Overlay as the Basic Operation for Discourse Processing in a Multimodal Dialogue System *

Jan Alexandersson and **Tilman Becker**
DFKI GmbH,
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
{becker|janal}@dfki.de

Abstract

We describe a nonmonotonic operation on typed feature structures – overlay – and show how it is applied to discourse processing in a multimodal dialogue system. A detailed example highlights the necessity and strength of such an operation.

1 Introduction

In this paper we make precise the common conception that discourse information is enriched by some kind of *unification* or *inheritance* from the discourse memory. We introduce *overlay* as this operation, give a precise definition and discuss its use as the default strategy in multimodal dialogue processing.

Similar problems in the area of dialogue processing has been investigated by [5]. In media fusion, as argued in [4; 3], unification of typed feature structures is an appropriate operation to combine information from different modalities. Here, the prerequisite for using unification generally holds: namely that the resulting information from various sources contains *no* conflicting parts. This is not the case in discourse processing, however, typed feature structures can and should be used in discourse processing too. We will show how overlay as a variant of unification is the appropriate operation in this case.

In short, overlay is a binary, non-commutative operation that uses unification where possible and else overlays, i.e., uses information from the first argument.

2 The SmartKom Project

SmartKom is a multimodal dialogue system currently being developed at several academic and industrial partners (see www.smartkom.org).

The key idea behind the system is to develop a kernel system which can be used within several application scenarios. Currently there exist three such scenarios – public, home, and mobile – which all are different in their external appearance, but share a lot of basic processing techniques. The system depicted in Figure 1 describes the "public scenario" system. Within this scenario, an intelligent telephone booth is developed with which one is able to book tickets, get information about different (local) activities, attractions etc.

2.1 Architecture

Technically, the system is composed by a number of components as depicted in Figure 1: First there are two recognizers (speech and gesture) which deliver input for the language and gesture analysis. Their hypotheses are brought together in media fusion. Important for this paper is that the output from media fusion is encoded in the domain modeling representation, which is also used by the discourse module and intention analysis module. The task of the intention analysis module is to rank and select which hypothesis is the most probable one. This process is done by, e.g., comparing scores from different modules involved in the analysis and context processing. During this process the hypotheses are sent to the domain knowledge modeling module and to the discourse module. The domain knowledge modeling module adds, e.g., default information to the hypotheses whereas the discourse module compares the hypotheses against the discourse history. The discourse module also computes a score which describes how good each hypothesis fits the discourse history. Based on the ranking performed by the intention analysis module, the action planner computes an appropriate action, which is then visualized and uttered. An action might involve communication with some external device, e.g., looking up a database, or switch on/off a video cassette recorder.

*The research within SmartKom presented here is funded by the German Ministry of Research and Technology under grant 01 IL 905. We'd like to thank Michael Kipp, Norbert Pfleger and Markus Löckelt for comments on earlier drafts.

SmartKom uses a multi-blackboard architecture (as used in [10]), where different components communicate by listening or writing to so-called data pools. Each component/module in the system can listen/write to any of these pools. All communication is coded in XML, and for each pool there exists an XML schema.

The current version is connected to a database containing information about the TV program and movie schedule in Heidelberg. Future versions of the system will include more external devices like phone, fax, home entertainment/TV, car navigation system, etc.

2.2 Discourse Processing

Even though the discourse module in the current version of SmartKom has several tasks, we will concentrate on one: the validation and enrichment of analysis (intention) hypotheses with information from the discourse memory.

The discourse module receives hypotheses directly from the intention analysis module. The hypotheses are validated and enriched with (consistent) information from the discourse history. During this process a score is computed which mirrors how well the hypothesis fits the history. Depending on the scores by the analysis modules and the score by the discourse modeler, the intention analysis module then picks the "best" hypothesis.

Thus there are two subtasks: (i) fill in *consistent* information from history and (ii) compute a score. Currently, we employ a rather simple algorithm for (i) and a heuristics for (ii):

(i) To add information from the dialogue history to the current hypothesis, the current hypothesis is iteratively compared to the previous discourse segments. We use a *combination* of unification and overlay: discourse markers (a parameterized "change"[1]) signal which parts must be overlaid, the remainder of the structures are unified. This combined operation might actually fail, in which case the algorithm steps back one segment in the discourse history.

