The Formal Foundations Underlying Overlay*

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

December 5, 2002

Abstract

Combining information from the current utterance with its context in some discourse is a very general task for which many different approaches have been proposed. For dialog systems with frame-based representations, we present a single, well-formalized operation, called overlay that computes the maximally compatible combination of new information and old information. In this combination, preference is always given to new information. In the strict definition of overlay, all compatible information from the background is integrated into the result. However, with a generalized definition of overlay, partial information from the background can be left out. An implementation of overlay is used in the SmartKom system as an embedded part of a discourse modeler.

1 Introduction

In most types of discourse, a new communicative action introduces information that can be added to the context either without changes or with retracting old information. When adding information without conflicts, information is given on topics that had not been mentioned before, e.g., the time of some event, or information is made more precise, e.g., *tonight* is extended to *after 9pm.* Often, however, old information is retracted: be it that information on some topic is changed, e.g., *not tonight but tomorrow night*,

*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 Pfleger and Hans-Ulrich Krieger for discussions and the anonymous reviewers for comments on earlier drafts.

or that the topic itself is changed, making (parts of) the current context invalid, e.g., *and now I need to find a parking garage near the movie theater.*

In this work, we give an enhanced formalization for the process of combining the information from the current utterance with the previous context. We call this operation overlay, see also [AB01, PAB02, LBPA02]. Throughout this paper, we present overlay in terms of single utterances in a multimodal dialog system with a frame-based meaning representation. However, we believe that this work applies to other types of discourse, such as documents and even intra-sentential phenomena like gapping and one-anaphora. We have already applied overlayto anaphora resolution, see [PAB02].

The user intention in SmartKom is represented by instances of our domain model. However, we have driven this approach to the extreme since everything, including dialogue acts, processes, etc., is represented in the domain model (see [GMP03] for more information).

A crucial element of this work is the assumption of a typed frame-based formalism, where we use typed feature structures as a well-understood formalization. We would like to stress that although the aim of what we discuss in this paper should be modeling-neutral we have some basic assumptions on the modeling. Objects which share properties should be modeled together. As we will see below, an "entertainment" frame is more general than, for instance a "go-to-see-a-movie" frame. Moreover a "watch-something-on-TV" frame is related to "go-to-see-a-movie" in a sense that they both have, e. g., a "begin-time" property or feature. This property should be defined in the entertainment frame. One the other hand, a *location* feature in the "go-to-see-a-movie" frame should only be defined for precisely this frame, since it is obsolete for watching TV. In the same way, the *channel* only makes sense for the "watch-something-on-TV" frame. In our implementation, with which we verify our theory – the SmartKom system (see below) – an ontology based on the ideas of [BFL98, RN95] has been constructed (see also [PG02, GMP03] for another usage of the ontology). A simplified excerpt of the ontology describing just those frames discussed above is shown in figure 1.

Consider the following example of a dialog[1] between a user who is seeking information about the movie and TV program and the SmartKom system:

(1) **User**: I'd like to go to the movies tonight.

(2) **SmartKom**: Here (↗) are the films showing in Heidelberg.

[1] The only multi-modal aspect of this dialog is the presentation of the actual film titles on the display and a pointing gesture by the presentation agent, marked by ↗.

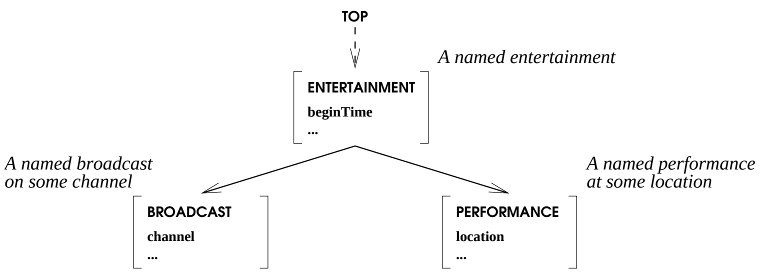

Figure 1: An excerpt from the SmartKom ontology showing the more general entertainment frame and the two specialized frames performance and broadcast.

