POSITION PAPER
Some Requirements for Practical Modeling in Dialogue Systems

Michael Knorr, Bernd Ludwig, Günther Görz
Computer Science Institute and FORWISS, Erlangen, Germany
Email: goerz@informatik.uni-erlangen.de

Abstract

The goal of this paper is to stimulate a discussion between description logic system developers and users of description logic systems with an emphasis on applications in the field of dialogue systems. It consists of two parts: The first part is a report on experiences in temporal representation and calendrical reasoning with description logics from which we derive requirements on decription logics as a useful modelling tool.

The second part which is more speculative in nature complements these considerations with further modelling tasks and corresponding expressive means modelling languages should meet to deal with them in a satisfactory way.

1 Experiences from using Description Logic for Time Representation

1.1 Introduction

Our research group has been involved in the development of speech dialogue systems for many years. These include EVAR [8], a system for train information, and EMBASSI[1] an application for interacting with home entertainment systems. A common feature of these and many other domains is that the exchange of temporal information is an essential part of the dialogue, be it the departure time of a train or the start time of a TV programme. Therefore we decided to investigate the syntax and semantics of German temporal expressions independently of a specific domain.

[1]See the contribution by Bücher et al. in this volume

Nearly all working speech dialogue systems, including the aforementioned EVAR, model dialogues with some kind of finite state machine. An evaluation of dialogues from the EVAR system showed, that such systems lack the flexibility to deal with "meta-dialogues", like corrections or changes in the dialogue goal. As a consequence we decided that the understanding of utterances should rely on well-defined inference procedures to reason about the meaning of utterances and the dynamic evolution of the dialogue structure. In order to obtain the semantics of natural language utterances one should rely on a domain model, defining the actions and objects salient to the application. We decided to use Description Logics (DL) for the construction of the domain model[18].

This leads to the need to represent the semantics of temporal expressions in Description Logics. When we started this work in 1999 we investigated the suitability of four DL-systems, namely CICLOP [17], CLASSIC [19], FaCT [13], and RACE [10] for the representation of temporal information. At this time only CLASSIC fulfilled our requirements. In the following sections we describe the representation of temporal information in our application and discuss what requirements for DL systems arise from it. Implementors of DL-systems might use them to consider what features their systems need to be useful for this domain. Users wishing to employ DL-systems for tasks similar to ours, might consider how far similar requirements apply to their own applications.

1.2 Representing Temporal Information

In this section we give a short description of our semantic model for temporal expressions. The words in our lexicon correspond to concepts in the T-Box of the DL system. When an utterance is parsed an individual is created in the A-Box of the DL system for every word encountered in the expression. The application of a grammar rule leads to either

- the creation of a new individiual, representing the whole phrase, with the individuals representing its parts used as role fillers.
- the selection of a specific individual and its connection to the other individuals in the expression by roles.
- or the mergence of the individuals representing the parts of the expression.

In many cases this triggers (forward-chaining) rules to move information or perform calculations whose results are added to the appropriate individuals.

The top concepts of our concept hierarchy are `TimePoint`, `TimeInterval`, `Duration` and `TimeUnit`. Most temporal expressions describe a point in time. They are represented by merging individuals of `Clocktime` (defining minute and hour), `PartOfDay`, `Date` (defining day, month and year) and `ISO-Date` (defining day of week and week). These concepts are therefore subsumed by `TimePoint`. `Duration` represents the length of an interval and the distance between two time points; intervals are represented by `TimeInterval`. The individuals representing time units, such as minute, hour, day, are of concept `TimeUnit`. Details of the representation can be found in [14].

1.3 Examples of Temporal Expressions

We now show in detail how the representation for a temporal expression is constructed. As example we take: *am siebzehnten dritten zweitausend um dreiviertel vier nachmittags* (on the 17 March 2000 at quarter to four in the afternoon). This expressions consists of three chunks, that could appear in any order. The semantics of the chunks can also be constructed and merged in arbitrary order.