E.g., in the case of a change in one parameter (*"Then another cinema–this one* [deictic gesture to another cinema on the map]*"*), the discourse markers as analyzed by speech-analysis provide the information that the `<movieTheater>` part of the current/new discourse structure must be overlaid over old segments from discourse memory. Other information like date and time of a show are unified, however. See Figure 6 for details.

(ii) Our simple heuristics for scoring the augmentation from discourse memory is a combination of the amount of information contained in the (combined) output and the (time) distance in discourse history.

[1]The user changes his mind. See section 4 for an example of this.

```
<complexType name="entertainmentEvent">
   <element name="time"
            type="sk:Time"/>
   <element name="location"
            type="sk:Location"/>
</complexType>
<complexType name="cinemaEntertainmentEvent"
             base="entertainmentEvent">
   <element name="film"
            type="sk:avMedium"
            minOccurs="1" maxOccurs="1"/>
</complexType>
<complexType name="Appointment">
   <element name="event"
            type="cinemaEntertainmentEvent">
</complexType>
```

Figure 2: XML type inheritance. `entertainmentEvent` is a general type, extended by `cinemaEntertainmentEvent`. However, an occurrence of an element `event` of type `cinemaEntertainmentEvent` *must* contain a sub-element of name `film` and type `sk:avMedium`, since `minOccurs="1"`.

3 Overlay

In this section, we present the idea that hypotheses are enriched with discourse information by a nonmonotonic operation from the discourse memory. We formally define *overlay* as this operation.

3.1 XML, Frames, and Feature Structures

All information in SmartKom is encoded in XML [8]. In particular, a MultiModal Markup Langage (M3L) is used which is defined in a set of XML schemas [9].

The formal definitions in this section are based on *typed feature structures* [2] since unification and other operations similar to overlay are usually presented this way. XML Schemas can be viewed as a kind of typed feature structures [2]: Types can be defined in XML schemas and prescribe the structure of XML elements. However, there are a few differences between the types of XML schemas and the types of typed feature structure: A type in an XML schema can be constructed by (unary only) inheritance (see Figure 2 for an example of a specialized type.). The content of an element must be filled with an element of *exactly* the defined type. Filling it with a super- or subtype is thus not allowed.

However, given a domain representation as (typed) feature structures, XML expressions together with appropriate algorithms can be *used* to implement these feature structures.

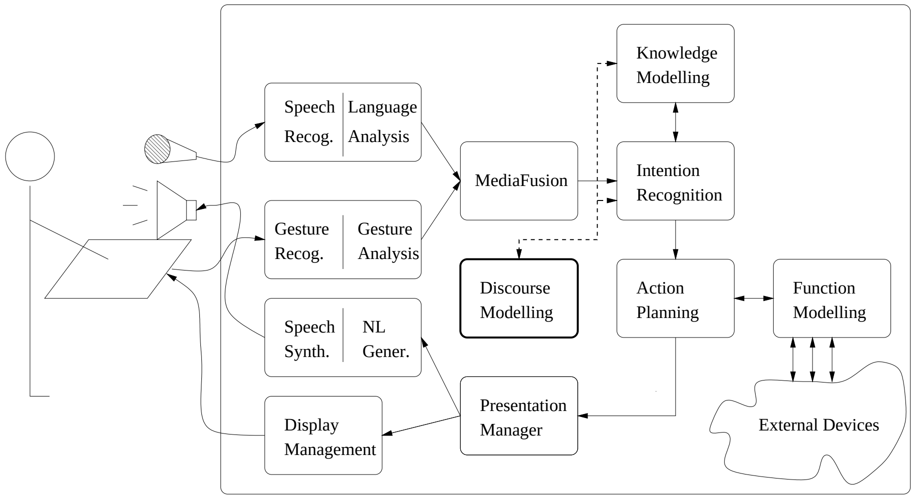

Figure 1: The architecture of the SmartKom demonstrator V1.

