Scoring Functions for Overlay and their Application in Discourse Processing*

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

Abstract

Overlay is a formal operation that is based on unification and allows the combination of new and possibly conflicting old information. Since overlay never fails, it is important to assign a score to the result of an overlay operation. We identify fundamental parameters for scoring functions and discuss a particular scoring function based on these parameters. We demonstrate two applications of overlay in discourse processing within the SmartKom project.

1 Introduction

Overlay is a formal operation that is based on unification of typed feature structures (TFS). It is a binary, non–commutative operation that allows the combination of conflicting structures, e.g., representing new and old information. Overlay uses unification where possible and otherwise overlays, i.e., uses information from the first argument over of the second.

It is particularly useful in discourse processing, see (Alexandersson and Becker, 2001), although it is defined there as a general operation. Overlay is a formalization of the procedures commonly used, e.g., when enriching an analysis hypothesis by discourse information through some kind of *unification* or *inheritance* from the discourse memory, see (Reithinger et al., 2000).

Similar problems arise in related fields, see (Kipp et al., 2000) for a discussion of dialogue processing and (Johnston et al., 1997; Denecke, 1999) for work in media fusion. They also use TFS for the representation of discourse objects.

Unlike unification, overlay never fails. E.g., when comparing a referring expression with multiple objects from discourse memory with overlay, every comparison will yield a result. Therefore it is necessary to assign a score to the result of overlay. In this paper, see section 4, we identify four fundamental parameters as the basis for a heuristic scoring function that is used in two applications of overlay in discourse processing. First, however, we set the context for our work by outlining the SmartKom project and the applications of overlay in discourse processing in section 2. Section 3 then recaptiulates the definition of overlay and section 4 describes our approach towards a heuristic scoring function. Finally, a conclusion and current work is outlined in section 5.

* The research within SmartKom presented here is funded by the German Ministry of Research and Technology under grant 01 IL 905. The responsibility for the content is with the authors. We would like to thank Norbert Reithinger and Bernd Kiefer for comments on earlier drafts.

2 The SmartKom Project

SmartKom is a multi–modal dialog system currently being developed by 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 as shown on the left 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 of a number of components as shown on the right in figure 1: First, there are recognizers for each modality (speech, gesture, and facial expression) which deliver input for the corresponding analysis modules. Their hypotheses are brought together in media fusion. The language analysis module resolves anaphora directly by querying the discourse modeling module. Important for this paper is that the output from media fusion (and the analysis modules) is encoded in the domain modeling representation, which is also used by the discourse module and intention analysis module.[1]

The output of media fusion is taken by intention analysis and then sent to three modules to enrich

[1] And also by all other relevant modules, e.g., by the action planner and the presentation modules.

it further: interaction modeling, domain knowledge, and discourse modeling. The task of the intention analysis module is then to rank and select which hypothesis is the most probable one. In this process, the discourse modeling module compares the hypotheses against the discourse history, which is stored locally in the discourse module. It also computes a score which describes how good each hypothesis fits the discourse history. Finally, based on the ranking performed by the intention analysis module, the action planner computes an appropriate system action, the results of which are then visualized and uttered. An action might involve communication with some external device, e.g., searching a database, or switching on/off a video cassette recorder.

SmartKom uses a multi-blackboard architecture (as used in (Wahlster, 2000)), 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 encoded in XML, and for each pool there exists an XML schema. Together these schemas form the *Multi-Media Markup Language* - M3L.

The current versions of SmartKom are connected to various external services and functionalities, e.g., a database containing information about the TV program and movie schedule in Heidelberg, a phone, a Lotus Notes database for address book and email functionalities, a document scanner[2] home entertainment like TV set and VCR, a car navigation system, etc.

2.2 Discourse Processing

A task one is often faced with in discourse processing can be described as *"Given incomplete information, enrich it from other sources in order to proceed."* Examples are media fusion (e.g., combining speech and gesture input), discourse plan recognition, and the retrieval of information from discourse memory. In this very general view, there are many diverse and often highly specialized strategies to obtain missing information. However, often one is tempted to generalize and say that the incomplete information is "unified" with information from other sources. In (Alexandersson and Becker, 2001) we argue that the operation that is applied is in fact not unification but a similarly universal operation which we have named *overlay*.