(3) **User**: No, there is nothing interesting there,

(4) **User**: what is running on TV?

After the system's reply (2), the context contains information about the topic *going to the movies* and some details like *in Heidelberg* and *tonight*. Note, that in the SmartKom system, certain defaults, e. g., the location Heidelberg in this example, are incorporated into the representation of the user utterances. The next user utterance (3) could be seen as meta-talk, signaling a change of topic which is then made explicit in the final utterance (4). Since the two topics *movies* and *TV* are clearly related, some of the information from the old context should be retained, i. e., *in Heidelberg* and *tonight*.[2] The deep reason for this is that both topics serve the common goal of finding some entertainment which is still valid. Which parts of the old context should be retained in this case depends on the *type* of the topic. Time and place is relevant for both movie theaters and the TV program while any information (other than the city) about the movie theaters is irrelevant for the TV program. The overlay operation is based on typed feature structures as the representation formalism for the meaning representation of the context and the new utterance. Hence, our operation automatically takes these differences into account.

Although overlay always succeeds, we have defined a measure for the degree of compatibility between background and cover through a scoring function, see also section 2.3. This score expresses a degree of *topic cohesion*, i.e., whether the new utterance is on the same topic as the discourse context. However, it is not a measure for general discourse cohesion (see also [Pfl02]).

[2]We are not taking into account *inferred* information like, e.g., eliminating films shown in the movie theaters from presentations of the TV program, since the user has marked them as uninteresting.

Architecture of the SmartKom system

Figure 2 shows the relevant part of the system architecture. The input by the user is analyzed separately in the modalities speech, gesture and facial expression and a *modality fusion* module integrates them into a single representation. Since the analysis modules generate multiple hypotheses of the user's intention, all combinations are computed by the modality fusion module and are presented as a hypothesis lattice. In the following process of selecting the most likely new state of the dialog, defaults, situative knowledge, and finally context information is added to the hypotheses. There, the task of the *discourse modeling* module is to *enrich* the hypotheses with information from the current context and also to *validate* the resulting combination. Enrichment is basically computed by the overlay operation which is combined with a scoring function, see [PAB02], to validate the results of overlay. Since these operations are carried out after modality fusion, they are completely independent of the multi-modal setting.

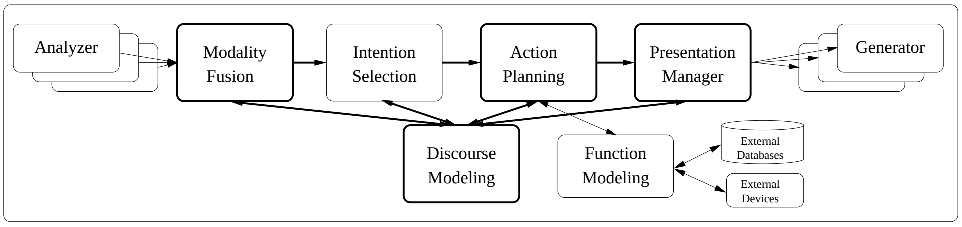

Figure 2: Architecture of SmartKom from the point of view of the discourse modeling module - DiM.

The representations of context and user intention are based on an ontology modeling the domain of the possible tasks that SmartKom can be used for and all objects and processes that can be parts of these tasks, see [GMP03]. The classes in the ontology are connected by unary inheritance only. However, in this paper we extend the definition of overlayto domain models with multiple inheritance structures, see section 2.2. In the implementation, these structures are encoded in M3L, an XML based language with a very strict XML schema description which directly reflects the ontology. For the purposes of this paper, we will use *typed feature structures* (henceforth TFS) [Car92, Kri95] which has a number of advantages. It is a well understood formalism which encompasses the same type of inheritance structures as the XML-encoded ontology[3] and the unification operation is closely related to the overlay operation, in fact it is incorporated in the definition of overlay. Finally, TFS allow for a very compact presentation. In figure 3 we show a possible corresponding TFS for the sample sentence *I'd like to go to the movies tonight.*

$$\left[\ldots \quad \mathit{entertainment}: \begin{bmatrix} \textsc{Performance} & \\ \mathit{beginTime}: & \mathit{tonight} \\ \mathit{cinema}: & \left[\ \textsc{Cinema}\ \right] \end{bmatrix}\right]$$

Figure 3: TFS for *I'd like to go to the movies tonight.*

For a more detailed description of the discourse modeling module and how TFS and overlay are embedded into a complex representation of the discourse history in SmartKom, see [Pfl02, PAB02].