The construction of the semantic representation starts when the word *siebzehnten* is parsed. An individual of concept `Date` is constructed and the attribute `has-day` is filled with 17. In the next steps the attributes `has-month` and `has-year` are set to 3 and 2000. Now a test function (cf. [5]) checks that 17 March 2000 is indeed a valid date in the Gregorian Calendar. Additionally a rule is triggered and calculates the day of week and the ISO week number. If later a day of week would be added which is different from the calculated one, the system would detect that the expression is inconsistent.

The individual now contains the following information:

```
(AND TimePoint (FILLS has-year 2000)
               (FILLS has-month 3)
               (FILLS has-week 11)
               (FILLS has-dow 5)
               (FILLS has-day 17))
```

The representation of the clocktime phrase starts with the construction of the individuals for *dreiviertel* and *vier*. *Dreiviertel* is represented as `SpecialMinutesFrom` with `has-timedistance` 15 Minutes and `has-direction` -1. *Vier* is mapped onto `TraditionalClocktime` with `has-traditional-hour` 4 and `has-minute` 0. A rule then derives that `has-hour` is `(ONE-OF 4 16)`. To represent the combination of *dreiviertel* and *vier*, the individual for the latter is filled in the `has-basetime`-attribute of the former. This triggers more information processing rules. The resulting representation is:

```
(AND SpecialMinutesFrom (ALL has-hour (ONE-OF 3 15))
                        (FILLS has-traditional-hour 3)
                        (FILLS has-minute 45)
                        (FILLS has-timedistance 15)
                        (FILLS has-direction -1)
```

```
                         (FILLS has-basetime OtherIndividual)
                         (SAME-AS (has-basevalue)
                             (has-basetime has-traditional-hour)))
```

With OtherIndividual:

```
(AND TwentyfourClocktime TraditionalClocktime (ALL has-hour (ONE-OF 4 16))
                         (FILLS has-traditional-hour 4)
                         (FILLS has-minute 0))
```

The adverb *nachmittags* (in the afternoon) leads to the creation of an individual of concept `Afternoon`. This places the constraint `(ALL has-hour (MIN 12)(MAX 18))` on the `has-hour`-role.

When the individual chunks are processed the individuals representing them are merged. The system can now infer that the hour is 15. The individual representing the complete expression is (only an extract is displayed):

```
(AND SpecialMinutesFrom (FILLS has-year 2000)
                        (FILLS has-month 3)
                        (FILLS has-week 11)
                        (FILLS has-day 17)
                        (FILLS has-dow 5)
                        (FILLS has-hour 15)
                        (FILLS has-minute 45)
                        (FILLS has-traditional-hour 3)
                        (FILLS has-basetime OtherIndividual))
```

This is still a fairly simple expression in terms of processing. Some very complex expressions look rather peaceful on the surface, consider *drei Tage vor Christi Himmelfahrt* (three days before Ascension Day). First we need to know the date of Ascension Day in a given year, than we need an algorithm that subtracts three days from it, taking care of month boundaries. To get the date of the holiday we either have to access an external calendar or add 39 days to the date of Easter. There are a number of algorithms to calculate the date of Easter, but they are all mathematically complex. We solved the problem by using an external library for calendrical calculations[7].

1.4 Requirements for a Description Logic for Time Representation

From our experience there are three main requirements for a DL-system to be used for time representation. First it needs to be able to deal with numbers. All information neccessary to represent time can be given using natural numbers.

For some expressions it would be useful to be able to use fractions (think of *quarter to* for example). In CLASSIC it is possible, to integrate objects of the host language, in this case LISP, in the Description Logic. Such host concepts can for example be strings or numbers. We used a number of host concepts based on LISP numbers.

Second we do not only want to represent natural numbers but also restrict their range and use them in simple calculations. Individuals of the concept `hour`, for instance, can be values between 0 and 24. It does not matter, from the applicational point of view, if the range check is applied on roles or concepts. An example for a simple calculation occurs, when we encounter *dreiviertel vier*. Here we have to set the role `has-hour` to 3, decrementing 4 by 1.

Third we either need external function calls or we have to be able to do more complex calculations within the DL-system. This starts with rather simple examples like testing if a date is valid and ends with very complex calculations for example to find the date of Easter in a given year. As mentioned above, test concepts as they have been introduced in CLASSIC, provide an appropriate extensible interface for reasoning with individuals. Intensional reasoning however requires an extension of subsumption for test concepts.