As an example consider the XML description of a 'cinemaEntertainmentEvent':

```
<performance>
  <time> ... </time>
  <location> ... </location>
  <film>
    <title> Schmalspurganoven
    </title>
  </film>
</event>
```

It can also be seen as a typed feature structure:

$$\left[\begin{array}{rl} \textit{type}: & \textit{cinemaEntertainmentEvent}' \\ \textit{time}: & ... \\ \textit{location}: & ... \\ \textit{film}: & [\ \textit{title}: \ \ \textit{Schmalspurganoven}\ ] \end{array}\right]$$

In section 4, we will switch back to XML based representations as used in the SmartKom system.

3.2 Typed Feature Structures and Unification

We first recall the definition of unification in untyped feature structures for an easier exposition of overlay and then proceed to the more complex case of overlay in typed feature structures.

For a detailed discussion of a related operation, *default unification*, see [6]. Default unification deals with similar problems of covering default information when defining specialized types in a type hierarchy. However, default unification is different in many respects, e.g., in that it calls for a commutative operation.

3.3 Unification in Untyped Feature Structures

Formally, the set FS of feature structures is recursively defined as follows:

Given a finite set A of attributes and a (possibly infinite) set V of values,

$a_1, \ldots, a_n \in A, n \geq 0$ and $f_1, \ldots, f_n \in FS \cup V$, then

$[a_1 : f_1, \ldots, a_n : f_n] \in FS$

The unification $a \sqcup b$ of two feature structures a and b is defined as:

$a = [a_1 : f_1, \ldots, a_n : f_n]$, $b = [b_1 : g_1, \ldots, b_m : g_m]$, then

$a \sqcup b = \{c_i : h_i \mid$
$c_i = a_j = b_k, h_i = f_j \sqcup g_k,$ where $f_j, g_k \in FS$ or
$c_i = a_j = b_k, h_i = f_j = g_k,$ where $f_j, g_k \in V$ or
$c_i = a_j, h_i = f_j, c_i \neq b_k, 1 \leq k \leq m$ or
$c_i = b_k, h_i = g_k, c_i \neq a_j, 1 \leq j \leq n\}$ or
FAIL if anyone recursive unification fails or values $f_j, g_k \in V$ differ.

3.4 Overlay

Informally the intention behind overlay is best explained by an analogy to putting shapes on top of each other as depicted in Figure 3.

The overlay $a \stackrel{+}{\Rightarrow} b$ of two feature structures a and b is defined as:

$a = [a_1 : f_1, \ldots, a_n : f_n]$, $b = [b_1 : g_1, \ldots, b_m : g_m]$, then

$a \stackrel{+}{\Rightarrow} b = \{c_i : h_i \mid$

$c_i = a_j = b_k, h_i = f_j \stackrel{+}{\Rightarrow} g_k,$ where $f_j, g_k \in FS$ or
$c_i = a_j = b_k, h_i = f_j,$ where $f_j, g_k \in V$ or
$c_i = a_j, h_i = f_j, c_i \neq b_k, 1 \leq k \leq m$ or
$c_i = b_k, h_i = g_k, c_i \neq a_j, 1 \leq j \leq n\}$

Note that overlay never fails, but it might not add any new information, i.e., $a \stackrel{+}{\Rightarrow} b = a$. Also, overlay is not a commutative operation (see below).

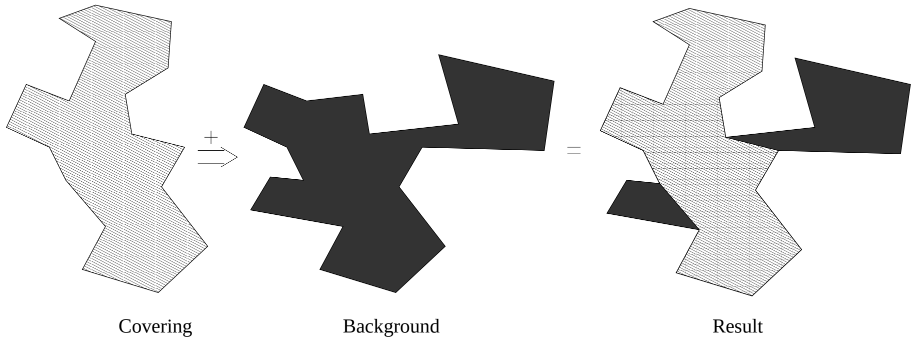

Figure 3: Overlay is "Putting shapes on each other"

