offered by RAT is the *simulated execution* of a plan, checking if a given plan is feasible and, if so, computing the global pre- and postconditions.

However, RAT does not allow the specialization of actions. It is not possible to define similar actions as special cases of a more abstract action. Every new action definition has to contain all action parameters, pre- and postconditions explicitly. This can lead to a time intensive and error prone modelling process.

In contrast to the RAT sytem, actions in CLASP [Devanbu and Litman, 1991] as well as in T-REX [Weida and Litman, 1992] are primitive non-decomposable units. Instead, their language for composing plans is much more richer. Common to these representations is their restricted language for expressing action descriptions. The reason for this limitation are the inference mechanisms of these systems, primarily developed for plan recognition tasks. Action specialization is not possible in any of these systems.

2.4 Extensions for Building Action Hierarchies

In this section we present our extensions to description logics for building action hierarchies which meet the previously mentioned requirements. Our approach carries on the theoretical framework of C. Kemke [Kemke, 1988] for representing actions in KL-ONE like systems.

We use a subset of first order predicate logic to describe pre- and postconditions. Fundamental for including these conditions into a KL-ONE system is the definition of a proper subsumption relation for these logical expressions, conform with the subsumption for ordinary KL-ONE constructs.

In a description logic a concept $P1$ subsumes a concept $P2$ (written $P2 \sqsubseteq P1$) iff all individuals of every semantical interpretation of $P2$ are also individuals of $P1$. Subsumption can therefore also be expressed in terms of implication: $P1$ subsumes $P2$ iff for every individual of concept $P2$ implicitly follows a membership for $P1$. On the basis of implication, Kemke defined three subsumption rules, which are theorems in first order predicate logic:

The expressions α_1 and α_2 are either a pre- or postcondition. When defining an action $P2$ inheriting an already defined action $P2$, the following leads to a subsumption in the form of $P2 \sqsubseteq P1$:

i. adding a new condition α_2 to the inherited condition α_1, because $\alpha_1 \wedge \alpha_2 \rightarrow \alpha_1$ ($\wedge$–rule),

ii. deleting an inherited condition α_2, because $\alpha_1 \rightarrow \alpha_1 \vee \alpha_2$ ($\vee$–rule),

iii. expanding an inherited condition α_1 to all individuals of the universe, because $\forall x : \alpha \rightarrow \exists x : \alpha$ (quantifier–rule).

2.5 Subsumption between Actions

The definition given above is only one possible realization of a subsumption relation between actions. Therefore, this definition cannot reflect other intended subsumption relationships based on action conditions. However, one can think of equally reasonable notions of subsumption. As mentioned in [Heinsohn *et al.*, 1992], the notion of subsumption between actions and plans is far less obvious than subsumption between concepts. The following three different subsumptions between actions (and plans) are proposed.

Abstraction-Subsumption: Abstraction-subsumption means that an action $A1$ subsumes another action $A2$ iff $A1$'s pre- and postconditions are more general than those of $A2$. Kemke's definition conforms to this kind of subsumption, as well as Devanbu and Litmans notion, called "p-subsumption" in [Devanbu and Litman, 1991].

Goal-Subsumption: This relation organizes actions simply by considering the subsumption ordering of the goal description. This is useful, if a library is searched for an action which secures the achievement of a given goal. Devanbu and Litman call this "g-subsumption".

Applicability-Subsumption: Another way to order actions is the applicability-subsumption, where an action $A1$ subsumes an action $A2$ iff $A1$ is always applicable when $A2$ is. In this case, it is necessary that $A1$ has a weaker precondition and a stronger postcondition than $A2$.

