Less is More:
Using a Single Knowledge Representation in Dialogue Systems

Iryna Gurevych†, Robert Porzel†, Elena Slinko†, Norbert Pfleger‡, Jan Alexandersson‡, Stefan Merten*

†European Media Laboratory
Schloss-Wolfsbrunnenweg 31c
69118 Heidelberg, Germany
{gurevych,porzel,slinko@eml.org}

‡DFKI GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany
{pfleger,janal}@dfki.de

***DFKI GmbH**
Erwin-Schrödinger-Str.57
67608 Kaiserslautern, Germany
merten@dfki.de

Abstract

The approach to knowledge representation taken in a multi-modal multi-domain dialogue system - SMARTKOM - is presented. We focus on the ontological and representational issues and choices helping to construct an ontology, which is shared by multiple components of the system, can be re-used in different projects and applied to various tasks. Finally, examples highlighting the usefulness of our approach are given.

1 Introduction

The ways in which knowledge has been represented in multi-modal dialogue systems (MMDS) show that individual representations with different semantics and heterogeneously structured content can be found in various formats within single natural language processing (NLP) systems and applications. For example, a typical NLP system, such as TRAINS (Allen et al., 1996), employs different knowledge representations for parsing, action planning and generation, despite the fact that what is being represented is common to all those representations, e. g., the parser representation for *going from A to B* has no similarity to the action planner's representation thereof (Ferguson et al., 1996). Also central concepts, for example *city*, are represented in multiple ways throughout the system.

The origin for this state of affairs is that the respective knowledge stores are hand-crafted individually for each task. Sometimes they are compiled into code and cease to be externally available. Where an explicit knowledge representation is used, we find a multitude of formats and inference engines, which often cause both performance and tractability problems. In this paper we introduce the results of an effort to employ a single knowledge representation, i. e., an *ontology*, throughout a complete multimodal dialogue system. Therefore, we will describe the underlying modeling principles and the benefits of such a rigorously crafted knowledge store for the actual and future MMDS.

In Section 2 we will introduce the representational formats pertinent to our ontology, followed by a description of our dialogue system in Section 3. In Section 4 we discuss the modeling principles underlying the ontology. Section 5 presents some examples of the various ways in which the common ontology is employed throughout the system. Concluding remarks are given in Section 6.

2 The Representational Formalism Used

Here we give a brief outline of the formalism pertinent to the following description of the ontology. Efforts originating in various W3C and Semantic Web projects brought about several knowledge modeling standards: Resource Description Framework (RDF), DARPA Agent Mark-up Language (DAML), Ontology Interchange Language (OIL), Web Ontology Language (OWL).[1] Domain and discourse knowledge represented in the ontology may be encoded using XML-based semantic mark-up languages, such as OIL, or DAML+OIL. In the work reported here, we used an ontology defined in the OIL-RDFS syntax. A detailed characterization of the formal properties of the OIL language can be found in Fensel et al. (2001). The FACT[2] system can be used as a reasoning engine for OIL ontologies, providing some automated reasoning capabilities, such as class consistency or subsumption checking. Graphical ontology engineering front-ends and visualization tools are available for editing, maintaining, and visualizing the ontology.[3]

The semantics of OIL is based on description logic extended with concrete datatypes. The language employs a combination of frame- and description logic. It provides most of the modeling primitives commonly used in the frame-based knowledge representation systems. Frames are used to represent concepts. These frames consist of a collection of classes along with a list of slots and at-

[1] See www.w3c.org/RDF, www.ontoknowledge.org/oil, www.daml.org and http://www.w3.org/2001/sw/WebOnt/ for the individual specifications.

[2] See also www.cs.man.ac.uk/∼horroks/FaCT/.

[3] See OilEd (oiled.man.ac.uk) for editing and FrodoRDFSViz (www.dfki.uni-kl.de/frodo/RDFSViz) for visualization.

tributes. Under the term *class* or *class expression* a class name, or an enumeration, or a property-restriction, or a boolean combination of class expressions is to be understood. Slots are interpreted as a collection of properties. They are divided into those that relate classes to other classes (so called *object properties*) and those that relate classes to datatype values (so called *datatype properties*). Slots can be filled by: class names, names of the atomic elements, collection of the above (conjunctive sets - *and*, disjunctive sets - *or*, or negation - *not*), concrete datatypes (*integers* and *strings*).