3.5 Overlay in typed feature structures

In the case of a type clash between two feature structures a and b with types t_a and t_b, one could simply define the result of overlay to be a, completely covering b. However, there might be a non-trivial common supertype of a and b, say t_s with attributes that are filled in the background b, but not in the covering a. Such information should be added to the result $a \stackrel{+}{\Rightarrow} b$ of covering the background b with a.

$$t_s := [s_1 : t_1] \quad t_a := t_s \& [a_1 : t_{a_1}] \quad t_b := t_s \& [b_1 : t_{b_1}]$$

$$a = \begin{bmatrix} \mathit{type}: & t_a \\ s_1: & \\ a_1: & 17 \end{bmatrix} \qquad b = \begin{bmatrix} \mathit{type}: & t_b \\ s_1: & 42 \\ b_1: & 3.141 \end{bmatrix}$$

The desired result of $c = a \stackrel{+}{\Rightarrow} b$ is:

$$c = \begin{bmatrix} \mathit{type}: & t_a \\ & \\ s_1: & 42 \\ a_1: & 17 \end{bmatrix}$$

Figure 4: Saving background information in a common supertype.

First, we have to introduce the *restriction* of a feature structure to a supertype. In the example of Figure 4, the restriction of b to the supertype t_s would be $b|_{t_s} = [s_1 : 42]$. The restriction of a typed feature structure $a = [a_1 : f_1, \ldots, a_n : f_n]$ of type t_a to a supertype t_s is defined as $a|_{t_s} := \{a_i : f_i \mid a_i \text{ is defined in } t_s\}$. Now, we can proceed to define *overlay of typed feature structures*:

The overlay $a \stackrel{+}{\Rightarrow} b$ of two typed feature structures a and b of types t_a and t_b respectively, is defined as:

if $t_{b'}$ is the least upper bound (LUB) of t_a and t_b and $a = [a_1 : f_1, \ldots, a_n : f_n]$, $b|_{t_{b'}} = [b_1 : g_1, \ldots, b_m : g_m]$, then $a \stackrel{+}{\Rightarrow} b := a \stackrel{+}{\Rightarrow} b|_{t_{b'}}$ and

$a \stackrel{+}{\Rightarrow} b|_{t_{b'}} = \{c_i : h_i \mid$
$c_i = a_j = b_k, h_i = f_j \stackrel{+}{\Rightarrow} g_k$, where $f_j, g_k \in FS$ or
$c_i = a_j = b_k, h_i = f_j$, where $f_j, g_k \in V$ or
$c_i = a_j, h_i = f_j, c_i \neq b_k, 1 \leq k \leq m$ or
$c_i = b_k, h_i = g_k, c_i \neq a_j, 1 \leq j \leq n\}$

Note that unlike unification, overlay is not a commutative operation. That is, for two structures a, b, $a \neq b$, we have $a \stackrel{+}{\Rightarrow} b \neq b \stackrel{+}{\Rightarrow} a$ in the general case. Only in the case when a and b are unifiable, we have $a \stackrel{+}{\Rightarrow} b = b \stackrel{+}{\Rightarrow} a = a \sqcup b$.

Finally, we believe that overlay can easily be extended to handle co-indexed structures, however, we defer this discussion to a forthcoming paper.

4 Two Examples

We present two examples from our current system. The first (see Figure 5) shows the case where unification would suffice, and the second (see Figure 6) where it is necessary to use the overlay operation.

We show here only the XML input, however our internal representation is isomorphic to it. It is based directly on the XML schema definitions given for M3L. We compile the XML schemas automatically into a hierarchical set of Java classes, in a way similar to that of the Castor project [7]. Additionally to their read, write (marshal/unmarshal) and access operations our implementation provides unification and overlay.

