THE NONDIRECTIONAL REPRESENTATION OF SYSTEMIC FUNCTIONAL GRAMMARS AND SEMANTICS AS TYPED FEATURE STRUCTURES

John A. Bateman*
Project KOMET, GMD/IPSI
Darmstadt, Germany
bateman@darmstadt.gmd.de

Martin Emele and Stefan Momma**
Project POLYGLOSS, IMS-CL
University of Stuttgart, Germany
{emele,momma}@informatik.uni-stuttgart.de

Abstract

A small fragment of the Systemic Functional Grammar of the PENMAN system is reformulated in the Typed Feature Structure language. Through this reformulation we gain full reversibility for the SFG description and access for unification-based grammar descriptions to the rich semantic levels of description that SFG supports. We illustrate this reformulation with respect to both generation and semantic analysis and set out the future goals for research this result establishes.

1 Introduction

The current state of the art in natural language processing calls for components capable of sophisticated deep semantic analysis, modular representation of resources, and re-usability of those resources across different NLP applications. Furthermore, it has been demonstrated that the sheer diversity of interactions between distinct kinds of linguistic information is guaranteed to defeat any staged approach to generation/understanding that successively maps between levels of representation [1]. One strategy for addressing these problems is to *stratify* resources so that inter-stratal mappings are simplified. This is aimed at allowing high-level information to apply as early in analysis as possible at a minimal cost. A number of current processing architectures call for such a design. The stratification technique is also one way of ensuring modularity and improved reusability. However, one important problem with almost all existing linguistic resources is that the inter-stratal mappings between, for example, strings and semantics, are anything but simple. This is because the standard syntax-semantics-pragmatics modularization *under*-stratifies by imposing fewer distinctions than are necessary.

Computational systemic-functional grammars [4] offer significant solutions to this deficiency by imposing a higher degree of stratification (cf. Section 3), thus keeping inter-stratal relations simple. SFGs have supported the construction of natural language generation components that are able to take highly abstract, purely semantic specifications as input and produce corresponding surface strings as output. Furthermore, the generation task has required extensive work on the more abstract strata: without a rich breadth of communicative goals, grammatical resources for expression cannot be satisfactorily constrained.

Problematic with current computational instantiations of SFG, however, is that implementations have been directionally committed: the higher strata of information have not been available for constraining lower level analyses. This problem has been compounded by a further lack of detail at the lower, syntagmatic/constituency stratum in SFG. In contrast to the generation perspective, work oriented towards analysis — particularly within current information-based grammars such as LFG and HPSG — has paid extensive attention to the less abstract strata of the linguistic system and have produced highly detailed accounts of syntagmatic organization. A combination of the two paradigms should enable us to fill gaps in the respective coverage of strata.

Information-based grammars have already been presented using declarative representations such as feature structures. In this paper, we present a formalization of all the information to be found in the strata of computational SFG and their inter-stratal mappings in terms of feature structures also in order to initiate a combined description of the two paradigms. In particular, we will formalize SFG in terms of the Typed-Feature System (TFS) developed within the POLYGLOSS project at Stuttgart. TFS has previously been applied to the strata of the linguistic system addressed by HPSG and LFG. Here, we argue further that it is appropriate to adopt this representation for SFGs and the higher strata of the linguistic system that SFG proposes. The result of this inquiry is then to provide the basis for populating a highly-stratified processing architecture with maximally appropriate linguistic representations. Moreover, the higher levels of abstraction are rarely dealt with *within* a theory that talks about syntax. Rather, their treatment is left to a separate process using a separate representation language and a completely separate processing strategy. Our approach offers a solution to this problem as well by providing a unique framework to talk about (almost) *all* levels of linguistic information in a single formal language using a single processing scheme.

2 A convergence of concerns

We now briefly introduce both SFG and TFS; substantial introductions are provided elsewhere and so here we shall only offer sufficient details to understand the examples offered in the paper.

2.1 The SFG framework

Analogously to HPSG and LFG, SFG belongs to the family of grammatical frameworks based on statements of the co-occurrence possibilities of grammatical features. In contrast, however, SFG places central focus on grammar as a complex resource for achieving communicative and social goals. Within SFG the entire grammatical description of a language is organized around a factorization of the task of finding grammatical structures appropriate for the expression of specified meanings and it is this orientation that has made it so appealing for the task of generation.

