Combining Expression and Content in Domains for Dialog Managers

Bernd Ludwig and **Günther Görz** and **Heinrich Niemann**
bdludwig@forwiss.uni-erlangen.de
{goerz,niemann}@informatik.uni-erlangen.de
Bavarian Research Center for Knowledge Based Systems (FORWISS)
Am Weichselgarten 7, D-91058 Erlangen, Germany.

Abstract

We present work in progress on abstracting dialog managers from their domain in order to implement a dialog manager development tool which takes (among other data) a domain description as input and delivers a new dialog manager for the described domain as output. Thereby we will focus on two topics; firstly, the construction of domain descriptions with description logics and secondly, the interpretation of utterances in a given domain.

1 Introduction

Current research on dialog management is guided by two different ideas: firstly, to describe the discourse structure that the dialog manager is able to handle by a finite automaton using possible utterances as transitions (e.g. [10]), and secondly, to view the detection of discourse structure as a parameter estimation problem and to use statistical models for the description of discourse structure (e.g. [12], [8]).

Some serious problems remain with each of these approaches: first of all, they do not integrate a user model. The finite state method imposes hard restrictions on "free dialog", as far as vocabulary, syntax and order of utterances are concerned. The statistical approach, on the other hand, suffers from the sparse data problem and does not give theoretical insight into discourse theory.

But as researchers point out (see e.g. [13], [2]), it is important to explore the structure of discourse and the interactions of dialog and user model in order to obtain robust dialog systems. Studying the effect of natural language expressions on the state of discourse this paper tries to do a step in this direction by discussing the use of description logics for defining dialog domains formally and by describing how inference is performed on the basis of such a framework.

Following Hjelmslev [7], Eco [4] defines two essential ingredients for any semiotic system (and, in particular, any natural language):

- Expressiveness: what is the vocabulary, phonetics, and syntax of the language considered?
- Content: what knowledge can be expressed by a given system? How (i.e. by which expressions) is this knowledge organised in the semiotic system?

Obviously, the key problem is how to connect content and expressions in order to capture the meaning of expressions. Russell [15] proposes to divide the vocabulary into two classes:

- object words: define basic notions (in a train information scenario e.g. *train*, *time*, or *station*)
- dictionary words: can be defined using other words with already defined meaning (e.g. *departure time* or *arrival station*)

Trivially, there is an obvious analogy between object words and primitive concepts in description logics and dictionary words and derived concepts. We exploit this analogy to describe the application domain for a dialog manager by means of a terminology in DL. Doing this, we can define the meaning of basic and derived notions that are relevant in the domain under consideration.

2 λ-DRT and DL

A basic concept of any dialog system is a theory of discourse. We use Discourse Representation Theory (DRT) as introduced by [9] to describe the discourse generated during a dialog. DRT has simple semantics: A discourse representation structure (DRS) K with

$$K = \left[\begin{array}{l} d_1\, d_2 \,... \\ \hline \mathrm{cond}_1(d_1, d_2, ...) \\ \mathrm{cond}_2(d_1, d_2, ...) \\ ... \end{array}\right]$$

can be mapped into the logic language capturing the syntax of $\mathrm{cond}_i(d_1, d_2, ...)$ by:

$$\exists d_1, d_2, ... : \mathrm{cond}_1(d_1, d_2, ...) \wedge \mathrm{cond}_2(d_1, d_2, ...)$$

This mapping shows that a DRS essentially defines a set of assertions and thereby expresses extensional knowledge declared throughout the discourse.

On the other hand, a terminology of a certain domain can describe intensional knowledge used while constructing DRS out of utterances: the linguistic parser has to verify constraints imposed by subcategorization. E.g. a train usually departs from a station, i.e., a city name is a valid object of DEPART if there exists a railway station. To evaluate this constraint the parser makes use of DL reasoning services while constructing the semantics of the current utterance.

In this sense the incorporation of DL into our approach to dialog theory is crucial to overcome some of the limitations[1] of first order logic for describing natural language semantics.