2 Overlay

We first give a short informal recapitulation of the overlay operation. Overlay is a non-monotonic function based on unification and hence working on two TFSs, namely the *covering* and the *background*, where covering in our usage stands for new and background for old information. The core idea behind the function is to unify the arguments where possible and to overwrite conflicting information with the value of the covering. In case two arguments are incompatible, i.e., the types of the arguments are not related via a super-type relation, the information and the type of the background is "transformed" to be of the type of the covering thereby possibly removing information which is not defined in the covering. This operation is referred to as *assimilation*. In that way we can inherit information like, e.g., time and place in the example in section 1. In this section we explore *strict* overlay, i.e., all possible information from the background is used (see section 3 for a generalization).

This paper makes two contribution to prior work (see [AB01, PAB02]):

- we extend the original definition to give an account for one new case – the covering is more general than the background – not tackled before.

[3] We will not give a formal proof of this relation here, but rather indicate that the transformation is more or less direct. In general, an ontology also allows for restrictions on role-fillers, e.g., a table must have 3 or 4 legs. Such constraints have to be encoded elsewhere.

- our first definition was based on a type hierarchy which was constructed using unary inheritance only. Below we will extend the definition of overlay to account for type hierarchies with multiple inheritance, thus allowing overlay to work in more expressive domains.

2.1 The formal semantics

We now turn to the formal definition of the overlay operation for typed feature structures, e. g., [Car92, Kri95]. Next, we define a machinery for a type system where types are inherited by unary inheritance. Later, in section 2.2, we will extend the definition to work for type systems where two types have no unique least upper bound. We start by defining types and typed feature structures.

For the following definitions we assume a finite set of types $\mathcal{T} = \{t, t_1, \ldots\}$. The root of the hierarchy is denoted t_{root}. The types are related within a hierarchy in which types can be defined by means of unary inheritance only. That t_i is a super-type of t_j is denoted $t_i \preceq t_j$. Important for the work here is that a feature is introduced once in the hierarchy. This is referred to as *Feature Introduction* [Car92, p. 86]. Clearly, (i) the type hierarchy forms a tree and hence (ii) every pair of types has a unique least upper bound (henceforth LUB) which might be t_{root}. We also assume a finite set of typed features $\mathcal{F} = \{f, f_1, \ldots\}$. f_{t_i} denotes a feature whose value is restricted to the type t_i. Furthermore, we have a (possibly infinite) set of atomic values $\mathcal{A} = \{a, a_1, \ldots\}$ and a set of values $\{v, v_1, \ldots\} \in \mathcal{V} = \mathcal{A} \cup \mathcal{S}$ where $\mathcal{S}$ is the set of typed feature structures. We use the notation $\langle t, \{f_1 : v_1, \ldots, f_n : v_n\}\rangle$ for a TFS where t is the type and $\{f_1 : v_1, \ldots, f_n : v_n\}$ the set of feature-value pairs of the TFS. The notation $f \in t_i$ indicates that the feature f is defined for the type t_i.

We first define the *assimilation*[4] of a TFS to another TFS. Informally, the assimilation operates on two TFS – covering (c) and background (b). There are two cases: If the covering is more general than the background, the type of the covering is specialized to the type of the background. The other case is where the two types are either incompatible or when the covering is more special than the background. Then the background is transformed

[4]The assimilation function was referred to as "restriction" in [AB01]. In this work we have slightly changed the definition and changed the name accordingly. The main reason for this is that the type of the resulting TFS no longer necessarily is a more general type but possibly an *incompatible* one, and "restriction" might then give a false impression of the semantics of the operation. While both approaches are equivalent, assimilation is easier to extend to multiple inheritance type structures.

to the type of the covering, possibly via the LUB of the two types. Note that assimilation not only transforms the *types* of the feature structures, at least for the background, some incompatible *features* might be removed. It does not otherwise affect the *values* of features, this takes place only in the overlay operation.