The organization of the PENMAN-style architecture within which the SFG of English we are working with is embedded decomposes the mapping from abstract semantics to surface string as follows. Nearest the surface there are *realization statements* of syntagmatic organization, or syntactic form. These are

*Also on indefinite leave from the Penman Project, USC/Information Sciences Institute, Marina del Rey, Los Angeles.

**Supported by BMFT Grant No. 08 B3116 3

classified in terms of a *grammatical system network* that denotes the paradigmatic, functional alternatives offered by syntactic forms.

The decisions in the grammatical systems network are motivated by semantic distinctions that classify semantic circumstances according to the grammatical features which are appropriate to express those situations: this classification is the combined responsibility of *choosers* and *inquiries*. Finally, the possibilities for classification that the inquiries have are defined in terms of an abstract ontology called the *upper model*. Knowledge about particular domains is subordinated to the upper model so that the concepts of those domains can inherit their possibilities for linguistic realization from those already specified for the upper model concepts. Also, lexical information is typically associated with the domain concepts.

All of these components are currently implemented and used within the PENMAN and KOMET projects. The upper model and domain model are implemented in the knowledge representation language LOOM (developed at USC/ISI); the remaining theoretical constructs involved in the generation process are realized as macros defined in CommonLisp. These latter are, therefore, in implementation strictly procedural and do not support analysis. For further details of the underlying theory and its application in the area of text generation see [4].

2.2 The TFS framework

The TFS formalism is a rewriting system for typed feature terms following ideas from [5]. The major goal when designing TFS as a grammar description language was to develop a formalism that inherently supports both modular, hierarchically organized descriptions and a declarative formulation of relationships between (partial) elements from different linguistic modules.

The objects used to represent linguistic information are **typed** feature terms, i.e., feature terms that have a *type symbol* associated with each node in the directed graph representing an ordinary feature term. A linguistic description consists of a set of *feature type definitions* which contain information about the placement of the feature term in the type hierarchy and about the set of well-formedness constraints that hold for this particular type. The feature type definitions define the well-formedness constraints for *all* representation levels, simultaneously specifying what constitutes a 'well-formed linguistic sign', i.e., an object containing *all* the information that can be collected about an utterance — be it analyzed (where the phonological form of the utterance is the input), or generated (where parts of the 'content' of the complete description serves as the input); an example of such an object appears below in Figure 3.

Given a partial description (a feature term with (possibly partial) type information) as input, the interpreter computes the set of most specific feature terms derivable from that term by first classifying the term according to the features it contains and their values, and subsequently recursively applying feature type definitions to all type symbols contained in the term. Only one operation is used: Rewriting based on unifying substitutions of feature terms. For full details of the formalism and its implementation see [6, 7]. Since the TFS language is fundamentally a constraint-based language, and none of the operations involved is dependent on a particular order of, e.g., the availability of certain pieces of information, no directionality is involved in the constraint-solving process. Thus, a successful encoding of the information contained in a systemic grammar description in TFS will of necessity be strictly bidirectional. In fact, the specification of well-formed linguistic signs *simultaneously* for all strata represented means that the term *non-directionality* is a better characterization of the distinctive property of this kind of system than, for example, "reversibility".

3 Modelling of SFG in TFS

We now describe in some detail how each of the strata, and the relations between them that are defined within computational SFG, may be defined uniformly within TFS. This supports the general claim made within SFG that statements of linguistic resources *should* be non-directional. We will begin with the least abstract stratum — the system network — and work up to the most abstract level — the upper model. We then describe the inter-stratal mappings that exist between these.

3.1 The System Network

A system network can be represented as a directed acyclic graph with labelled arcs whose nodes are choice points called *systems* and whose outward directed labelled arcs denote the *terms* of the system. Each system has two or more terms, or output features, which at the stratum of grammar represent minimal grammatical alternations. Reflecting the extensive stratification of the architecture, nothing is said in the system network proper about *how to choose* between these alternations. The inward directed arcs for each system denote an *entry condition* which determines the paradigmatic context in which the alternation represented by the system is relevant. As shown already in [3], the network is formally equivalent to the specification of a subsumption hierarchy and so the translation of the *connectivity* of that network alone into TFS is quite straightforward. The result is a lattice with most general type RANK.

The system network does not itself generate grammatical structures; this information is also stratified out. In SFG grammatical structure is built up in terms of 'syntagmatic specifications' which are derived by means of *realization statements* associated with paradigmatic selections. Kasper [2] shows how these realization statements can be represented in terms of feature structure descriptions. We largely adopt this representation here. The possible realization statements are listed in the table shown below, together with their translations into TFS feature terms.