3 Partial Information in Dialogs

A key issue for dialog management is the handling of partial information provoked by "incomplete" utterances. By this notion we mean the fact that in the *shared knowledge* of the participants there is not enough information available to, e.g., answer a question posed by the speaker. For the sake of illustration, let us consider the query

When does a train leave to Rome?

The information given in this query is (only) partial as it is incomplete or insufficient for the hearer to answer it. At least the station of departure is unknown and has to be asked for by the hearer if he wants to give some reasonable answer. Notably, the open world assumption of DL does not suffice to handle, because it does not allow to make an assumption about a formula (representing a query as above), not even by default.

We use First Order Partial Information Ionic Logic (FIL), as introduced by Abdallah [1], to handle situations of partial knowledge. FIL is a first order language whose interpretation function is partial, i.e. a formula can have a undefined truth value. Of course, interpretation functions can be ordered partially by

$$\begin{gathered}\mathcal{I} \leq \mathcal{J} :\Longleftrightarrow \mathrm{dom}(\mathcal{I}) \subseteq \mathrm{dom}(\mathcal{J}) \\ \forall x \in \mathrm{dom}(\mathcal{I}) : \mathcal{J}(x) = \mathcal{I}(x)\end{gathered}$$

In addition to first order formulae, FIL provides formulae (so called *ionic formulae*) of the form

$$*(\{\phi_1, ..., \phi_k\}, \xi).$$

Given a partial interpretation $\mathcal{I}$ that satisfies a set of standard first order formulae Σ, the ionic formula above says that ξ is true in an extension of $\mathcal{I}$ as long as there is no extension of $\mathcal{I}$ that assigns false to at least one ϕ_i.

$\Phi = \{\phi_1, ..., \phi_k\}$ is called *justification set* or *justification context*. Stating the semantics of a *ionic formula* more informally, we have that ξ is true (by default) as long as there is no evidence to the contrary from *justification context* Φ[2].

[1]First order logic lacks a mechanism for constructing well-defined terms from utterances. Its model theory is intensionally and ontologically inadequate for natural language expressions.

[2]In fact, [1] explains that an interpretation function for Φ can assign 1 to Φ (Φ is acceptable, written: $+*\Phi$), or 0 (Φ is inacceptable: $-*\Phi$), or not assign 1 (Φ is not acceptable: $\overline{+*}\Phi$), or not assign 0 (Φ is not inacceptable: $\overline{-*}\Phi$). This is due to the fact that in a partial logic for a relation symbol $R(x_1, ..., x_n)$ one has two sets R^+ and R^- to declare the semantics of $R(x_1, ..., x_n)$: $R(x_1, ..., x_n)$ is true if and only if $(x_1, x..., x_n) \in R^+$ and false if and only if $(x_1, x..., x_n) \in R^-$. $R^+ \cup R^-$ do not exploit the universe $\mathcal{U}$. So one can describe the four "positions" of $(x_1, ..., x_n)$ relative to R^+ or R^-, respectively.

4 Partial Information and Dialog Plans

As outlined above, the main characteristic of dialogs is the fact that information is given stepwise in the course of several utterances. For the design of a domain-independent dialog manager it is therefore important to develop an interpretation algorithm for utterances that is able to interact with the user in order to collect the neccessary information in any order.

Ionic formulae provide such a mechanism. Their *justification contexts* allow for infering what information is still missing while interpreting the current utterance. On the other hand, a *justification context* can be seen as a set of default assumptions to be accepted or rejected by the user later on.

From this point of view a *justification context* is the set of dialog goals to be fulfilled by the dialog manager in order to compute an answer to the user's question.

5 Integrating FIL and DL

To combine the advantages of DL as a concept language for domain modelling and FIL to describe partiality we integrate domain models into the FIL based reasoning.