Then, domain and range restrictions of the slots can be defined. Domain restriction asserts that the property only applies to the instances of particular class expressions. Range restriction specifies that a property only assumes values that are instances of the respective class expressions. Slot fillers can have several types of further constraints, also called *facets*. These include *value-type* restrictions (all fillers must be of a particular class), *has-value* restrictions (there must be at least one filler of a particular class). The *value-type* restriction corresponds to the universal quantifier of the predicate logic. The *has-value* restriction is analogous to the existential quantifier. Another constraint on the slot fillers is *cardinality*, which limits the number of possible fillers of the given class. Atomic elements or individuals can also be associated with a class definition via slot constraints.

The decision to restrict ourselves to schemes based on the description logic is due to the fact that it allows to represent enough knowledge for the effective operation of envisaged NLP applications, e. g., those described in Section 5. We used the OIL language in particular as a whole range of software is freely available to support ontology construction as mentioned above. Additionaly, the usage of the ontology in Semantic Web applications would be simplified.

3 The SMARTKOM Ontology

The SMARTKOM system (Wahlster et al., 2001) comprises a large set of input and output modalities which the most advanced current systems feature, together with an efficient fusion and fission pipeline. SMARTKOM features speech input with prosodic analysis, gesture input via infrared camera, recognition of facial expressions and their emotional states. On the output side, the system features a gesturing and speaking life-like character together with displayed generated text and multimedia graphical output. It currently comprises nearly 50 modules running on a parallel virtual machine-based integration software called *Multiplatform*.[4]

As mentioned in the introduction, complex MMDS such as SMARTKOM require a homogeneous world model. This model serves as a common knowledge representation for various modules throughout the system. It represents a general conceptualization of the world (top-level or generic ontology) as well as of particular domains (domain-specific ontologies). This way, the ontology represents language-independent knowledge. The language-specific knowledge is stored elsewhere, e.g. in the lexicon containing lexical items together with their meanings defined in terms of ontology concepts.

The ontology described herein was initially designed as a general purpose component for knowledge-based NLP. It includes a top-level developed following the procedure outlined by Russell and Norvig (1995) and originally covered the tourism domain encoding knowledge about sights, historical persons and buildings. Then, the existing ontology was adopted in the SMARTKOM project and modified to cover a number of new domains, e. g., new media and program guides. The top-level ontology was re-used with some slight extensions. Further developments were motivated by the need of a *process hierarchy*. This hierarchy models processes which are domain-independent in the sense that they can be relevant for many domains, e. g., *InformationSearchProcess* (see Section 4.3 for more details).

Currently, the ontology employed by the system has about 730 concepts and 200 relations. The acquisition of the ontology went in two directions: top-down to create a top level of the ontology and bottom-up to satisfy the need of mapping lexical items to concepts. The purpose of the top-level ontology is to provide a basic structure of the world, i. e., abstract classes to divide the universe in distinct parts as resulting from the ontological analysis (Guarino and Poli, 1995). The domain concepts emerged through a comprehensive corpus analysis. The most important modeling decisions will be discussed in Section 4. Once available, the ontology was augmented with comments containing definitions, assumptions and examples that facilitate its appropriate use in a multi-component system such as SMARTKOM and its possible re-use in other systems. Such descriptions of ontology classes are particularly important as the meanings associated with them may vary considerably from one ontology to another.

4 Our Approach to Knowledge Representation

4.1 Type versus Role

Following the distinctions made by Guarino and Welty (2000), we first defined a collection of concepts that have *primary* ontological status. The guiding principle was to differentiate between the basic ontological entities and the roles taken by them in

[4]The abbreviation stands for "MUltiple Language / Target Integration PLATform FOR Modules".

particular situations, events, or processes. For example, a building can be a hospital, a railway station, a school, etc. But while taking all these roles, it doesn't cease to be a building. Another example is a person who can take the role of a school teacher, a mother, etc., but it still remains a person for its entire life.