Insert:	+F	[F:RANK]
Expand:	$F_1(F_2)$	[F_1:[F_2:#1=RANK], F_2:#1]
Inflectify/ Classify:	$F_1::l_1$	[F_1:[l_1:+]]
Conflate:	F_1/F_2	[F_1:#1, F_2:#1]
Preselect:	$F_1:f_1$	[F_1:f_1]
Lexify:	F!l	[F:lex-name]

An Insert statement (+F in traditional systemic notation) corresponds to the introduction of the feature F into the feature term, at the same time classified as requiring a value of type RANK. The Preselect, Classify/Inflectify and Lexify statements have similar translations, the only difference being that they do not only determine the *type* of the value of the feature they introduce, but also specifically determine the value itself; lexify is then the limiting case where a type drawn from the sublattice of lexical items may be specified directly. A second class of statements is used to express co-labelling of structures. Conflate (F_1/F_2) expresses the identity between two features in a complex structure. In the feature structure description we denote this by introducing a reentrancy. The way TFS denotes reentrancies is by use of a *tag* (e.g. #1) that appears at the two endpoints of the paths that are being made reentrant.

3.2 Upper Model and Domain Model

The representation of the Upper Model in LOOM is also straightforward to translate into the TFS notation. Every concept in the hierarchy corresponds to a type in TFS, where the content of the :is slot corresponds to the specification of the appropriate supertype for the given concept.

There are two possible ways to build up the hierarchy of concepts for the Upper Model: we can follow a pure top-down strategy, specifying via stepwise refinement what subconcepts make up a given concept. This is appropriate whenever the LOOM definition contains a statement about :disjoint-covering. The second possibility is to build up the hierarchy bottom up, i.e., for a type we specify what its supertype is. This is mostly used when the type-definition has additional constraints, which are then conjunctively conjoined to the supertype specification, thus refining the definition of the current type. An example for such a translation is shown in Figure 1. The result of the translation is a type lattice with most general type UM-THING.

```
(defconcept Process
      :is (:and UM-Thing :primitive)
      :constraints (:at-least 1 Participant)
      :disjoint-covering
          (Material-process Mental-Process
           Relational-Process Verbal-Process))
                            ⇓
PROCESS < UM-THING.
PROCESS =  MATERIAL-PROCESS | MENTAL-PROCESS
        |  RELATIONAL-PROCESS | VERBAL-PROCESS.
PROCESS[participant: UM-THING].
```

Figure 1: LOOM definition and TFS definition for the concept Process

Subsequently, semantic specifications — isomorphic to the Sentence Plan Language (SPL) standardly used as input to PENMAN-style text generation systems or to a set of LOOM A-Box assertions — are then defined by a type LOCAL-PLAN which specifies the semantic process, semantic participants, and certain details of textual presentation and interpersonal effect. The semantic specification (simplified for illustration purposes) for the sentence *Kim devours every cookie*, for example, can be seen under the sem attribute in the feature structure shown in Figure 3. In this expression, DEVOUR, COOKIE, and KIM are concepts in the domain model that are subordinated to types defined in the upper model in the standard way defined for interfacing with PENMAN-style systems. favor, set-totality-individuality are semantic correlates of textual inquiries concerning the communicative functions of referring expressions. speechact is the semantic correlate of an interpersonal inquiry concerned with illocutionary force. A full specification would contain many more details (cf. [4]).

3.3 Choosers and Inquiries: interstratal relationships

As noted in Section 3.1, the systemic network alone does not specify the semantic motivations for selections of particular grammatical features from the network. This task is handled in PENMAN-style implementations of SFGs by the stratum of the *chooser&inquiry* semantics. Choosers are typically described, and used in generation, as decision trees, and one such tree is associated with each grammatical system in the system network. This rather simple organization can also, however, be straightforwardly interpreted in terms of the semantic conditions of applicability that choosers define for each grammatical feature. This provides for a declarative modelling in TFS as follows.

The decisional components of the decision tree are *branching inquiries*. In the original procedural implementation of the PENMAN system, a branching inquiry takes as argument a semantic entity (identified via the grammatical label for the constituent realizing that entity) and returns a value from a closed set of possible responses that are defined for each inquiry. The chooser then provides a series of actions that are to be followed for each possible response, analogously to a case-statement in Lisp.