2.6 Syntax

The conditions contained in pre- and postconditions mostly consist of agreements between feature chains, denoted in LOOM by the keyword `:SAME-AS`. Feature chains are composed feature atoms, which are in general a set of ordered roles forming a role chain between two concepts. We will restrict our language to conditions of feature chain agreements. A role chain can therefore be specified by one or more roles:

```
<role-chain> ::=
   RelationName |
   (RelationName | role-chain) ;
```

The syntax for pre- and postconditions is then as follows:

```
<condition-expr> ::=
   (:AND condition-expr+) |
   (:SAME-AS role-chain role-chain) |
   (:SOME role-chain ConceptName) |
   (:ALL role-chain ConceptName) |
   (:OR condition-expr+) ;
```

Additional syntax is needed for the extension of concepts in order to define actions. Since we are using the knowledge representation system LOOM, we propose the following additional grammar for the `<concept-clause>` (for the original definition see [Brill, 1995] p. 305):

```
<concept-clause> ::=
   ...
   :attribute RelationName |
   :precondition (<condition-expr>) |
   :postcondition (<condition-expr>) ;
```

The actions `change` and `change-temperature` could then be formulated as follows:

```
(defconcept change
   :is (:and action
            (:the has-object object)
            (:the has-new-value value)
            (:the has-old-value value))
   :attribute has-attribute
   :precondition
      ((:same-as (has-object ?attribute)
                 has-old-value))
   :postcondition
      ((:same-as (has-object ?attribute)
                 has-new-value)))

(defconcept change-temperature
   :is (:and change
            (:the has-new-value temperature)
            (:the has-old-value temperature))
   :attribute has-temperature)
```

The variable `?attribute` is bound to the role specified (or inherited) in `:attribute` and is therefore accessible in pre- and postcondition. Given the inheritance mechanisms, the action `change-temperature` has the following explicitly expanded definition:

```
(defconcept change-temperature
   :is (:and change
            (:the has-object object)
            (:the has-new-value temperature)
            (:the has-old-value temperature))
   :attribute has-temperature
   :precondition
```

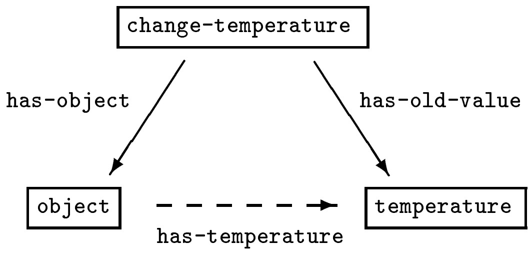

Figure 2: The precondition of action `change-temperature` graphically. Boxes represent actions and concepts; arrows represent roles.

```
      ((:same-as (has-object has-temperature)
                 has-old-value))
   :postcondition
      ((:same-as (has-object has-temperature)
                 has-new-value)))
```

Figure 2 presents the inherited precondition of the action `change-temperature`. This condition claims the agreement between the feature chain `has-object` / `has-temperature` and `has-old-value` (see definition above).

2.7 Semantics

What is the semantics for the introduced syntax? The `:attribute` keyword has a very special meaning: It is only used to store the specified or inherited role name in order to bind the `?attribute` variable. It should be noted, that the role (`has-temperature`) specified with help of the `:attribute` keyword is not a role of the action concept (`change-temperature`) but of the action object (see dashed arrow in figure 2).[2]

In a description logic with a role hierarchy (like LOOM), this hierarchy has to be taken into account while computing subsumption between actions. Overwriting an inherited attribute role should not conflict with the existing role hierarchy. This implies that the role `has-attribute` has to subsume `has-temperature` from above.

For determination of subsumption between actions, one should choose a subsumption relation between pre- and postconditions. C. Kemke. Obviously, the dynamic binding of the `?attribute` variable has to be taken into account for this purpose.

In contrast to ordinary statements in concept definitions, pre- and postconditions do operate on the knowledge base at the time the action is performed. For the preconditions a mapping has to be found into syntactic and semantic correct queries of the host knowledge representation system, in order to check its truth-value. Correspondingly postconditions need to be mapped into expressions adding new facts to the knowledge base. For example, the preconditions of the action `change-temperature`:

[2] Kemke's framework was insufficient at this point because it modelled the attribute role as a role of the action concept.

```
((:same-as (has-object has-temperature)
           has-old-value))
```

should be translated into a Loom ask-statement of the form:

```
(ask (:same-as (has-temperature (has-object ?self))
               (has-old-value ?self)))
```

The variable ?self has to be bound to the current action instance in this query.