Here the question arises, how deep the differentiation should go. Consider the example of a person: we give a concept *Person* a primary ontological status, but what about the concepts *Man* and *Woman*? Should they be given the same status? Our answer is positive and is based, on one hand, on the assumption that sex is the primary property that defines a person as a man or a woman, on the other hand, a functional approach shows that relations of these two classes to other classes and their other attributes can be determined by this property. In this way, the basic top-level ontological categorization in our system divides all concepts into two classes *Type* and *Role* (see Figure 1). As the class *Type* includes concepts with primary ontological status independent of the particular application, every system using the ontology for its specific purposes deals with the class *Role*.

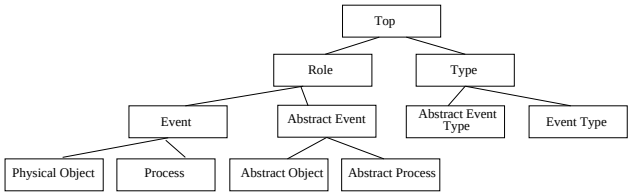

Figure 1: Top-level part of the ontology.

Role is the most general class in the ontology representing concrete roles that any entity or process can perform in a specific domain. It is divided into *Event* and *AbstractEvent*. Along with concrete events, i. e., free-standing entities existing essentially in space or in time, our model includes abstract objects, e. g., numbers or abstract properties, such as spatial relations, and abstract processes or rather abstracted states every real process can go through, such as *Start*, *Done*, *Interrupt*, etc. These are modeled separately thereby allowing a uniform description of the processes throughout the ontology.

4.2 Event versus Abstract Event

On the *Role* level we distinguish between *Event* and *AbstractEvent*. *Event* is used to describe a kind of role any entity or process may have in a real situation or process, e.g. a school or an information search. It is contrasted with *AbstractEvent*, which is abstracted from a set of situations and processes. It reflects no reality and is used for the general categorization and description, e.g., *Number*, *Set*, *SpatialRelation*. *AbstractEvent* has subclasses *AbstractObject* and *AbstractProcess*.

Event's are further classified in *PhysicalObject* and *Process*. In contrast to abstract objects, they have a location in space and time. The class *PhysicalObject* describes any kind of objects we come in contact with - living as well as non-living. These objects refer to different domains, such as *Sight* and *Route* in the tourism domain, *AvMedium* and *Actor* in the TV and cinema domain, etc., and can be associated with certain relations in the processes via slot constraint definitions.

4.3 Representing Processes

The modeling of *Process* as a kind of event that is continuous and homogeneous in nature, follows the frame semantic analysis used for generating the FRAMENET data (Baker et al., 1998). Based on the analysis of our dialogue data, we developed the following classification of processes (see Figure 2):

- *GeneralProcess*, a set of the most general processes such as duplication, imitation or repetition processes;
- *MentalProcess*, a set of processes such as cognitive, emotional or perceptual processes;
- *PhysicalProcess*, a set of processes such as motion, transaction or controlling processes;
- *SocialProcess*, a set of processes such as communication or instruction processes.

While the three last classes can be understood intuitively, the first one needs further explanation. It consists of several subclasses, such as *AbstractDuplicationProcess*, *AbstractRepetitionProcess*, *AbstractImitationProcess*, etc. These are abstract processes that are independent from the real processes and can take place at the same time with the main process.

The *MentalProcess* subtree includes *CognitiveProcess*, *EmotionProcess* and *PerceptualProcess*. Under *CognitiveProcess* we understand a group of processes that aim at acquiring information or making plans about the future. The further division of *EmotionProcess* into the following subclasses - *EmotionExperiencerObjectProcess* and *EmotionExperiencerSubjectProcess* - is due to the fact that an emotion can be either provoked by an object (e.g. The cry scared me) or can be experienced by an agent towards some object (e.g. I want to go home).