To do this, we have to characterize the part of a domain model that can cause situations of partial information during a dialog. We state that any role that has the concept USER as domain or range is a source of "missing information" as these roles describe the user's attitudes eventually to be clarified in several dialog steps. Therefore, we define a set USERREL as follows:

$$\textsc{UserRel} = \{R : \mathrm{dom}(R) \subseteq \textsc{User} \vee \mathrm{range}(R) \subseteq \textsc{User}\}$$

For a formal description of the connection of DL and FIL, we give a translation mapping τ from DL to FIL which is sort of sensitive to USERREL defined above. The mapping, of course, resembles Borgida's [3] and the one by [16] and is identical except of the names of roles:

$$\tau(R) = \begin{cases} \lambda\, x, y. *(\{R(x,y)\}, R(x,y)),\ R \in \textsc{UserRel} \\ \lambda\, x, y.R(x,y), \text{ otherwise} \end{cases}$$

This says that all role names of a given TBox marked as partial are translated as ionic formulae. All other symbols are translated to first order formulae as usual by structural induction on the syntax of the DL. For example, given roles R, S_1, and S_2 with $R = S_1 \sqcup S_2$, the definition of τ is:

$$\tau(R) = \lambda\, x, y.\tau(S_1)(x,y) \vee \tau(S_2)(x,y)$$

whereas for role R and concepts C_1 and C_2 with $C_1 = \forall R.C_2$ we have

$$\tau(C_1) = \lambda\, x.\forall y : \tau(R)(x,y) \longrightarrow \tau(C_2)(x)$$

As it is shown by e.g. Borgida in [3], the mapping preserves satisfiability of DL expressions that are translated into a FIL formula, as FIL is an extension of First Order Predicate Logic and the translation from FIL to FOPL only uses the FOPL "sublanguage" of FIL. I.e. if a formula has some model in DL, then it has one in FIL, too. On the other hand, if a FIL expression is satisfiable, then it is in DL, too, if no ionic formula is contained (see [16]).

A ionic formula $*(R(\alpha,\beta),R(\alpha,\beta))$ is the translation of some role $R \in$ UserRel whose *justification context* $R(\alpha,\beta)$ can have one of the following states of acceptance

- $+ * R(\alpha,\beta)$ or $\overline{-*}R(\alpha,\beta)$: I.e. for any model $\mathcal{M}$ it is impossible that $\mathcal{M}|\not\models R(\alpha,\beta) \leftrightarrow \mathcal{M} \models \neg R(\alpha,\beta)$. This observation implies that either $\mathcal{M} \models R(\alpha,\beta)$ or $R(\alpha,\beta)$ is undefined. So $\models \tau(\alpha :: R : \beta) = *(R(\alpha,\beta),R(\alpha,\beta))$ does not contradict $\models \alpha :: R : \beta$.
- $- * R(\alpha,\beta)$ or $\overline{+*}R(\alpha,\beta)$: in this case, analogously, either $\mathcal{M}|\not\models R(\alpha,\beta)$ or $R(\alpha,\beta)$ is undefined implying that $\alpha :: \neg R : \beta$ is satisfiable.

In any case, τ preserves satisfiability depending on the state of acceptance of *justification contexts*.

6 Discussion of an Example

In the train information application that is considered throughout this paper, an appropriate terminology could include the following concepts and roles:

- Train, Depart, Time, Station
- At : Train $\times$ Time
  To : Train $\times$ ArrStation
  From : Train $\times$ DepStation
  DepartFrom : User $\times$ Station

DepartFrom is in UserRel and therefore mapped as:

$$\lambda\, u, s. *(\{\text{DepartFrom}(u,s)\}, \text{DepartFrom}(u,s))$$

Among many others we have the concepts

$$\begin{aligned}
\textsc{DepStation} &= \exists\textsc{DepartFrom}^{-1}.\textsc{User} \\
&\quad \sqcap \textsc{Station} \\
\textsc{TrainFrom} &= \exists\textsc{From}.\textsc{DepStation} \\
&\quad \sqcap \textsc{Train} \sqcap \textsc{Depart} \\
\textsc{TrainAtFrom} &= \exists\textsc{At}.\textsc{Time} \sqcap \textsc{TrainFrom} \\
\textsc{TrainAtFromTo} &= \exists\textsc{To}.\textsc{Station} \\
&\quad \sqcap \textsc{TrainAtFrom}
\end{aligned}$$