Our encoding of this level of organization in TFS moves away from the implementation in PENMAN by making use of the fact that choosers are themselves hierarchically organized. While in PENMAN this organization is maintained only indirectly by the association with grammatical systems, in TFS we define the sublattice explicitly using types defined for choosers. There is then no need for the branching inquiries since chooser decision trees may be directly folded into the hierarchy and their possible outcomes are represented as distinct types.

In PENMAN the arguments required for the branching inquiries are picked out in a way that depends on another type of inquiry: an *identifying inquiry*. The function of these is precisely to locate particular semantic entities with respect to semantic entities that are already known. It is clear that for these inquiries to be implemented, a specific semantic representation must have been selected. We have, for the time being, folded this information into the TFS translation: that is, we use the concrete implementations of identifying inquiries (which are rather simple) to fix the particular path that we specify as a value for the sem attribute. Identifying inquiries are used in chooser actions of the following form: (identify F_1(inq-ID F_2)). This specifies that the semantic entity returned by applying the inquiry inq-ID to the semantic entity associated with the grammatical constituent labelled as F_2 is associated with the grammatical constituent labelled as F_1. In the TFS translation, wherever mention is made to a semantic entity by means of these grammatical constituents, we instead pick out the semantic entity directly by incorporating sufficient path information in the partial feature structure description in the sem slot. The translation of the above identify action is:[1]

```
        [syn:[F1:#S2], sem:[F1:[...[ #S1]...] ]]
         :- inq-ID[syn:#S2, sem:#S1].
```

where, as explained above, the precise path under the first sem slot is defined by the implementation of the inquiry inq-ID with respect to the upper and domain models. For the identification: (identify Gr-actor (actor-id Gr-process)), for example, the corresponding TFS term is:

```
[syn:[Gr-actor:#S2], sem:[process: [actor:#S1]]]
        :- actor-ID[sem:#S1, syn:#S2].
```

Inquiries of this type are necessary since they provide an additional interface structure between actual upper model concepts and objects in the system

[1] Using Prolog's "neck" symbol to introduce the condition.

network. Subsequently, the relationships they define between the grammatical and ontological sublattices are folded into the types of the chooser sublattice directly as described above.

Finally, the concrete statement that a particular grammatical feature is entailed by the semantic circumstances that pertain is made by the choose chooser action which takes as parameter the grammatical feature from the grammatical system to which the chooser is assigned that is to be selected. This action is trivially represented by adding in the grammatical feature as a type constraint on the syn side of the relation at the appropriate chooser subtype; i.e., (choose GRAM-FEATURE) ⟹[syn: GRAM-FEATURE].

Choosers as a whole then form a sublattice whose most general type is RANK-CHOOSER. Figure 2 shows an example translation of two chooser nodes from this sublattice, where we see the above translation principles at work.[2]

```
((ASK (STATIC-CONDITION-Q GR-PROCESS)
 (STATIC
    (ASK (MENTAL-PROCESS-Q GR-PROCESS)
    (MENTAL
       (IDENTIFY GR-ACTOR (SENSER-ID GR-PROCESS))
       (CHOOSE MENTAL))
    (NONMENTAL ...)))))

                    ⇓

PTC-STATIC-MENTAL-CHOOSER
[sem: [process:MENTAL-PROCESS[senser:#a1]],
 syn: MENTAL[gr-actor:#a2]] :-
CHOOSER[sem:#a1, syn:#a2].

PTC-STATIC-NONMENTAL-CHOOSER
[sem:[process:
       (MATERIAL-PROCESS |
        VERBAL-PROCESS | RELATIONAL-PROCESS)]].
```

Figure 2: Translation of chooser nodes

An important point to note here is the strict separation of 'syntactic' and semantic information that is enforced. Complete modularity of the syn and sem descriptions is maintained and the choosers&inquiries are defined as a lattice of relations between these informational domains: there is no intermixing of categories within informational domains. Associations between semantics and syntax are preserved only in the conditions that specify the mappings across strata. The lattice of relations that the CHOOSER-sublattice defines permits the implicit definition of the complete cross product of the RANK and UM-THING sub-lattices. This avoids the combinatorial explosion of type symbols that would otherwise ensue. The existence of a particular subtype of sub-CHOOSER on a certain level of hierarchical embedding excludes all others on that level that would exist had we taken the complete cross product.