The *PhysicalProcess* has the following subclasses: the semantics of *ControllingProcess* presupposes the controlling of a number of artifacts, e. g., devices, *MotionProcess* models different types of agent's movement regarding some object or point in space, *PresentationProcess* describes a process of displaying some information by an agent, e. g., a TV program by Smartakus, an artificial character embedding the SMARTKOM system, *StaticSpatialProcess* consists in the agent's dwelling in some point

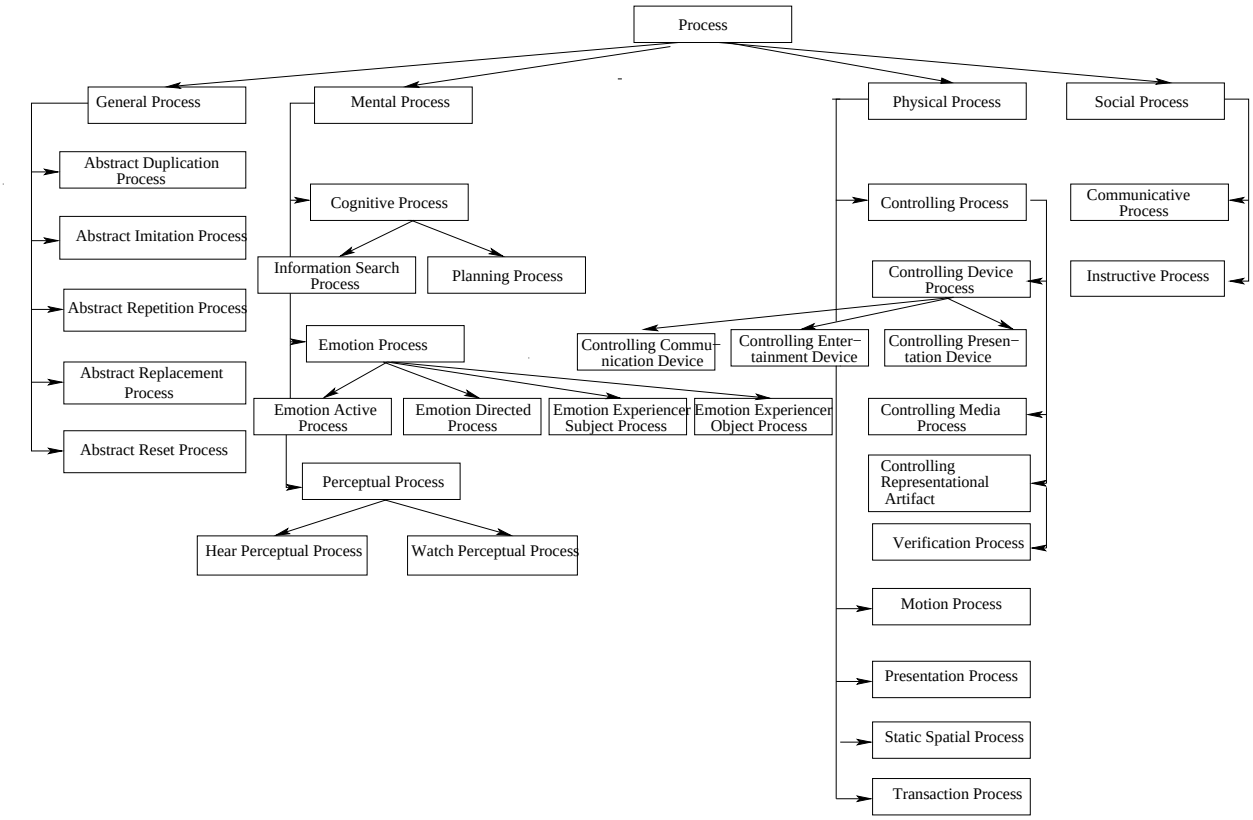

Figure 2: Process Hierarchy.

in space, *TransactionProcess* presupposes an exchange of entities or services among different participants of the process.

Another subclass of the *Process* - *SocialProcess* includes *CommunicativeProcess*, which consists in communicating by the agent a message to the addressee by different means, and *InstructiveProcess* which describes an interaction between an agent and a trainee.

4.4 Slot Hierarchy

The slot structure also reflects the general intention to keep abstract and concrete elements apart. A set of most general properties has been defined with regard to the role an object can play in a process: *agent*, *theme*, *experiencer*, *instrument* (or *means*), *location*, *source*, *target*, *path*. These general roles applied to concrete processes may also have subslots: thus an agent in a process of buying (*TransactionProcess*) is a *buyer*, the one in the process of cognition is a *cognizer*. This way, slots can also build hierarchical trees. The property *theme* in the process of information search is a required *piece-of-information*, in presentation process it is a *presentable-object*, i. e., the item that is to be presented, etc.