This terminology is translated to FIL follows (variables all-quantified):

$$\begin{aligned}
\text{DepStation}(s) &\iff \text{DepartFrom}^{-1}(s,u) \\
&\quad \wedge \text{User}(u) \wedge \text{Station}(s) \quad (1) \\
\text{TrainFrom}(t) &\iff \text{From}(t,s) \wedge \text{DepStation}(s) \\
&\quad \wedge \text{Train}(t) \wedge \text{Depart}(t) \quad (2) \\
\text{TrainAtFrom}(t) &\iff \text{At}(t,d) \wedge \text{Time}(d) \\
&\quad \wedge \text{TrainFrom}(t) \quad (3) \\
\text{TrainAtFromTo}(t) &\iff \text{To}(t,s) \wedge \text{ArrStation}(s) \\
&\quad \wedge \text{TrainAtFrom}(t) \quad (4)
\end{aligned}$$

On the basis of the terminology above the utterance

When does a train depart to Rome?

has the discourse representation structure (DRS)

$$\lambda x. \left[\begin{array}{l}
\text{t Rome} \\ \hline
\text{Train(t)} \\
\text{Depart(t)} \\
\text{Time}(x) \\
\text{ArrStation(Rome)} \\
\text{At}(\text{t}, x) \\
\text{To(t, Rome)}
\end{array}\right]$$

This structure is built up relying on DL reasoning: Rome is contained in the (linguistic lexicon) as CityName(Rome). In this example, the preposition *to* is mapped onto To as defined above. In order to combine *train*, *to*, and *Rome*, one has to check whether Rome $\in \exists$HasArrStation.Station which is evaluated by the data base to be true for *Rome*. This results in ArrStation(Rome). Generally, DRS are constructed by means of instance checking that expands type unification.

To infer TrainFrom(t) on the basis of the information available from the utterance, it is necessary to determine the truth value of $\phi(s) = \lambda s.\text{DepartFrom}^{-1}(s, \text{u})$ (see eq. 1; variable u is bound to constant u $\in$ User). A first order approach would answer false, because its interpretation function is total and there is no information in the DRS above in order to substitute a station name for s (making ϕ true). But in FIL we have (see above):

$$\begin{aligned}
\text{DepartFrom}^{-1}(s,u) \quad \iff \quad & *(\{\text{DepartFrom}(u,s)\}, \\
& \text{DepartFrom}(u,s))
\end{aligned}$$

Based on this rule, we can conclude immediately that $\phi(s)$ be true iff $\xi(s) = \lambda s.\text{DepartFrom}(\text{u}, s)$ is true unless there is information to the contrary (see Sect. 3).

As ξ's *justification context* still contains an unbound variable, the dialog manager interprets it as a question to be posed to the user (no default assumptions can be made). Therefore, the discourse plan will be updated and the dialog manager will react with

Where do you depart to from?

because $s \in$ Station. The answer

From Milan.

adds DepartFrom(u, Milan) to the *shared knowledge* so that the dialog manager can bind s to Milan. After that, we can infer by eq. 3 and eq. 4 TrainAtFromTo(t) substituting x by d and d by all constants c for whom At(t, c) $\wedge$ Time(c) is true.

The dialog plan is based on a speech act model[3]. Speech acts are determined by reasoning on the user's attitudes, grammatical information from, and coherence of utterances.

[3]For information dialogs we assume as minimal the set {inform, query, suggest, accept, reject}. This set can be extended to fit the needs of a certain application. See [11] for details.

7 From Notions to Vocabulary

To make a dialog system understand the user it has to know how expressions in natural language are connected to the abstract notions of the domain modell.

Natural languages normally offer the possibility to express the same notion by different synonyms. For example, one can say: *When does the train leave to Rome?* or, alternatively *When does the train depart to Rome?* In both cases, the notion of *train departure* is expressed.

When we inspected the EVAR[4] corpus of train information dialogs, we could see that synonyms occur frequently, even in relatively simple domains as train information. Our hypothesis is that in much larger more complicated domains that allow for a greater variability of expressions and vocabulary there are even more synonyms for a certain abstract notion.