Even though the discourse module in SmartKom peforms several tasks, we will use two tasks which are based on overlay to discuss the scoring function introduced in section 4: the validation and enrichment of analysis (intention) hypotheses with information from discourse memory as well as the matching of referential expressions (currently only anaphora) with the objects in discourse memory.

2.2.1 Enriching from Discourse Memory

The discourse module receives hypotheses directly from the intention analysis module. Each hypothesis is compared and enriched with a selected number of discourse states from memory using overlay. The scoring function mirrors how well the hypothesis fits the history.

Thus there are two subtasks: (i) fill in *consistent* information from history and (ii) compute a score. The algorithm for (i) is described in more detail in (Alexandersson and Becker, 2001), whereas this paper emphasises the scoring function used in (ii).

(i) When comparing a hypothesis against a discourse memory object, some information may be present in both descriptions, signaling a potential match. Some information may be missing in the hypothesis and can then be added from that object. But some information may be contradicting – it must be taken from the hypothesis, masking the corresponding part of the discourse object, thus the name overlay. Details are given in the next section.

(ii) In scoring the results, there a number of parameters. The amount of information that was present in hypothesis and stored state, the amount of information that could be added from the stored state, the amount of information in the stored state that had to be overlaid by new information from the hypothesis, and finally in this case the prominence/accessibility of the stored state. The scoring function is discussed in section 4.

2.2.2 Anaphora Resolution

Another usage of a discourse memory is resolving referential expressions, especially in language analysis. In SmartKom, we currently handle only anaphoric expressions, but we expect to extend our approach to general referential expressions. The text generation module sends descriptions (in the domain modeling representation language) of all objects that are mentioned in the output, together with all relevant linguistic features[3] to the discourse module. When the language analysis module finds an anaphoric expression, it sends a typed (as far as type can be inferred from context) but otherwise unspecified object, together with the linguistic features to the discourse module that uses overlay to enrich the information from a suitable object in the object memory. The highest scoring match is then returned to the language analysis module. The details of the algorithm and the scoring function are laid out in section 4.2.

[2]The document scanner is a camera integrated into the public setup, it scans documents placed onto the display surface.

[3]In German, these are gender and number.