Definition 1 (Assimilation)
Let

- $c, b \in \mathcal{S}$ such that the covering $c = \langle t_c, \{f_1 : v_1, \ldots, f_n : v_n\}\rangle$ and the background $b = \langle t_b, \{g_1 : w_1, \ldots, g_n : w_m\}\rangle$

then, the assimilation of c and b, $\alpha(c, b)$, is defined as:

$$\text{If } t_c \preceq t_b\text{: } \alpha(c,b) := (\langle t_b, \{f_1 : v_1, \ldots, f_n : v_n\}\rangle, b) \tag{1}$$

$$\text{otherwise: with } g_i = LUB(t_c, t_b)$$

$$\alpha(c,b) := (c, \langle t_c, \{g_i : w_i, \ldots, g_j : w_j\}\rangle) \tag{2}$$

□

In what follows, we will assume two operations – α_c and α_b – that select the assimilated covering and background respectively. We continue with the definition of overlay for typed feature structures:

Definition 2 (overlay)
Let

- c and b be two TFS such that the covering $c = \langle t_c, \{c_1 : f_1, \ldots, c_n : f_n\}\rangle$ and the background $b = \langle t_b, \{b_1 : g_1, \ldots, b_m : g_m\}\rangle$
- $A = \alpha(c, b)$

then $\underline{\textsc{Overlay}(c, b)}$ is defined as:

$$\textsc{Overlay}(c,b) := \textsc{Overlay}'(\alpha_c(A), \alpha_b(A)) \tag{3}$$

$$\textsc{Overlay}'(c,b) := \langle t_c, \{o_i : h_i \mid$$

$$o_i = c_j = b_k, h_i = \textsc{Overlay}(f_j, g_k), \text{ where } f_j, g_k \in \mathcal{S}, \text{ or} \tag{4}$$

$$o_i = c_j = b_k, h_i = f_j, \text{ where } f_j, g_k \in \mathcal{A}, \text{ or} \tag{5}$$

$$o_i = c_j, h_i = f_j, o_i \neq b_k, 1 \leq k \leq m, \text{ or} \tag{6}$$

$$o_i = b_k, h_i = g_k, o_i \neq c_j, 1 \leq j \leq n \quad \}\rangle \tag{7}$$

□

The first case (4) is the recursive step used when the values are typed feature structures. In the second case (5), when the values of covering and background are atomic, the value in the covering is used. The next case (6) applies when the feature is absent in the background and we use the value from the covering. Finally, (7) is the case when the feature of the covering has no value: then the value in the background is used.

Note that overlay is, unlike unification, not a commutative operation. Hence we have $\textsc{Overlay}(a, b) \neq \textsc{Overlay}(b, a)$, where a and b are two TFS such that $a \neq b$, and $unify(a, b)$ would fail. However, if a and b are unifiable, then $\textsc{Overlay}(a, b) = \textsc{Overlay}(b, a) = unify(a, b) = unify(b, a)$.

2.2 Overlay for Type Systems with Multiple Inheritance

We now generalize overlay for typed feature structures where types can be constructed by multiple inheritance. Formally, we have an inheritance hierarchy $\langle \mathcal{T}, \preceq \rangle$ which is a partial order.[5] A partial order is a relation (in our case $\preceq$) that is *reflexive*, *antisymmetric* and *transitive* (see, e. g., [Kri95, p. 96]. Again, we keep the feature introduction condition requirement for the types in the hierarchy. Now, for two types there might not exist a unique LUB but a set of *minimal upper bounds* (MUBs). In figure 4 the types t_3 and t_4 have no unique LUB but two MUBs - t_1 and t_2. A common way of

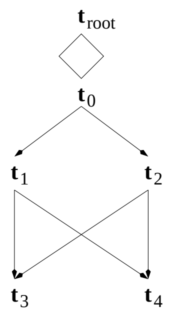

Figure 4: A sample type hierarchy.

computing the set of MUBs for two arbitrary types in the hierarchy is to use the *hereditary superset encoding* (henceforth HSE), e. g., [Grä78].[6] With

[5]Carpenter requires a finite bounded complete partial order [Car92, definition 2.1, p. 13] since (deterministic) unification requires a unique least upper bound.