Consider the class *Process*. It has the following slots: *begin-time*, a time expression indicating the starting point, *end-time*, a time expression indicating the time point when the process is complete, *state*, one of the abstract process states. These slots describe properties that are common to all processes, and as such they are inherited by all subclasses of the *Process* class. An *EmotionExperiencerSubjectProcess* inherits the slots of the *Process* class, among them the slot *theme* that can be filled with any process or object (the basic idea is that any physical entity or the performance of any process can become an object of someone's emotion). It also has several additional properties such as *experiencer* to denote the one who undergoes the process, and *preference* to define the attitude an experiencer has to the object of its emotion.

4.5 Ontology instances

Consider the definition of the *InformationSearchProcess* in the ontology. It is modeled as a subclass of the *CognitiveProcess*, which is a subclass of the *MentalProcess* and inherits the following slot constraints: *begin-time*, a time expression indicating the starting time point, *end-time*, a time expression indicating the time point when the process is complete, *state*, one of the abstract process states, e. g., start, continue, interrupt, etc., *cognizer*, filled with a class *Person* including its subclasses.

The *InformationSearchProcess* features one additional

slot constraint, *piece-of-information*. The possible slot-fillers are a range of domain objects, e.g. *Sight*, *Performance*, or whole *set*s of those, e.g. *TvProgram*, but also processes, e.g. *ControllingTvDeviceProcess*. This way, an utterance such as:[5]

(1) *I hätte gerne Informationen zum Schloss*
I would like information about the castle

can be mapped onto the *InformationSearchProcess*, which has an agent of the type *User*, and a piece of information of the type *Sight*. *Sight* has a name of the type *Castle*. Analogously, the utterance:

(2) *Wie kann ich den Fernseher steuern*
How can I the TV control

can also be mapped onto the *InformationSearchProcess*, which has an agent of the type *User*, and has a piece of information of the type *ControllingTvDeviceProcess*.

Another example demonstrating how slot structures can be shared between some super- and subclasses: the subclass *AvEntertainment* inherits from its superclass *Entertainment* the following slots: *duration*, *end-time*, and *begin-time*, filled by the *TimeDuration* and *TimeExpression* respectively. The class *AvEntertainment* features two additional slots: *language*, its filler is an individual *Language* and *av-medium*, its filler is a class *AvMedium*. The class *AvEntertainment* has further subclasses - *Broadcast* representing an individual entry in a TV program, and *Performance* modeling an entry in a cinema program. Both of them inherit the slots of the superclasses *Entertainment* and *AvEntertainment*, while also featuring their own additional slots, e. g., *channel* and *showview* for the *Broadcast*, *cinema* and *seat* for the *Performance*. In Section 5.2, we will show how this feature can be effectively utilized by a specific dialogue interpretation algorithm called *overlay*.

5 Example Applications of Ontology

There is no agreed methodology for ontology evaluation. In our opinion, the usefulness of an ontology can be evaluated by examining the ways it is employed within the system, allowing to draw tentative conclusions as for the re-usability of the ontology and its portability with respect to new applications and NLP tasks. The ontology described here is used by the complete core of the system (Löckelt et al., 2002). In the next sections we give some examples of the usage within the project.

[5] All examples are displayed with the Germano riginal on top and a glossed translation below.

5.1 Semantic Coherence Scoring

We introduced the notion of *semantic coherence* as a special measurement which can be applied to estimate how well a given speech recognition hypothesis (SRH) fits with respect to the existing knowledge representation (Gurevych et al., 2003). This provides a mechanism increasing the robustness and reliability of multi-modal dialogue systems.

5.1.1 Challenge

One of the major challenges in making an MMDS reliable enough to be deployed in more complex real world applications is an accurate recognition of the users' input. In many cases both correct and incorrect representations of the users' utterances are contained in the automatic speech recognizer's n-best lists. Facing multiple representations of a single utterance poses the question, which of the different hypotheses corresponds most likely to the user's utterance. Different methods have been proposed to solve this problem. Frequently, the scores provided by the recognition system itself are used. More recently, also scores provided by the parsing system have been employed, e.g. Engel (2002). In this application, we propose a new ontology-based method and show that knowledge-based scores can be successfully employed to re-score the speech recognition output.