4 Demonstration of generation and analysis

4.1 Generation

When we want to generate, we provide a specification of the semantic communicative functions that are to be achieved by the linguistic realization. Generation is then initiated by providing the local plan as the value of the sem attribute of the top-most chooser. After classifying the input structure according to the features it contains (already yielding a particular subtype of CHOOSER), the type of the topmost node of the input structure is then recursively expanded. Expansion is performed by rewriting all embedded types through unifying substitution of their definitions until no further rewriting is possible (i.e., until all types are ground types). Expansion terminates with a complete description compatible with the input partial description and with the definitions in the feature type system representing all the linguistic strata defined. In the general case, we will end up with not just one description, but rather with a set which is then to be interpreted as a disjunction of possible solutions to the initial problem. The complete structure which is the result of the interpretation of the semantic specification (given under the sem feature) is given in Figure 3.

4.2 Analysis

As stated in Section 1, SFG suffers from a lack of specificity in its syntagmatic representations; the kind of specifications that we find immediately underlying strings in the PENMAN and KOMET systems, for example, gives representations (expressed according to our TFS definitions of Section 3.1 above) such as the following:

```
RANK[gr-process: LEX-DEVOUR,
      subject: [thing: LEX-KIM],
      directcomplement: [thing: LEX-COOKIE,
                         deictic: LEX-EVERY]]].
```

However, information-based syntax, such as HPSG, does provide extensive detail at this level. Now, due to the strict modularity enforced in our translation, it is possible to explore combinations of approaches and, moreover, to combine descriptions from a theory like HPSG with the kind of descriptions employed in Systemic Linguistics or its Computational instantiations. This has been shown to be possible in a simple experiment carried out by Martin Emele where an existing HPSG grammar was taken and the *semantics* of that grammar (a simple situation semantics-informed frame-like representation) was rewritten to give the syntagmatic categories and structures of the SFG. This makes it possible to describe the information obtained from the two approaches within a *single* executable declarative specification. Here, however, our main concern has been with making available the higher-levels of specification, and so we will abstract away from the string to syntagmatic structure component of the mapping and take as the 'input' specification the lowest level of information obtained from the SFG, as shown above. Therefore, we proceed by putting this specification in the syn slot of the RANK-CHOOSER relation. Term rewriting applies to construct the sem side of the relation and also to complete the syn specification. The result is again the complete specification of the set of constraints that describe the structure, which is again the structure shown in Figure 3. This is precisely the same linguistic-sign that was produced as a result of "generation", starting from the pure semantic part of the description — thus illustrating the radical *nondirectionality* of the representation.

5 Conclusions and Future Work

The close fit between the linguistic description required in a TFS-based architecture and those being pursued within SFG have motivated a detailed investigation of the mutual compatibility of the representational and formal mechanisms and linguistic

[2] The second type definition gives a statement of the negative condition, which is presently represented by a disjunction of the categories defined in the upper model as sisters of MENTAL-PROCESS; future versions will rely on negation.

$$
\begin{array}{l}
\textbf{PTC-NONSTATIC-NONVERBAL-NONMENTAL-CHOOSER}\\
\left[\begin{array}{ll}
\text{sem: } \textbf{LOCAL-PLAN} & \left[\begin{array}{ll}
\text{process:} & \textbf{DEVOUR}\left[\begin{array}{ll}
\text{actor:} & \textbf{KIM}\left[\text{favor: } +\right]\\
\text{actee:} & \textbf{COOKIE}\left[\begin{array}{ll}\text{minimal-attention:} & -\\ \text{favor:} & -\\ \text{set-totality-individuality:} & \textbf{collection}\end{array}\right]
\end{array}\right]\\
\text{speechact:} & +
\end{array}\right]\\
\text{syn: } \textbf{MATERIAL} & \left[\begin{array}{ll}
\text{gr-process:} & \boxed{3}\textbf{LEX-DEVOUR}\left[\begin{array}{ll}\text{spelling:} & \textbf{"devour"}\\ \text{presentform:} & +\\ \text{thirdperson:} & +\\ \text{singular:} & +\end{array}\right]\\
\text{gr-actor:} & \boxed{2}\textbf{INDIVIDUAL-NAME}\left[\begin{array}{ll}\text{thing:} & \textbf{LEX-KIM}\left[\begin{array}{ll}\text{spelling:} & \textbf{"Kim"}\\ \text{common:} & -\\ \text{noun:} & +\end{array}\right]\\ \text{case:} & \textbf{NOM}\end{array}\right]\\
\text{subject:} & \boxed{2}\\
\text{finite:} & \boxed{3}\\
\text{mood:} & \textbf{MOOD-UNIT}\left[\begin{array}{ll}\text{subject:} & \boxed{2}\\ \text{finite:} & \boxed{3}\end{array}\right]\\
\text{directcomplement:} & \boxed{4}\textbf{EVERY}\left[\begin{array}{ll}\text{deictic:} & \textbf{LEX-EVERY}\left[\begin{array}{ll}\text{singular:} & +\\ \text{spelling:} & \textbf{"every"}\end{array}\right]\\ \text{thing:} & \textbf{LEX-COOKIE}\left[\begin{array}{ll}\text{spelling:} & \textbf{"cookie"}\\ \text{singular:} & +\\ \text{common:} & +\end{array}\right]\\ \text{case:} & \textbf{OBLIQUE}\end{array}\right]\\
\text{theme:} & \boxed{2}\\
\text{medium:} & \boxed{4}\\
\text{goal:} & \boxed{4}
\end{array}\right]
\end{array}\right]
\end{array}
$$