3 Conclusion and Open Problems

We were searching for a description logic supporting the modelling of actions in a way similar to ordinary concept reasoning. Unfortunately, this kind of reasoning is not realized in current KL-ONE based knowledge representation formalisms. Therefore, we have extended the framework of description logics with syntax and semantics for a subsumption relation between action descriptions. This allows to build action ontologies.

However, the proposed framework for representing action hierarchies in a description logic leaves some open problems.

We have presented three semantic rules for extending the determination of subsumption between action descriptions. However, the proposed syntax does not yet supply the deletion of alternative conditions from pre- or postconditions. At present, an additional condition statement complies to adding a new or expanding an existing condition. Additional work has to be done to provide syntax for deleting an already inherited condition.

Beyond that, it is not inherently clear which proposed notion of subsumption should be integrated into the system. Different notions of subsumption are apparently suitable for different tasks (see [Heinsohn *et al.*, 1992]).

Another important question is how these extensions influence the complexity of the underlying knowledge representation system. However, we assume that the proposed enhancements do not affect the algorithmic properties of any of the reasoning services supported by the host system.

References

[Bateman *et al.*, 1990] John A. Bateman, Robert T. Kasper, Johanna D. Moore, and Richard A. Whitney. A general organization of knowledge for natural language processing: the penman upper model. Technical report, ISI, 1990.

[Brachman and Schmolze, 1985] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171 – 216, April 1985.

[Brill, 1995] David Brill. *Loom Reference Manual.* University of Southern California, version 2.0 edition, November 1995.

[Devanbu and Litman, 1991] Premkumar T. Devanbu and Diane J. Litman. Plan-based terminological reasoning. In J. F. Doyle, R. Files, and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning, Proceedings of the Second International Conference (KR '91)*, pages 128 – 138, Cambridge, MA, April 1991. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

[Heinsohn *et al.*, 1992] Jochen Heinsohn, Daniel Kudenko, Bernhard Nebel, and Hans-Jürgen Profitlich. RAT: representation of actions using terminological logics. Technical report, DFKI, Saarbr"ucken, November 1992.

[Kemke, 1988] Christel Kemke. Darstellung von Aktionen in Vererbungshierarchien. In Wolfgang Hoeppner, editor, *Proceedings der 12. Jahrestagung Künstliche Intelligenz (GWAI-88)*, pages 66 –76, Berlin, 1988. Springer–Verlag.

[Liebig and Rösner, 1996] Thorsten Liebig and Dietmar Rösner. Description logic as core machinery for the automatic generation of multilingual technical documents. In Lin Padgham, Enrico Franconi, Manfred Gehrke, Deborah L. McGuinness, and Peter F. Patel-Schneider, editors, *Collected Papers of the 1996 International Description Logics Workshop*, pages 160 – 164, Cambridge, Massachusetts, November 2–4 1996. American Association for Artificial Intelligence, AAAI Press.

[MacGregor and Bates, 1987] R. MacGregor and R. Bates. The LOOM knowledge representation language. Technical report, ISI/RS-87-188, ISI, University of Southern California, 1987.

[Mann, 1983] William C. Mann. An overview of the PENMAN text generation system. In *Proceedings of the National Conference on Artificial Intelligence*, pages 261 – 265. AAAI, August 1983.

[Stede and Rösner, 1994] Manfred Stede and Dietmar Rösner. Generating multilingual documents from a knowledge base: The TECHDOC project. In *COLING-94, Proceedings*, Kyoto, 1994.

[Weida and Litman, 1992] R. Weida and D. Litman. Terminological reasoning with constraint networks and an application to plan recognition. In B. Nebel, W. Swartout, and Rich, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference*, Cambridge, MA, October 1992. Morgan Kaufmann Publishers Inc.