In order to perform reasoning with time intervals, there are two possibilities in principle, for both of which we are not aware of an operational implementation: Temporal Description Logics like $\mathcal{TL} - \mathcal{ALCF}$ (cf. e.g. [3]) or a temporal concrete domain based on Allen's interval algebra [2] as proposed e.g. by Kullmann [15].

A description logic allowing the use of concrete domains is $\mathcal{ALC}(\mathcal{D})$[4]. A well known problem of this logic is that it can become undecidable when transitive roles and role hierarchies are allowed (cf. e.g. [12]). However none of these are necessary in our application. According to [11] the RACER-system will soon offer $\mathcal{ALCQHI}_{R+}(\mathcal{D})^{-}$. A constraint of this logic is, that it does not allow to use feature chains. We have not investigated yet if we can dispense with them in all cases; probably not for simulating feature unification (sse below).

With respect to the use of concrete domains, an analogous situation exists with spatial reasoning, for which there is also a need in a variety of applications. Considerable research work has been done in this field, cf. [9], to quote at least one prominent example. We had the opportunity to try out an experimental implementation done by the Hamburg group, based on CLASSIC. In a project in the field of historical cartography, modelling the Behaim Globe of 1492, we implemented an extension to CLASSIC, based on test concepts, for topological and directional reasoning [6]. However, a spatial reasoning service in a more expressive DL system like RACE is an important desideratum.

2 Modelling Tasks and Further Requirements for Expressive Means of Modelling Languages

2.1 Access and Debugging

A Description Logic system that is used in real applications should also fulfill two further requirements. First, the knowledge assembled in the A-Box of the DL system should be easily accessible from the outside. We might, for example, wish to use the temporal information we collected in the system in a database query. For this we need to access the role-fillers of the individual containing the information. In CLASSIC this can be accomplished easily with the function `(cl-fillers @i{Individualname} 'Rolename)`.

Second, a system should contain functions that allow easy debugging of the knowledge base. In a complex knowledge base it is not always easy to see where errors in the inference process arise. It is therefore useful to have functions that print traces of the inference process and to be able to access the full information collected for an A-Box individual at intermediate states of its assembly. In CLASSIC this is facilated by the function `(cl-ind-expr @i{Individualname})`.

2.2 Simulating Feature Structure Unification

In parsing with unification grammars constraints are expressed as path equations. Instead of representing feature structures in a separate formalism, they can as well be expressed in DL. As a small experiment using CLASSIC showed, the unification of feature structures can then be achieved by means of the **same-as** construct. The experiment however was too small for a comparison of performance times.

In the context of a complex computational linguistics application[2], of course one might debate whether the use of a uniform representation formalism ranging over several processing levels is preferable as opposed to level-specific streamlined representation formalisms. In the latter case, processing might be more efficient, but at the cost of translating the resulting representations between levels. In our particular case, on the semantic, application and discourse pragmatic levels, DL is used anyway. So, it makes sense to consider whether the genuine feature structure representation on the syntactic level could be replaced by a DL representation, in particular, because syntactic analysis and semantic construction with linguistic chunks are closely interlocked in an incremental fashion.

[2]cf. our contribution on a dialogue system application in this volume

2.3 A-Box Reasoning

Resolution of ambiguities, in particular on the semantic and pragmatic levels as well as the completion of partial information – e.g. by means of clarification subdialogues – is a central task in dialogue systems. The linguistic analysis module of the dialogue system constructs semantic representations in the form of Discourse Representation structures (DRSs) which are then being refined to application specific DRSs using a formal domain ontology. These instantiated DRSs are the A-Box items on which various reasoning steps must be applied to solve the mentioned resolution and completion tasks. This means that beyond the standard DL reasoning services application domain specific rules are to be applied. Whereas some cases may be covered by forward-chaining rules as e.g. provided by CLASSIC, the general case is to complete subgoals generated by general dialogue goals. In other words, we need additional expressive means for dialogue (step) planning. From an abstract point of view, a language like Reiter's GOLOG language, which has been implemented on top of PROLOG, would provide sufficient means for this purpose. This in turn would lead to the requirement of an A-Box reasoning facility on the basis of Horn rules (cf. CARIN [16]).