[6]Grätzer actually defines the "hereditary *subset* encoding" for computing the greatest lower bound (GLB). Our version is analogous for the MUB.

this encoding we can compute the set of MUBs by taking the intersection of the sets of all super-types for the two types of the TFSs. However, this method will possibly compute obsolete, more general types (e. g., t_0 in figure 4). These are harmless with respect to the result of overlay and will merely - using a naïve implementation - cause unnecessary computation. Therefore we remove any type that is a super-type of another type in the resulting set.

The assimilation operation has to be extended accordingly. First, we define the computation of the set of all super-types for an arbitrary type in our type lattice:

Definition 3 (The set of all super-types of a type)
Let t be an arbitrary type in a type lattice then, the <u>set of all super-types of a type</u>, $\jmath\,(t)$, is defined as:

$$\jmath\,(t_i) = \{t_j \mid t_j \preceq t_i\} \tag{8}$$

□

Next, we compute the set of all MUBs for two arbitrary TFSs:

Definition 4 (The set of all minimal upper bounds for two types)
Let $t_a, t_b \in \langle \mathcal{T}, \preceq \rangle$ then, the <u>set of all minimal upper bounds</u> for t_a and t_b, $\imath\,(t_a, t_b)$, is defined as:

$$\imath'\,(t_a, t_b) = \jmath\,(t_a) \cap \jmath\,(t_b) \tag{9}$$

$$\imath\,(t_a, t_b) = \{t_i \mid t_i \in \imath'(t_a, t_b) \wedge \neg(t_i \preceq t_j \wedge t_j \in \imath'(t_a, t_b))\} \tag{10}$$

□

Equation (9) computes the set of all common super-types, $\imath'$ for t_a and t_b. In (10), all types that are super-types of some type in the set of all common super-types are removed. We end up with a set of *minimal upper bounds*. Now we can continue with the extended version of assimilation:

Definition 5 (Extended assimilation)
Let

- $c = \langle t_c, \{f_1 : v_1, \ldots, f_n : v_n\}\rangle$ and $b = \langle t_b, \{g_1 : w_1, \ldots, g_n : w_m\}\rangle$
- $t_c, t_b \in \langle \mathcal{T}, \preceq \rangle$
- $T_s = \imath\,(t_c, t_b)$ is the set of MUBs for t_c and t_b

then, the extended assimilation of c and b, $\alpha'(c,b)$, is defined as:

$$\text{If } t_c \preceq t_b\text{: } \alpha'(c,b) := (\langle t_b, \{f_1 : v_1, \ldots, f_n : v_n\}\rangle, b) \tag{11}$$

otherwise: with g_i defined for some $t_s \in T_s$:

$$\alpha'(c,b) := (c, \langle t_c, \{g_i : w_i, \ldots, g_j : w_j\}\rangle) \tag{12}$$

□

Finally, to adapt overlay for this scenario, the only thing we need to do is to replace the assimilation operation as defined in definition 1 with the extended assimilation operation as defined in definition 5.

2.3 The Scoring Function

In order to clarify the notion of scoring, we briefly summarize the work of [PAB02, Pfl02]. Overlay is a non-monotonic operation that always succeeds. If our task is to validate a hypothesis against the discourse memory, we will succeed for all referents, independent from how good or bad the hypothesis fits the referent at hand. Our solution to this behavior is a scoring function which computes a metric that can either be used for ranking several possible hypotheses, or given a threshold, even disqualify the result of the operation.