Figure 3: Feature structure for "Kim devours every cookie."

descriptions being developed within the PENMAN, KOMET and POLYGLOSS projects. We have shown that systemic-functional grammars and semantics can easily be converted to the TFS formalism. This has produced a fragment that can both generate and analyse. Furthermore, the analysis achieved with our experimental fragment supports the mapping from surface representation to deep semantic levels of representation that are far removed from the contingencies of surface syntax. These representations also preserve breadth, in that the semantic distinctions necessary for generation concerning 'pragmatic' information such as textual organization and interpersonal communicative goals are also recovered. It is especially important that all of these diverse levels have now been made accessible for analysis within a system where there is only *one* representational formalism and only *one* interpretational device operating on the representations.

This paper has described and motivated the basis for a host of important further research questions, some of which we are now following up. For example, the fragment we have illustrated here is very small: the problem of handling large lattices needs to be addressed both on implementational and theoretical levels. A full specification of the grammar component of PENMAN alone as we describe it here would involve tens, possibly hundreds, of thousands of types: this needs to be supported by sufficiently powerful and robust implementations. But on the theoretical level, there are also further modularities within the SFG account that we have not yet utilized to constrain term explosions due to forming cross-products across sublattices: two areas here clearly present themselves — stronger modularization according to the paradigmatic/syntagmatic dimension and according to *functional regions* in the grammar [4], which already provide a meta-level of organization across sublattices that remains unused. A further area is a closer study of the similarities and differences between, e.g., the information of the SFG and the HPSG modules — it is to be expected that there is currently duplication which could be more effectively distributed, perhaps providing a more effective TFS translation. Finally, the availability of a representation of some systemic-functional grammars in a standard formalism should further facilitate comparison and evaluation of the grammatical description with respect to other current computational accounts of grammar: it should be more straightforward to identify the distinctive features and claims of the approach, thus opening the door to an easier exchange of information and analyses. Further, performing the TFS translation for the entire PENMAN grammar would provide an effective test of the TFS formalism (and its implementation) overall since there are no comparable grammars (i.e., paradigmatic feature based without a phrase structure skeleton) of this size available elsewhere.

References

[1] Martin Emele, Ulrich Heid, Stefan Momma, and Rémi Zajac. Interactions between linguistic constraints: Procedural vs. declarative approaches. *Machine Translation*, To appear in special issue on generation.

[2] Robert T. Kasper. An Experimental Parser for Systemic Grammars. In *Proceedings of the 12th International Conference on Computational Linguistics, August 1988, Budapest, Hungary, 1988.*

[3] Robert T. Kasper. Unification and classification: an experiment in information-based parsing. In *Proceedings of the International Workshop on Parsing Technologies*, pages 1–7, 1989. 28-31 August, 1989, Carnegie-Mellon University, Pittsburgh, PA.

[4] Christian M.I.M. Matthiessen and John A. Bateman. *Text generation and systemic-functional linguistics: experiences from English and Japanese.* Frances Pinter, London, 1991.

[5] Hassan Aït-Kaci. An Algebraic Semantics Approach to the Effective Resolution of Type Equations. *Theoretical Computer Science 45*, 293–351.

[6] Martin C. Emele and Rémi Zajac. A Fixed Point Semantics for Feature Type Systems. In: Stéphane Kaplan and Mitsuhiro Okada (eds.): *Proceedings of the 2nd International CTRS Workshop*, Montreal, Canada, June 1990. Heidelberg: Springer 1992, LNCS 516, pp. 383–388.

[7] Martin Emele and Rémi Zajac. Typed Unification Grammars. In: *Proceedings of the 13th International Conference on Computational Linguistics (CoLing 90)*, Helsinki, 20 – 24 August, 1990.