5.1.2 Solution

The software for scoring the SRHs and classifying them in terms of their semantic coherence employs the ontology described herein. This means, that the ontology crafted as a general knowledge representation for various processing modules of the system is additionally used as the basis for evaluating the semantic coherence of sets of concepts.

The scoring software performs a number of processing steps:

- converting each SRH into a concept representation. For this purpose, each entry of the system's lexicon was augmented with zero, one or multiple ontology concepts;
- converting the domain model, i.e. an ontology, into a directed graph with concepts as nodes and relations as edges;
- scoring concept representations using the shortest path between concepts based scoring metric.

For example, in our data (Gurevych et al., 2002) a user expressed the wish to get more information about a specific church, as:

(3) *Kann ich bitte Informationen zur*
May I please Information about the
Heiliggeistkirche bekommen
Church of Holy Spirit get

Looking at two SRHs from the ensuing n-best list we found that Example (5) constituted a suitable representation of the utterance, whereas Example (4) constituted a less adequate representation thereof, labeled accordingly by the human annotators:

(4) *Kann ich Information zur*
May I Information about the
Heiliggeistkirche kommen
Church of Holy Spirit come

(5) *Kann ich Information zur*
May I Information about the
Heiliggeistkirche bekommen
Church of Holy Spirit get

According to the lexicon entries, the SRHs are transformed into two alternative concept representations:

CR_1:{Person; Information Search Process; Church; Motion Directed Transliterated Process};
CR_2:{Person; Information Search Process; Church; Transaction Process}.

The scores are normalized as numbers on a scale from 0 to 1 with higher scores indicating better semantic coherence. Then, the resulting score assigned to Example 4 is 0.6, and the score of Example 5 is 0.75. The evaluation of the method against the hand-annotated corpus has shown that it successfully classifies 73.2% in a German corpus of 2.284 speech recognition hypotheses as either coherent or incoherent, given a baseline 54.55% derived from the annotation experiments (the majority class).

Additional application of the semantic coherence scoring method is the calculation of a semantic coherence score for SRHs taking into account their conceptual context (Porzel and Gurevych, 2003). Currently we are also experimenting with the ontology-based automatic domain recognition and domain change detection.

5.2 Computing Dialogue Coherence

The ontology provides a good basis for the enrichment and scoring of hypotheses - the two main tasks for the discourse module (Pfleger, 2002). What we call discourse processing is an essential processing step for any dialogue system since it provides an *interpretation* of the hypotheses based on the discourse history.

5.2.1 Challenge

As indicated in Section 5.1, a system processing spoken language and gestures is faced with analysis modules producing several hypotheses. A discourse module has not just the task of scoring these hypotheses in respect to the discourse state, but also interpretating and resolving ambiguities, e. g., (Löckelt et al., 2002).

5.2.2 Solution

There are several advantages for using an ontology. First, it enables a convenient way for interpreting common phenomena like partial utterances and ellipses. Second, and most notably, using overlay (Alexandersson and Becker, 2003) we can straightforwardly inherit information from one discourse state to another, even if the focussed instance of the ontology is from a different but related type than the one of the current hypothesis. The advantage of this technique becomes evident in the dialogue excerpt below.

The data structure of the discourse memory is based on the ideas presented in LuperFoy (1992), Salmon-Alt (2000). A three-tiered partition of a modality, discourse and domain layer is connected with a double threaded focus structure.

A non-monotonic unification-like operation called overlay serves as the main algorithm for manipulating instances of the ontology. It combines new information (cover) with old context information (background) by unifying where possible, and overwriting where unification would fail. Additionally, the operation does not fail if the types differ, but *assimilates* the background to the type of the cover - thereby possibly deleting information of the background - before the cover is layed over the background. During overlay we record a number of parameters, e. g., the number of type clashes (tc) , the amount of information stemming from background (bg) and cover (co) and the number of conflicting values (cv), which is combined using the formula below to form the score (see (Pfleger et al., 2002)).

$$score(co, bg, tc, cv) = \frac{co + bg - (tc + cv)}{co + bg + (tc + cv)}$$

The total score includes a fifth parameter *recency* that expresses how accessible the considered discourse state is.