To handle this phenomenon, we have to extend our domain description by adding formulae like

$$\mathrm{Syn}_1^C(x) \vee \cdots \vee \mathrm{Syn}_n^C(x) \Longrightarrow C(x)$$

for any concept C and and its synonymic expressions $\mathrm{Syn}_i^C(x)$ and, equally,

$$\mathrm{Syn}_1^R(x,y) \vee \cdots \vee \mathrm{Syn}_n^R(x,y) \Longrightarrow R(x,y)$$

for any role R and its synonyms $\mathrm{Syn}_i^R(x,y)$.

For the two questions above we would introduce

$$\mathrm{leave}(x) \vee \mathrm{depart}(x) \Longrightarrow \textsc{Depart}(x)$$

"mapping" the verbs *leave* and *depart* to DEPART.

8 Pragmatics of Concepts and Roles

Except of constructing a linkage between a domain model in terms of DL and a language model for the given domain, a configurable dialog manager must define an interface between its logical domain model and an arbitrary (mostly non-logical) problem solving component for the domain.

The reasoning mechanisms of the dialog manager and the problem solver, respectively, can be linked as follows: The problem solver evaluates relations between (i.e. roles of) discourse referents that are instantiations of concepts according to the previous utterances.

In the example outlined above, t can be asserted TRAINATFROMTO only if the query

$$\mathrm{At}(\mathrm{t},x) \wedge \mathrm{From}(\mathrm{t},\mathrm{Milan}) \wedge \mathrm{To}(\mathrm{t},\mathrm{Rome})$$

can be evaluated successfully (e.g. as a query to a database) and returns a list of connections from Milan to Rome. They can serve as data for continuing the dialog.

In the way outlined we can define an interface between dialog management and the pragmatics of the application which is by nature independent of a specific domain and therefore allows for abstraction of the dialog manager from its underlying domain-dependent problem solving component, linguistics, and discourse structure.

[4]EVAR is a publicly accessible information system on German Railway InterCity connections ([5]). The existing corpus of more than 1100 annotated dialogs contains samples of "real world" data with "naive" users.

9 Configuring a Dialog Manager

For the practical purpose of adapting a dialog manager for a specific task it is of great importance to observe that dialogs consist of a domain-independent and domain-dependent utterances.

Domain independence is related to establishing mutual understanding, dialog segmentation, and reference resolution. It is very important to consider these phenomena as they can be expressed in natural language: e.g. "Could you repeat that?", "Pardon?", "Next I want to ask you ...". Not to take such expressions into account would result in poor understanding capabilities of the dialog manager. Intentions, speech acts, and obligations can be expressed explicitly as well[5].

A model of these domain independent notions forms the basic capabilities of the dialog manager to engage in natural language conversation. To configure it for a certain application, one has to expand the model describing application-specific (i.e. domain-dependent) notions by

1. Adapting, extending, and specializing the given DL model so that it defines all notions of importance for the application. DL systems support the phase of designing a domain model by means of testing the satisfiability of terminologies. This is a major practical advantage for ensuring the robustness of the dialog manager compared to other approaches to domain modelling.
2. Defining the interface between the problem solving component and the dialog manager. I.e. defining what concept and role symbols will be evaluated by what functions of the problem solver.

10 Conclusions

We have established a connection between DL and FIL in terms of satisfiability via a mapping between formulae of each language. The correctness of inferences is assured by FIL (see [1]). This allows to conclude the correctness of reasoning with DRS as they can be mapped onto FIL formulae (see [9] and 2). We consider the combination of DL's knowledge representation facilities and FIL's inference mechanism for partial knowledge a well-founded basis for utterance interpretation and the description of mixed initiative dialogs in order to implement the "core engine" of an adaptive, configurable, and domain independent dialog manager. In this way we can separate linguistics and discourse theory from the knowledge engineering task to describe the application domain. This task can be performed by an application expert even without deep knowledge of dialog managers.

11 Future Research

Evidence from a number of experiments shows that humans perform domain reasoning while incrementally matching the speaker's utterances with their own expectations (see e.g. [14, 18]). Following this line of research,

[5]By modifying this dialog model, one can influence the planning of the dialog manager (c.f. footnote in Sect. 6).

we are studying the use of domain descriptions and DL reasoning services to model how dialog participants establish mutual beliefs on the basis of utterances.