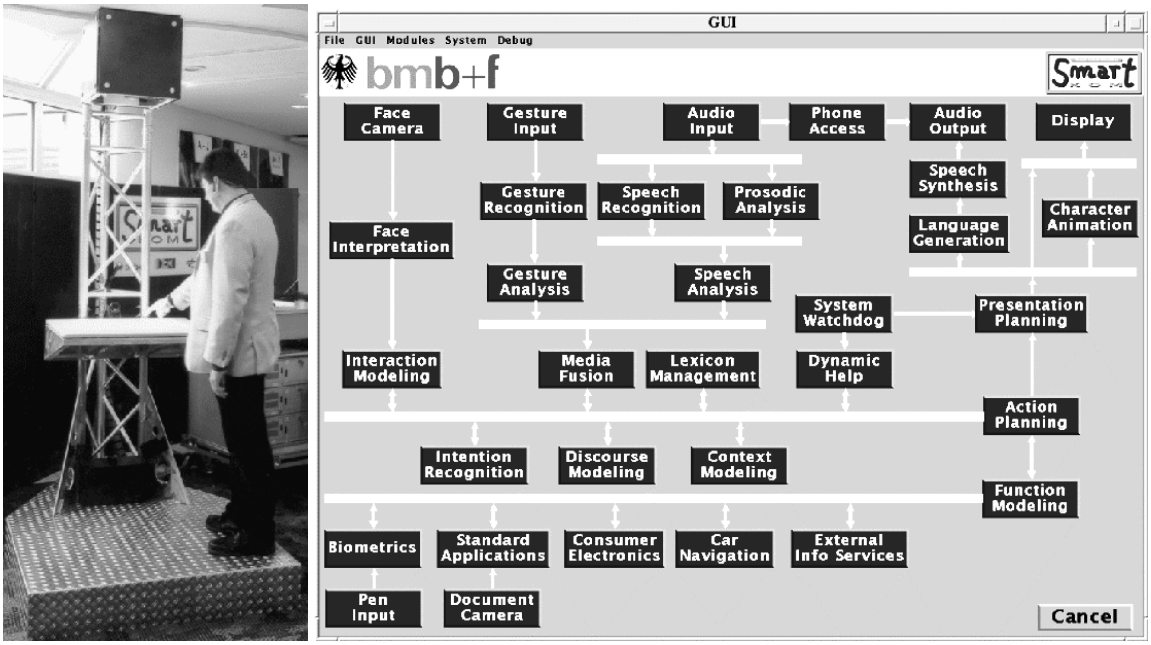

Figure 1: The public scenario and the architecture of the SmartKom demonstrator V2.1.

3 Overlay

In the SmartKom project we have choosen to encode all communication between different modules in XML (W3C,). For each interface there exists (at least) one XML Schema describing the syntax of a message. Each message is hence an XML document obeying the corresponding schemata. For the domain model we transform the ontology[4] into an XML-Schema, where each class in the ontology is mapped onto a `ComplexType`. This process is non-trivial because ontologies allow for, e.g., multiple inheritance whereas XML schemata does not (see below). Therefore some simplifications has to be made by hand.

In (Alexandersson and Becker, 2001) an operational semantics was given for UNIFY and OVERLAY based on TFS (Carpenter, 1992). The reason for using TFS instead of XML documents is that it is the way unification and similar operations are presented. It was indicated that there exist a transformation of XML Schema[5] to TFS, but there are some important differences:

- In XML Schemata a type can be defined by unary inheritance only
- In an XML document it is possible to give a feature (in XML terminology: an `element`) a more *specific* type than the one indicated by the corresponding Schema[6].
- A type definition inheriting elements must not specialize the type of these elements.

Instead of a formal procedure for the mapping of XML documents to typed feature structures we indicate that such transformation exists. Given a domain representation as (typed) feature structures, XML expressions together with appropriate algorithms can be *used* to implement feature structures. As an example, consider the XML schema definition of a "Entertainment", "AvEntertainment", and "Performance"

```
<xsd:complexType name="Entertainment">
 <xsd:complexContent>
  <xsd:extension base="event:CulturalArtifact">
   <xsd:sequence>
    <xsd:element name="beginTime"
     type="timeID:TimeExpression_id"/>
    <xsd:element name="endTime"
     type="timeID:TimeExpression_id"/>
    <xsd:element name="length"
     type="m3lID:TimeDuration_id"/>
   </xsd:sequence>
  </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>
```

[4] edited in OilEd – see `http://oiled.man.ac.uk/`

[5] or more precise: XML documents

[6] See `http://www.w3.org/TR/xmlschema-0/#UseDerivInInstDocs`

```
<xsd:complexType name="AvEntertainment">
  <xsd:complexContent>
    <xsd:extension base="event:Entertainment">
      <xsd:sequence>
        <xsd:element name="language"
          type="m3lID:Language_id"/>
        <xsd:element name="subtitles"
          type="m3lID:Language_id"/>
        <xsd:element name="avMedium"
          type="event:AvMedium"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>


<xsd:complexType name="Performance">
  <xsd:complexContent>
    <xsd:extension base="event:AvEntertainment">
      <xsd:sequence>
        <xsd:element name="seats"
          type="domainID:DomainSeats_id"/>
        <xsd:element name="fees"
          type="domainID:DomainFees_id"/>
        <xsd:element name="cinema"
          type="event:Cinema"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

and a relevant XML document together with its corresponding TFS:

```
...
<performance>
  <beginTime> ... </beginTime>
  <language> ... </language>
  <avMedium>
    <title> Schmalspurganoven
    </title>
  </avMedium>
  <cinema> ... </cinema>
</performance>
...
```

$$\Longleftrightarrow$$

$$\left[\begin{array}{ll} \textsc{AvEntertainment} & \\ \textit{beginTime}: & \ldots \\ \textit{language}: & \ldots \\ \textit{avMedium}: & \left[\begin{array}{ll} \textsc{AvMedium} & \\ \textit{title}: & \textit{Schmalspurganoven} \end{array}\right] \\ \textit{cinema}: & \ldots \end{array}\right]$$

3.1 Overlay for Typed Feature Structures

OVERLAY operates on two typed feature structures called *covering* and *background*. The names indicates that OVERLAY can be seen as putting structures on top of each other. Important for OVERLAY is a procedure that allow for the computation of the *assimilation* of a TFS, i.e. the features in the background defined in the least upper bound type (lub) of the types of the covering and background. In our case we are using a type hierarchy imposed by the limitations of XML schema. This hierarchy describes a tree of types and in what follows we assume a function $lub(a,b)$ that computes the most specific common supertype for two types of a and b. We denote a TFS $\langle t,[a_1:f_1,\ldots,a_n,b_n]\rangle \in TFS$ where t is the type, $a_1,\ldots,a_n \in F$ the features and $f_1,\ldots,b_n \in V = A \cup TFS$ the values. As indicated above and in what follows, we will also use the following graphical representation for denoting a TFS where, however, the type might be omitted.

$$\left[\begin{array}{ll} \textsc{T} & \\ a_1: & f_1 \\ \ldots & \\ a_n: & f_n \end{array}\right]$$

For formal definitions of TFS we refer to, c.f., (Carpenter, 1992; Krieger, 1995). Now we can define the *assimilation* of a TFS to another TFS.

Definition 1 *(assimilation)*
Let

- $a,b \in TFS$, *such that* $a = \langle t_a,[a_1:f_1,\ldots,a_n:f_n]\rangle$
- $t_s = lub(t_a,t_b)$

then, the assimilation of a *to* b $(a|_b)$ *is defined as* $a|_b := \langle t_b,[a_i:f_i,\ldots,a_j:f_j]\rangle$ *such that* a_i *is defined in* t_s. □

We can proceed to define *overlay for typed feature structures*:

Definition 2 *(overlay)*
Let

- $a = \langle t_a,[a_1:f_1,\ldots,a_n:f_n]\rangle$ *and* $b = \langle t_b,[b_1:g_1,\ldots,b_m:g_m]\rangle$
- $t_{b'} = lub(a,b)$

then $overlay(a,b)$ *is defined as:*
$overlay(a,b) := overlay'(a,b|_a)$
$overlay'(a,b) = \langle t_a,\{c_i:h_i \mid$
•(i) recursion:
$c_i = a_j = b_k, h_i = overlay(f_j,g_k)$, where $f_j,g_k \in FS$ or
•(ii) if covering and background have (atomic) values, use covering:
$c_i = a_j = b_k, h_i = f_j$, where $f_j,g_k \in A$ or
•(iii) if feature is absent in background, use covering:
$c_i = a_j, h_i = f_j, c_i \neq b_k, 1 \leq k \leq m$ or
•(iv) if feature is absent or has no value in covering, use background:
$c_i = b_k, h_i = g_k\}\rangle$

□

Overlay is unlike unification not a commutative operation. That is, for two structures a, b, $a \neq b$, we have $overlay(a,b) \neq overlay(b,a)$ in the general case. However, there is one exception: if a and b are unifiable, then $overlay(a,b) = overlay(b,a) = unify(a,b)$. Finally, note that the ability to define types via inheritance in XML schema in fact controls the behaviour of OVERLAY.

4 A Scoring Function

The overall score of an OVERLAY operation should reflect how well the covering fits the background in terms of non-conflicting features. Another important point which should be covered by the scoring mechanism is the occurrence of a type clash between two features. In this case a unification of two feature structures would fail and this has to beexpressed by a lower score.

These assumptions lead to a heuristic that is based on the contrast between the amount of non-conflicting features and the amount of conflicting features (including type-clashes). This heuristic uses four scoring parameters (initialized to zero). During OVERLAY these parameters are incremented as indicated below where the roman numbers refer to definition 2:

co a feature or a (atomic) value in the result stems from the covering. **co** is incremented for each feature in the covering, cases $(i) - (iii)$ and also for each value in the covering, cases (ii) and (iii).

bg a feature or a (atomic) value in the result occurs in the background. **bg** is incremented for features at (i), (ii), and (iv), and also for values in (ii) and (iv).

tc type clash, i.e., the type of the covering and background was not identical. This is identified during the computation of the assimilation.

cv conflicting values. This occurs when the value of a feature from the background is overwritten (ii).

The sum of **co** and **bg** minus the sum of **tc** and **cv** will be weighted by the sum of **co**, **bg**, **tc** and **cv**. This leads to a function (shown in formula 1) whose codomain is $[-1, 1]$.

Formula 1

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

□

The positive extremal ($score(co, gb, tc, cv) = 1$) indicates that the feature structures are unifiable. The negative extremal ($score(co, gb, tc, cv) = -1$) indicates that all information from the background has been overwritten by information from the cover. A score within this interval indicates that the cover more or less fits the background: the higher the score is, the better the cover fits the background. Negative values signals that conflicting and thus overlayed values outweigh unifiable values (positive values vice versa). In the following two subsections, we show the practicality of the scoring function applied to the tasks of the discourse module addressed here – enrichment, validation and anaphora resolution. Both applications use a fifth parameter *recency* that expresses how accessible the discourse state under consideration is. However, this parameter is particular to discourse processing and not general for OVERLAY.

4.1 Enrichment & Validation

During processing, the discourse module receives a set of intention hypothesis from the intention analysis module. These hypotheses are validated and enriched with consistent information from the discourse history. In (Alexandersson and Becker, 2001) we argued that the fundamental operation for this is the OVERLAY operation. Now we show the usefulness of the above mentioned scoring function by means of the processing of the dialogue excerpt depicted in figure 2.

U1: *What's on TV tonight*
S2: [Displays a list of films] *Here you see a list of films.*
U3: *That seems not very interesting, show me the cinema program.*

Figure 2: An excerpt of a sample dialogue

For U1 the analysis produces the TFS[7] as depicted in figure 3. This structure is stored into the discourse memory and put in focus.

The system responds by showing a list of films that are running on TV within the considered time interval (S2). The discourse memory now contains a user request (TV program tonight) and a system inform (list of films).

Then, the user changes her mind and asks for the cinema program (U3). The resulting analyses is depicted in figure 4.

$$\begin{bmatrix} \text{PERFORMANCE} \\ beginTime: & [] \\ avMedium: & [] \end{bmatrix}$$

Figure 4: Analysis for (U3).

The discourse module now compares the structure against the dialogue memory (in this case (U1)) using OVERLAY. Applying a unification operation on these two arguments would fail due to the type clash between `performance` and `broadcast` (as depicted in figure 5), but OVERLAY succeeds. However, this type clash causes a lower score for resulting structure (depicted in figure 6). So the discourse module will

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

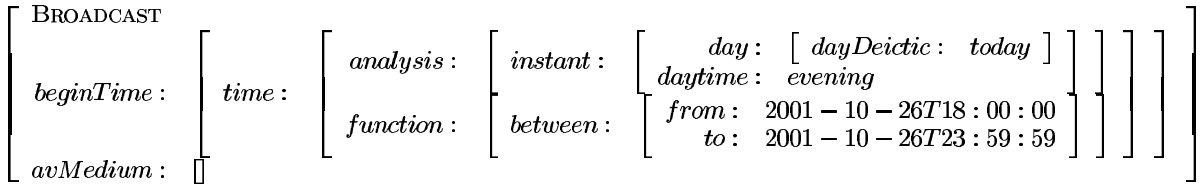

Figure 3: TFS for U1

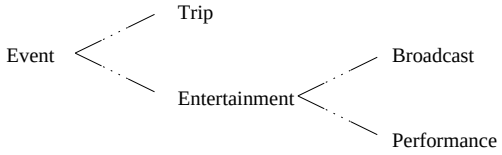

Figure 5: An excerpt of the type hierarchy for the two subtypes of `entertainment`

return the result of overlay enriched with temporal information from the context.

co two features from covering are included in the result; $\mathbf{co} = 2$