As a framework for processing partial information, we found out that FIL [1] meets all our requirements. We started with the implementation of a prover for a Horn clause subset of FIL in Prolog technology, which has later been replaced by a tableau-based reasoner, operating as a separate module. Whether such a service can be integrated as a kind of augmented A-Box reasoner with a DL system is an open question we pose to the DL community.

References

[1] N. Abdallah, *The Logic of Partial Information*. New York: Springer, 1995

[2] J.F. Allen, *Maintaining Knowledge about Temporal Reference*. CACM Vol. 26 Nr. 11, 832–843, 1983

[3] A. Artale, E. Franconi, *Temporal Description Logics*, 1998. To appear in: Handbook of Time and Temporal Reasoning in Artificial Intelligence.

[4] F. Baader, P. Hanschke, *A Scheme for Integrating Concrete Domains into Concept Languages*. Proceedings of the 12th International Joint Conference on Artificial Intelligence, IJCAI-91, 452–457, Sydney, 1991.

[5] A. Borgida, C.L. Isbell, D.L. McGuinness, *Reasoning with Black Boxes: Handling Test Concepts in CLASSIC*. Proceedings of the Description Logics Workshop, 1996

[6] D.M. Deang, *Geometrical and Logical Modelling of Cartographic Objects*. Master Thesis in Computational Engineering, Computer Science Institute, University of Erlangen-Nuremberg, 2000

[7] N. Dershowitz, E.M. Reingold, *Calendrical Calculations*. Cambridge University Press, 1997

[8] F. Gallwitz et al., *The Erlangen Spoken Dialogue System EVAR: A State-of-the-Art Information Retrieval System*. In: *Proceedings of 1998 International Symposium on Spoken Dialogue - ISSD'98*. Sydney, 1998, 19–26

[9] V. Haarslev, R. Möller, *Spatioterminological Resoning: Subsumption Based on Geometrical Inferences*. Proceedings of the Description Logics Workshop, 1997, 74–78

[10] V. Haarslev, R. Möller, *RACE User's Guide and Reference Manual Version 1.1*. Memo-HH-M-289/99, Computer Science Dept., University of Hamburg, 1999

[11] V. Haarslev, R. Möller, *Description of the RACER System and its Applications*. To appear in: Proceedings International Workshop on Description Logics (DL-2001), Stanford, 2001.

[12] V. Haarslev, R. Möller, M. Wessel, *The Description Logic ALCNHR+ Extended with Concrete Domains: A Practically Motivated Approach*. International Joint Conference on Automated Reasoning, IJCAR'2001, Siena. Berlin: Springer, 2001

[13] I. Horrocks, *FaCT Reference Manual Version 1.6*. Computer Science Dept., University of Manchester, 1998

[14] M. Knorr, *Sprachliche und logische Zeitrepräsentation für Dialogsysteme*. Studienarbeit, Universität Erlangen-Nürnberg, 2000

[15] M. Kullmann, *Applying Description Logics to Decision Support*. Ph.D. thesis, LIIA-ENAIS, Strasbourg, 2001

[16] A.Y. Levy, M.-C. Rousset, *Combining Horn rules and description logics in CARIN*. Artificial Intelligence Journal, Vol. 104, 1998, 165–209

[17] LIIA-ENSAIS, *Usermanual for CICLOP Version 1.3b*. Strasbourg, 1999

[18] B. Ludwig, G. Görz, H. Niemann, *An Inference-Based Approach to the Interpretation of Discourse*. First Workshop on Inference in Computational Semantics (ICoS-1), Amsterdam. Jl. of Language and Computation, Vol. 1, No. 2 (2000), 241–258

[19] L.A. Resnick et al., *CLASSIC Description and Reference Manual for the COMMON LISP Implementation: Version 2.3*. AI Principles Research Department, AT&T Bell Laboratories, 1995