Secondly, cooperating industrial partners we want to generate domain models for different domains. For the acquisition of "synonym knowledge" we will apply learning techniques from corpora of example dialogs collected by our partners. General work on DL learning (see e.g. [6]) as well as work on the combination of DL and linguistic processing (as done in [17]) will have to be considered.

Finally, we are working on user-friendly tools to define interfaces between domain models and its related problem solving components.

12 Acknowledgements

We want to thank two anonymous reviewers for their comments that helped to improve this paper.

References

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

[2] **Agarwal, R.:** *Towards a PURE Spoken Dialogue System for Information Access*, in: *Proceedings of the ACL/EACL Workshop on Interactive Spoken Dialog Systems: Bringing Speech and NLP Together in Real Applications*, Madrid, Spain, pp. 90–97, 1997

[3] **Borgida, A.:** *On the Relative Expressiveness of Description Logics and Predicate Logics*, Artificial Intelligence, volume 82 (1996), pp. 353–367

[4] **Eco, U.:** *La ricerca della lingua perfetta nella cultura europea*, Laterza, Roma, Bari 1993

[5] **Eckert, W., Niemann, H.:** *Semantic Analysis in a Robust Spoken Dialog System*, in: *Proc. Int. Conf. on Spoken Language Processing*, Yokohama, 1994, pp. 107–110

[6] **Frazier, M. and Pitt, L.:** *CLASSIC learning*, Machine Learning, 25 (1996), pp. 151–194

[7] **Hjelmslev, L.:**, *Omkring sprogteoriens grundlaeggelse*, Munksgaard, Kopenhagen 1943; engl. *Prolegomena to a Theory of Language*, Univ. of Wisconsin Press, 1961

[8] **Jurafsky, D., et al.:** *Automatic Detection of Discourse Structure for Speech Recognition and Understanding.*, in: *Proc. 1997 IEEE Workshop on Speech Recognition and Understanding*, Santa Barbara, 1997

[9] **Kamp, H. and Reyle, U.:** *From Discourse to Logic*, Kluwer, Dordrecht 1993

[10] **Larsen, L. B.:** *Development and Evaluation of a Spoken Dialogue for a Telephone Based Transaction System*, in: *Proc. EUROSPEECH 95*, pp. 1973–1976

[11] **Ludwig, B., Görz, G., Niemann, H.:** *User Models, Dialog Structure, and Intentions in Spoken Dialog*, submitted to KONVENS 98

[12] **Moeller, J.-U.:** *DIA-MOLE: An Unsupervised Learning Approach to Adaptive Dialogue Models for Spoken Dialogue Systems*, in: *Proc. EUROSPEECH 97*, pp. 2271-2274

[13] **Ward, K. and Novick, D. G.:** *On the Need for a Theory of Integration of Knowledge Sources for Spoken Language Understanding*, in: *Proceedings of the AAAI-94 Workshop on the Integration of Natural Language and Speech Processing*, Seattle, 1994, pp. 23–30

[14] **Poesio, M.:** *Discourse Interpretation and the Scope of Operators*, Ph.D. Thesis, Computer Science Dept., University of Rochester 1994

[15] **Russell, B.:** *The Object Language*, in: Russell, B.: *An Enquiry into Meaning and Truth*, Allen and Unwin, London 1950

[16] **Schmolze, J.G. and Israel, D.:** *KL-ONE: Semantics and Classification*, in *Research in Knowledge Representation for NL Understanding*, Tech Report 5421, BBN Laboratories 1983

[17] **Schnaittinger, K. and Hahn, U.:** *Constraining the Acquisition of Concepts by the Quality of Heterogeneous Evidence*, in: *Proceedings of the 21st Annual German Conference on Artificial Intelligence* (LNAI 1303), Springer, Berlin 1997, pp. 255–266

[18] **Traum, D.:** *A Computational Theory of Grounding in Natural Language Conversation*, Ph.D. Thesis, Computer Science Dept., University of Rochester 1994