The first example is this:

U2: *I'd like to see a movie tonight*
S3: [Displays a list of films] *Here you see a list of the movies running in Heidelberg.*
U4: *I'd like to see this* [pointing gesture] *one. Where does it run?*

Figure 5: First dialogue excerpt

After the analyzing the user request (U2), the place is added to the analyzed structure (by the knowledge modeler module) and the time expression *"tonight"* is mapped to an interval (by the discourse module) yielding the following structure[2], which is put into focus:

```
<domainObject>
  <entertainment>
    <performance>
      <beginTime>
        <function>
          <between>
            <from>
              2000-12-13T12:34:56
            </from>
            <to>
              2000-12-13T23:59:59
            </to>
          </between>
        </function>
      </beginTime>
      <cinema>
        <movieTheater>
          <contact>
            <address>
              <town>
                Heidelberg
              </town>
            </address>
          </contact>
        </movieTheater>
      </cinema>
    </performance>
  </entertainment>
</domainObject>
```

The system responds by showing a list of movies for Heidelberg (S3). The discourse memory now contains a user request (movies tonight) and a system inform (list of movies).

Then, the user selects one of the films by pointing at one particular film and uttering (U5) resulting in the following two analyses:

```
<domainObject>
  <entertainment>
    <performance>
      ...
    </performance>
  </entertainment>
</domainObject>
```

```
<domainObject>
  <entertainment>
    <broadcast>
      ...
    </broadcast>
  </entertainment>
</domainObject>
```

That is to say, the analysis component(s) could not determine if the user is speaking about film in a cinema or on TV. The discourse memory now compares the two structures against the dialogue memory. The first one even unifies with the focussed structure. Therefore the overlay operation produces the same resulting structure. For the second structure unification fails, but overlay succeeds. But, we assign a higher score to an overlay operation which detects that it would suffice with unification. The result of overlaying the first structure against the discourse memory is assigned a higher score, and hence will be preferred by the intention analysis:

```
<domainObject>
  <entertainment>
    <performance>
      <beginTime>
        ...
      </beginTime>
      <cinema>
        <movieTheater>
          <contact>
            <address>
              <town>
                Heidelberg
              </town>
            </address>
          </contact>
        </movieTheater>
      </cinema>
      <avMedium>
        ...
        <title>
          Schmalspurganoven
        </title>
      </avMedium>
    </performance>
  </entertainment>
</domainObject>
```

U7: *I'd like to make a reservation in this* [pointing gesture] *cinema.*
S8: *This cinema does not take reservations.*
U9: *Then another cinema. This one* [pointing gesture] *perhaps.*

Figure 6: Second dialogue excerpt

For the second example the dialogue has reached the point where the user has found a film and now selects a cinema to be able to reserve a ticket. (S)he performs the action U7 and after comparison with the discourse history and selection of the most probable hypothesis, the following structure is put as focus in the discourse memory:

```
<domainObject>
  <entertainment>
    <performance>
      <beginTime>
        <function>
          ...
        </function>
      </beginTime>
      <cinema>
```

[2]For the sake of clarity and space restrictions, we only show selected parts of the structures.

```
        <movieTheater>
          ...
          <name> Kamera </name>
          <contact>
           ...
          </contact>
        </movieTheater>
      </cinema>
      <avMedium>
        ...
        <title>
          Schmalspurganoven
        </title>
      </avMedium>
    </performance>
  </entertainment>
</domainObject>
```

Now we have a cinema for the film selected, but the system recognizes that it cannot reserve tickets in this movie, and informs the user (S8)[3] who selects another cinema (U9). The analysis components produce two hypotheses: first, the new cinema ("Studio Europa") wrapped in the structure representing "watch a movie" and second just the cinema, representing "information about this cinema."

```
<domainObject>
  <entertainment>
    <performance>
      <cinema>
        <movieTheater>
          ...
          <name> Studio Europa </name>
          <contact>
           ...
          </contact>
        </movieTheater>
      </cinema>
    </performance>
  </entertainment>
  <movieTheater/>
</domainObject>


<domainObject>
  <movieTheater>
    ...
    <name> Studio Europa </name>
    <contact>
     ...
    </contact>
  </movieTheater>
  <movieTheater/>
</domainObject>
```

We now have a conflict: The focus contains the selected cinema from U7 ("Kamera"), and the new structure contains another cinema ("Studio Europa"). The effect of overlaying the first hypothesis[4] with the focussed structure is shown below. The structure is exactly the focussed one except for the new name of the cinema (and some specific information not shown).

```
<domainObject>
  <entertainment>
    <performance>
      <beginTime>
       ...
      </beginTime>
      <cinema>
        <movieTheater>
          ...
          <name> Studio Europa </name>
          <contact>
            ...
          </contact>
        </movieTheater>
      </cinema>
      <avMedium>
        ...
        <title> Schmalspurganoven </title>
      </avMedium>
    </performance>
  </entertainment>
</domainObject>
```

5 Discussion and Future Work

We have presented a new operation on (typed) feature structures – overlay – and showed its necessity for discourse processing in the SmartKom system. Overlay is designed to use unification to retrieve information from the background for features where the overlay provides no information.