In [AB01] the scoring function was based on simple heuristics consisting of a combination of the amount of information contained in the (combined) output and the distance in discourse history. In [PAB02, Pfl02] the scoring function has received a more intuitive and formal design. For the new version of the scoring function we collect during the overlay operation a number of values as indicated below. The numbers in brackets indicate in which equation the respective variables are incremented.

co the number of values stemming from the cover (5, 6)

bg the number of values stemming from the background (7)

tc the number of type-clashes (1, 2)

cv the number of conflicting values (5)

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 function (13) whose codomain is between -1 and 1.

$$score(co, bg, tc, cv) = \frac{co + bg - (tc + cv)}{co + bg + (tc + cv)} \tag{13}$$

The positive extremal indicates two unifiable arguments whereas the negative extremal indicates that all information in the result stems from the covering. All scores between these two extremals indicate that the cover more or less fits the background; the higher the score the better the fit. For the overall discourse processing we have to take into account other, additional metrics, like the distance to the referent etc.

The use of the scoring mechanism alone is not sufficient in real applications. There is a need for additional metrics for computing how well the hypothesis fits the context. One simple but well functioning such heuristics which is used in SMARTKOM is *recency*, i. e., how "far back" the background is found or how "focused" it is. In SMARTKOM, backgrounds being further away are penalized whereas backgrounds in focus are rewarded.

2.4 An Example

To highlight the intention behind overlay, we pick up the example of the introduction. Assumed is a domain model where the classes PERFORMANCE and BROADCAST have a common super-type (let us call it BROADCASTPERFORMANCE) defining common features, such as *beginTime*.

1 **User**: I'd like to go to the movies tonight.

2 **SmartKom**: Here (↗)are the films showing in Heidelberg.

3 **User**: No, there is nothing for me,

4 **User**: what is running on TV?

Utterance 1 is represented with following structure:[7]

$$\left[\ldots \quad entertainment: \left[\begin{array}{ll} \textsc{Performance} & \\ beginTime: & tonight \\ cinema: & \left[\begin{array}{ll} \textsc{Cinema} & \\ location: & Heidelberg \end{array}\right] \end{array}\right]\right]$$

For utterance 4 we receive the following analysis:

$$\left[\ldots \quad entertainment: \left[\ \textsc{Broadcast}\ \right]\right]$$

Now, since the LUB of PERFORMANCE and BROADCAST is BROADCASTPERFORMANCE, the *assimilation* of PERFORMANCE to BROADCAST will contain, amongst other things, the *beginTime* feature (but not the *cinema* feature):

[7]The feature *location* is set to *Heidelberg* because of the situative context

$$\left[\ldots \quad \mathit{entertainment}: \begin{bmatrix} \text{BROADCAST} & \\ \mathit{beginTime}: & \mathit{tonight} \end{bmatrix}\right]$$

The BROADCAST thus inherits the time information, and overlay produces the structure:

$$\left[\ldots \quad \mathit{entertainment}: \begin{bmatrix} \text{BROADCAST} & \\ \mathit{beginTime}: & \mathit{tonight} \end{bmatrix}\right]$$

The system can thus retrieve the films broadcasted on TV for tonight, and does not need to ask the user about the time.

3 Generalizing Overlay

Until now, we have considered changes (restrictions) in the type of the background as the only means to exclude information from the background in the result of overlay. Consider the following example:

(5) **User**: What's on TV tonight?

(6) **System**: Here ($\nearrow$) is an overview of tonight's TV program.

(7) **User**: OK, what's on channel three?

(8) **System**: This ($\nearrow$) is tonight's program of channel three.

(9) **User:** Oh well, what's on TV tomorrow night?

The intention of the user in this situation is ambiguous. She might be asking for the program of channel three or a general overview. The type-based assimilation mechanism of overlay does not apply in this example, since the final utterance clearly is of type BROADCAST, but nevertheless the channel slot of the background might or might not be relevant. In the strict interpretation of overlay, it would be added to the intention. To extend overlay to also generate the general interpretation, the definition must be extended as follows: For every feature that is not set in the cover, generalized overlay has two valid results, with or without the information from the background (i.e., formally the result is a set with all possibilities).[8]