To highlight the advantage of the ontology, consider the following dialogue excerpt between the user (U) and the system (S):

(6) U: *What's on TV tonight*

(7) S: [Displays a list of films] *Here you see a list of films.*

(8) U: *show me the cinema program.*

In our ontology, the structure for showing cinema program and the structure for showing tv program are related in that there exists a common superclass *AvEntertainment*

(see also Section 4.5) defining common slots, e. g., *begin-time*. Overlay makes it possible to inherit the time information deployed in (6) while enriching the hypotheses for (8) with contextual information.

5.3 Generating Interface Specifications

In this additional application, we proposed to use the knowledge modeled in the ontology as the basis for defining the semantics and the content of information exchanged between various modules of the system.

5.3.1 Challenge

In NLP systems, modules typically exchange messages, e.g., a parser might get word lattices as input and produce corresponding semantic representations for later processing modules, such as a discourse manager. The increasing employment of XML-based interfaces for agent-based or other multi-blackboard communication systems sets a *de facto* standard for syntax and expressive capabilities of the information that is exchanged amongst modules. The content and structure of the information to be represented are typically defined in corresponding XML schemata (XMLS) or Document Type Definitions (DTD).

As discussed above, ontologies are a suitable means for knowledge representation, e.g. for the definition of an explicit and detailed model of a system's domains. That way, they provide a shared domain theory, which can be used for communication. Additionally, they can be employed for deductive reasoning and manipulations of models. The meaning of ontology constructs relies on a translation to some logic. This way, the inference implications of statements, e.g. whether a class can be related to another class via a subclass or some other relation, can be determined from the formal specification of the semantics of the ontology language. However, this does not make any claims about the syntactic appearance of the representations exchanged, e.g. an ordering of the properties of a class.

An interface specification framework, such as XMLS or DTD, constitutes a suitable means for defining constraints on the syntax and structure of XML documents. Ideally, the definition of the content communicated between the components of a complex dialogue system should relate both the syntax and the semantics of the XML documents exchanged. Those can then be seen as instances of the ontology represented as XMLS-based XML documents. However, this requires that the knowledge, originally encoded in the ontology, is represented in the XMLS syntax.

5.3.2 Solution

The solution proposed states that the knowledge representations to be expressed in XMLS are first modeled in OIL-RDFS or DAML+OIL as *ontology proper*, using the advantages of ontology engineering systems available, and then transformed into a communication interface automatically with the help of the software developed for that purpose.[6]

Employing this approach, XMLS and DTDs are created such that they:

- stay logically consistent,
- are easy to manage,
- enable a straightforward mapping back to the respective knowledge representation for inference,
- allow the handling of a range of NLP tasks immediately on the basis of XMLS.[7]

The resulting schemata capture the hierarchical structure and a significant part of the semantics of the ontology. We, therefore, provide a standard mechanism for defining XMLS-based interface specifications, which are *knowledge rich*, and thus can be used as a suitable representation of domain and discourse knowledge by NLP components. Since the software that has been developed completely automates the transformation process, the resulting XMLS are congruent with the XML schema specifications. Furthermore, the ontology can be re-used in multiple systems as a single ontology can be used to generate application-specific communication interfaces.

However, the main advantage of our approach is that it combines the power of ontological knowledge representation with the strengths of XMLS as an interface specification framework in a single and consistent representation. Our experience shows, this would not have been possible for a complex dialogue system, if XML schemata were defined from scratch or hand-crafted, and constitutes a step towards building robust and reusable NLP components.

6 Concluding Remarks

In this paper, we presented an ontology which can be used as a single knowledge representation in a multi-modal and multi-domain dialogue system. We described the major modeling principles and the design choices made. Furthermore, we sketched some examples of the ontology application within the system.

Together these examples suffice to demonstrate the benefits of using a single knowledge representation throughout a dialogue system as opposed to using multiple knowledge representations and formats. An additional advantage of such a homogeneous world model

[6] This is a free software project. The package and respective documentation can be obtained from http://savannah.nongnu.org/projects/oil2xsd.