5.1 Scores

However, the background information might not always be applicable. Therefore, some kind of algorithm, using an evaluation or scoring of the quality of overlaying new input over different states from discourse memory as presented in sections 2.2 and 4 is needed.

At least two kinds of information can be used to determine such a score. On the one hand, there may be explicit (discourse) markers signaling overlay–see the discussion below. On the other hand, a heuristics can be developed based on (i) the amount of information that needed to be covered and (ii) on the amount of information gained by overlay through unification of nonconflicting features. Type (i) signals a deviation from the previous discourse situation and task. Type (ii) usually signals a refinement of the current situation where pure unification would often also succeed.

[3] We admit that this response is not the most cooperative one, but this example suits our needs.

[4] The second hypothesis will be ruled out because of its low score. Since comparing it against the focus results in a top-level unification failure (`<movieTheater>` vs. `<performance>`), overlay will not change it at all, and thus the score for the contribution of discourse memory is zero.

Some open questions remain that are not covered by overlay: For example, set valued objects such as the recipients of an e-mail message can either be accumulated by a union-like operation or replaced by a true overlay-like operation when the user gives new recipients and had previously mentioned others.

5.2 Discourse Markers

Overlay is typically used in situations where either some action or presupposition has failed (e.g., making a reservation is impossible) or the user utters corrections or repairs, indicating that the discourse status has changed (e.g., the user has changed their mind). Such situations are often marked with discourse cues like *but, though, then*. Often, the parts of the discourse history that need to be covered by overlay are explicitly given, e.g., in *"Then another cinema."* Given that SmartKom is a multimodal system, we have the opportunity to investigate such kinds of discourse markers in other modalities like gestures.

5.3 Overlay and Multimodality

For media fusion purposes the use of overlay would be necessary if we can point at (select) things and modify these with language on the fly: Suppose we select an object on the screen, and modify some of its facets (e.g., its colour) by uttering *"but red."* Then media fusion has to be able to overwrite that particular facet – a task that could be solved by overlay.

Finally, we will have to compare our definition of overlay with existing works. In, c.f., [1] it is argued that the effect of all nonmonotonic operations on feature structures can be captured by means of one (nonmonotonic) operation – default unification.

References

[1] Gosse Bouma. Feature structures and nonmonotonicity. *Computational Linguistics*, 18(2):183–203, 1992.

[2] R. Carpenter. *The logic of typed feature structures*. Cambridge University Press, Cambridge, England, 1992.

[3] Matthias Denecke. Integrating Knowledge Sources for the Specification of a Task-Oriented System. In *Workshop Proceedings 'Knowledge And Reasoning in Practical Dialogue Systems' of IJCAI '99*, 1999.

[4] Michael Johnston, Philip. R. Cohen, David McGee, Sharon L. Oviatt, James A. Pittman, and Ira Smith. Unification based multimodal integration. In *Proceedings of the 35th ACL*, pages 281–288, Madrid, Spain, 1997.

[5] Michael Kipp, Jan Alexandersson, Ralf Engel, and Norbert Reithinger. Dialog Processing. In Wolfgang Wahlster, editor, *VERBMOBIL: Foundations of Speech-to-Speech Translation*. Springer, 2000.

[6] Alex Lascarides, Ted Briscoe, Nicholas Asher, and Ann Copestake. Order independent and persistent typed default unification. *Linguistics and Philosophy*, 19(1):1–89, 1996. Revised version of ACQUILEX II WP NO. 40.

[7] The Castor project. `http://castor.exolab.org/`.

[8] W3C. XML. `http://www.w3.org/XML/`.

[9] W3C. XML schema. `http://www.w3.org/XML/Schema`.

[10] Wolfgang Wahlster, editor. *VERBMOBIL: Foundations of Speech-to-Speech Translation*. Springer, 2000.