[8] One might argue that the last utterance is really a reference back to the initial utterance *What's on TV tonight?* and thus overlay should not take the current context as the background but rather "close" that node in the discourse history and go back to the initial utterance. Such an approach would be more difficult to apply to a discourse where the initial two utterances are **User:** *What can I do tonight?* and **System:** *May I propose an overview over the TV program?*

Another case in which a user utterance might have to be interpreted in a more general way affects the definition of overlay in a more direct sense. If, in the above example, the final utterance by the user is instead *What movies are there tomorrow night?* then the strict interpretation is that it refers to watching TV. Since the user has not mentioned watching TV explicitly, the utterance is ambiguous and a generalized interpretation would be *What movies are playing in movie theaters or on TV tomorrow night?* Provided that such an interpretation of type BroadcastPerformance is generated by the analysis modules, overlay as defined in step (1) of definition 1 of assimilation would always refine the type to Broadcast. Thus, in the same manner as discussed above for using values from the background or not, the assimilation of a cover that is of a more general type than the background must be extended to return a set of all possibilities. Step (1) could be reformulated as

$$\alpha(c,b) := \{(\langle t_g, \{f_1 : v_1, \ldots, f_n : v_n\}\rangle, b)\} \text{ where } t_c \preceq t_g \preceq t_b \tag{14}$$

It applies similarly to step (11) of definition 5 of assimilation in multiple-inheritance type systems. Note that steps (2) and (12) are not affected since the type of the covering is not changed there.

4 Conclusion

We have provided an extended formalization of overlay as an operation that combines new and old information that is encoded as typed feature structures in such a way that preference is given to new information but as much as possible of the old information is added to the result. In particular, we use this information in dialog systems where a new user utterance must be combined with old information from the discourse history. Overlay hinges on a typed, frame-based representation framework. We have given a strict definition of overlay that keeps all possible old information for tree-structured as well as multiple-inheritance networks. In order to cover cases where not all compatible old information is desired, the generalized version of overlay as discussed in the last section applies, as it spans the search space of possible interpretations.

We believe that overlay is a very general operation that can be applied in to many phenomena. In [PAB02], we have already laid out how overlaycan be used to resolve anaphora and we are currently exploring the use of overlay for intra-sentential phenomena like gapping and one-anaphora.

References

[AB01] Jan Alexandersson and Tilman Becker. 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 2001.

[BFL98] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet Project. In *Proceedings of COLING-ACL*, Montreal, 1998.

[Car92] Bob Carpenter. *The logic of typed feature structures.* Cambridge University Press, Cambridge, England, 1992.

[GMP03] Iryna Gurevych, Stefan Merten, and Robert Porzel. Generating natural language processing interfaces from ontologies. Under submission, 2003.

[Grä78] George Grätzer. *General Lattice Theory*, volume 52 of *Mathematische Reihe.* Birkhäuser, Basel, 1978.

[Kri95] Hans-Ulrich Krieger. *$\mathcal{TDL}$—A Type Description Language for Constraint-Based Grammars. Foundations, Implementation, and Applications.* PhD thesis, Universität des Saarlandes, September 1995.

[LBPA02] Markus Löckelt, Tilman Becker, Norbert Pfleger, and Jan Alexandersson. Making sense of partial. In Bos, Foster, and Matheson, editors, *Proceedings of the sixth workshop on the semantics and pragmatics of dialogue (EDILOG 2002)*, pages 101–107, Edinburgh, 2002.

[PAB02] Norbert Pfleger, Jan Alexandersson, and Tilman Becker. Scoring functions for overlay and their application in discourse processing. In *KONVENS-02*, Saarbrücken, September – October 2002.

[Pfl02] Norbert Pfleger. Discourse processing for multimodal dialogues and its application in smartkom. Master's thesis, Universität des Saarlandes, Saarbrücken, 2002.

[PG02] Robert Porzel and Iryna Gurevych. Towards context-adaptive utterance interpretation. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialogue*, pages 154–161, Philadelphia, July 2002. Association for Computational Linguistics.

[RN95] Stuart J. Russell and Peter Norvig. *Artificial Intelligence. A Modern Approach.* Prentice Hall, Englewood Cliffs, N.J., 1995.