bg eight features and three values from background are included in the result; $\mathbf{bg} = 11$

tc one type clash has been detected (`performance` vs. `broadcast`); $\mathbf{tc} = 1$

cv no conflicting features; $\mathbf{cv} = 0$

Applying the above introduced scoring function 1 would result in a score of: $score(2, 11, 0, 1) = 0.857$. This result reflects a good matching between covering and background, although a type clash occured.

4.2 Anaphora Resolution

In this section we show how OVERLAY can be used for anaphora resolution focusing on anaphora–antecendent pairs which do not agree in some linguistic features.[8]

Typically, the identification of the correct antecedent for an anaphor is based on a search over a list consisting of the potential candidates for antecedent. This search is based on a number of anaphora resolution factors which are used to track down the correct antecedent. Factors employed frequently in the resolution process include number and gender agreement, semantic consistency, semantic and syntactic parallelism, proximity etc. These factors decompose into two classes: one concerns the properties of the candidates (like number and gender agreement and semantic consistency) and the other class concerns the structure of discourse (like syntactic and semantic parallelism, or proximity). Although both classes are mandatory, we will focus here on the first class. Since the resolution factors of the first class serve more as a filter to eliminate unsuitable candidates, our approach focuses on assessing the list of candidates. The resulting list has to be processed further, using the second class of resolution factors but this is not focus of this paper.

The initial antecedent candidates are either mentioned by the system or mentioned by the user. This list is narrowed by a focus structure and typically contains candidates stemming from a couple of previous turns. Since the focus of this paper is OVERLAY and an adequate scoring function we refer to (Pfleger, 2002) for details of the underlying focus structure or the representation of the stored discourse objects. For our purpose it is sufficient to assume a list of candidates where each candidate is represented within our domain modeling language, together with all relevant linguistic features.

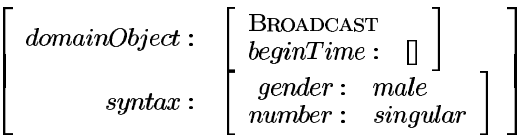

Figure 7: Sample Request from the language analysis module for the utterance: "Dann nimm den auf."(*"Then tape it!"*)

When the language analysis module finds an anaphoric expression, it sends a typed (as far as the type can be inferred from context) but otherwise unspecified object (depicted in figure 7), together with the linguistic features to the discourse module. This object is then compared to each element of the candidate list via OVERLAY.[9] Consider for example the follwing discourse excerpt:

[8]The following example will be German; in English there are, e.g., several collective nouns which disagree in number with their corresponding anaphor (e.g. government, team, etc.)

[9]So far, the actual implementation has only been tested in another project, named NaRATo, which is based on the same modules.

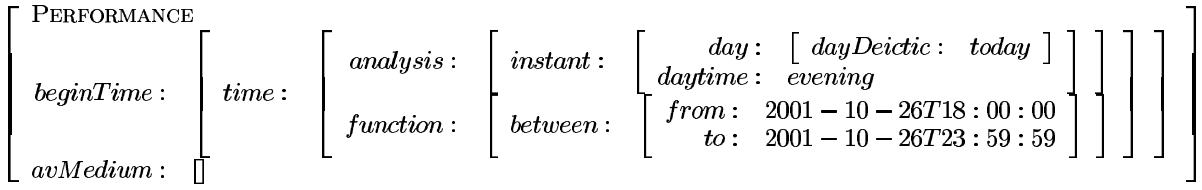

Figure 6: The result for applying OVERLAY on (U3) and (U1)

U1: *Ich würde gerne heute Abend einen lustigen Film sehen.*
I would like to see a comedy tonight
S2: [Displays a list of films] *Diese Sendung [↗] ist der Tagestip*
(This broadcast [↗] is the tip of the day.)
U3: *Dann nimm den auf!*
(Then tape it!)

The intended antecedent for the anaphor *den* (gender: male, number: singular) will be the focused object (depicted in figure 8) from the previous system utterance (S2). However, since the focused object was verbalised as *Sendung* (gender: female, number: singular) a unification–based approach for anaphora resolution would rule out this antecedent canidate because of a mismatch of the gender information. Using OVERLAY this candidate remains in the list, however the mismatch of the gender information is expressed by a lower score:

co three features and two values from covering are included in the result; $\mathbf{co} = 5$

bg eleven features and six values from background are included in the result; $\mathbf{bg} = 17$

tc no type clash; $\mathbf{tc} = 0$

cv one conflict between the values of the feature `gender` (female versus male); $\mathbf{cv} = 1$

Applying the above introduced scoring function 1 would result in a score of: $score(5, 17, 0, 1) = 0.913$. This result reflects a good matching between covering and background, although the values of one feature differed in the two structures.

5 Conclusion and Further Work

We have taken up the task of defining a scoring function to rate the results of the OVERLAY operation, which unlike unification, never fails. As the basis, we have identified four fundamental parameters which correspond to common, new, and old information. In section 4, we have given an general scoring function that is based on these parameters and used in two applications of overlay in discourse processing in the project SMARTKOM. As a special parameter, *recency* is used in discourse processing in addition to the scoring function for overlay itself.

5.1 Further Work

We are currently working on extending this work in a number of directions:

- The scoring function can be extended with weighting factors on the parameters, so we can employ learning techniques to calibrate the weights based on a corpus. This will also allow for an evaluation of the scoring function.
- The work on anaphora resolution is extended to handle general referring expressions. This depends mainly on the language analysis module while little change in the discourse processing module is needed.
- The formal properties of overlay call for further investigation, e.g., the relation between overlay and variants of default unification as defined in (Carpenter, 1993) appears to be quite close.
- Finally, we hope to see other applications of overlay outside of discourse processing.

References

Jan Alexandersson and Tilman Becker. 2001. Overlay as the Basic Operation for Discourse Processing in a Multimodal Dialogue System. In *Workshop Notes of the IJCAI-01 Workshop on "Knowledge and Reasoning in Practical Dialogue Systems"*, Seattle, Washington, August.

Bob Carpenter. 1992. *The logic of typed feature structures.* Cambridge University Press, Cambridge, England.

Bob Carpenter. 1993. Skeptical and credulous default unification with application to templates and inheritance. In A. Copestake E. J. Briscoe and V. de Paiva, editors, *Inheritance, Defaults and the Lexicon*, pages 13–37. Cambridge University Press, Cambridge, England.

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

Michael Johnston, Philip. R. Cohen, David McGee, Sharon L. Oviatt, James A. Pittman, and Ira Smith. 1997. Unification based multimodal inte-

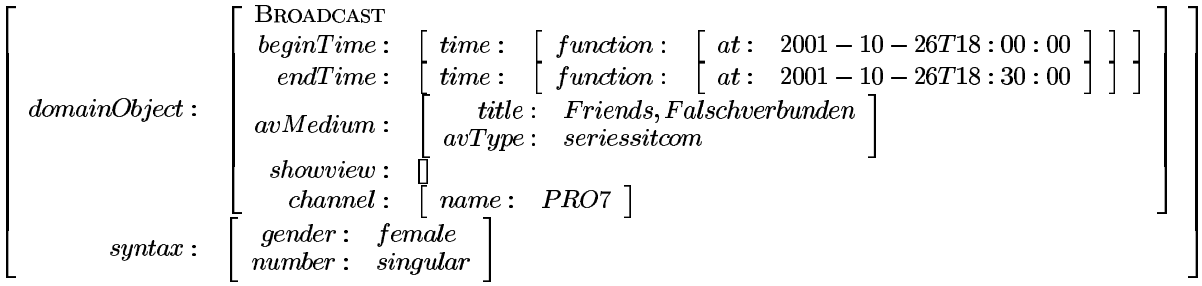

Figure 8: Output from the text generation module for the focused object in the system utterance "*Diese Sendung [↗] ist der Tagestip*" *(This broadcast [↗] is the tip of the day).*

gration. In *Proceedings of the 35th ACL*, pages 281–288, Madrid, Spain.

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

Hans-Ulrich Krieger. 1995. *TDL—A Type Description Language for Constraint-Based Grammars. Foundations, Implementation, and Applications.* Ph.D. thesis, Universität des Saarlandes, Department of Computer Science, september.

Norbert Pfleger. 2002. Discourse processing in multimodal dialogues. Master's thesis, Unversität des Saarlandes. Forthcoming.

Norbert Reithinger, Michael Kipp, Ralf Engel, and Jan Alexandersson. 2000. Summarizing multilingual spoken negotiation dialogues. In *Proceedings of the 38th Conference of the Association for Computational Linguistics (ACL'2000)*, pages 310–317, Hong Kong, China.

W3C. XML. `http://www.w3.org/XML/`.

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