[7] E.g., the discourse module described in the previous subsection operates on the XML schema obtained via ontology transformation.

that defines the processing interfaces as well as the system's world knowledge is that no costly mappings between them are any more necessary. This means that modules receive only messages whose content is congruent to the terminological and structural distinctions defined in the ontology.

Our additional concern while designing the ontology was the re-usability of this component within our MMDS as well as other NLP systems. So far, the top-level ontology proved stable. We found the extensions on the lower levels of the ontology to be comparatively cheap. This single knowledge base was successfully tested and applied to multiple NLP problems, e.g., resolving bridging expressions in texts as well as for the resolution of metonymical and polysemous utterances next to defining communication interfaces for NLP components of the system, scoring of speech recognition hypotheses and overlay mechanism described above.

Acknowledgments

This work was partially funded by the German Federal Ministry of Education, Science, Research and Technology (BMBF) in the framework of the SmartKom project under Grant 01 IL 905 K7 and by the Klaus Tschira Foundation. The responsibility for the contents lies with the authors.

References

Jan Alexandersson and Tilman Becker. 2003. The Formal Foundations Underlying Overlay. In *Proceedings of the Fifth International Workshop on Computational Semantics (IWCS-5)*, Tilburg, The Netherlands, February.

James F. Allen, Bradford Miller, Eric Ringger, and Teresa Sikorski. 1996. A robust system for natural spoken dialogue. In *Proc. of ACL-96*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of COLING-ACL*, Montreal, Canada.

Ralf Engel. 2002. SPIN: Language understanding for spoken dialogue systems using a production system approach. In *Proceedings of ICSLP 2002*.

D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. Patel-Schneider. 2001. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2).

George Ferguson, James F. Allen, Brad Miller, and Eric Ringger. 1996. The desgin and implementation of the TRAINS-96 system. Technical Report 96-5, University of Rochester, New York.

Nicola Guarino and Roberto Poli. 1995. Formal ontology in conceptual analysis and knowledge representation. *Special issue of the International Journal of Human and Computer Studies*, 43.

Nicola Guarino and Chris Welty. 2000. A formal ontology of properties. In R. Dieng and O. Corby, editors, *Proceedings of EKAW-2000: The 12th International Conference on Knowledge Engineering and Knowledge Management*, volume 1937, pages 97–112. Springer Verlag.

Iryna Gurevych, Robert Porzel, and Michael Strube. 2002. Annotating the semantic consistency of speech recognition hypotheses. In *Proceedings of the Third SIGdial Workshop on Discourse and Dialogue*, pages 46–49, Philadelphia, USA, July.

Iryna Gurevych, Rainer Malaka, Robert Porzel, and Hans-Peter Zorn. 2003. Semantic coherence scoring using an ontology. In *Proceedings of the HLT-NAACL Conference*. to appear.

Markus Löckelt, Tilman Becker, Norbert Pfleger, and Jan Alexandersson. 2002. Making sense of partial. In *Proceedings of the sixth workshop on the semantics and pragmatics of dialogue (EDILOG 2002)*, pages 101–107, Edinburgh, UK, September.

Susann LuperFoy. 1992. The representation of multimodal user interface dialogues using discourse pegs. In *Proceedings of the ACL Conference*, pages 22–31.

Norbert Pfleger, Jan Alexandersson, and Tilman Becker. 2002. Scoring functions for overlay and their application in discourse processing. In *KONVENS-02*, Saarbrücken, September – October.

Norbert Pfleger. 2002. Discourse processing for multimodal dialogues and its application in SmartKom. Master's thesis, Universität des Saarlandes.

Robert Porzel and Iryna Gurevych. 2003. Contextual coherence in natural language processing. *Modeling and Using Context*, Springer, LNCS:to appear.

Stuart J. Russell and Peter Norvig. 1995. *Artificial Intelligence. A Modern Approach*. Prentice Hall, Englewood Cliffs, N.J.

Susanne Salmon-Alt. 2000. Interpreting referring expressions by restructuring context. In *Proceedings of ESSLLI 2000*, Birmingham, UK. Student Session.

Wolfgang Wahlster, Norbert Reithinger, and Anselm Blocher. 2001. SmartKom: Multimodal communication with a life-like character. In *Proceedings of the 7th European Conference on Speech Communication and Technology.*, pages 1547–1550.