EAGLES

Expert Advisory Group on
Language Engineering Standards

EAGLES

Report on the EAGLES Workshop on Linguistic Adequacy of Linguistic Formalisms for NLP

EAGLES Document EAG–FWKS–2

Version of Sep, 1994

Contents

1 Author

Dominique Estival
ISSCO, Université de Genève
54 rte des Acacias, CH-1227 Genève
estival@divsun.unige.ch

2 The Workshop

As planned during the kickoff meeting of the EAGLES Working Group on Grammar Formalisms in Saarbrücken (March 1993), a Workshop on "The Linguistic Adequacy of Linguistic Formalisms for NLP" was organized by Dominique Estival (coordinator of the Linguistic Adequacy subgroup), with the help of Klaus Netter, who was also organizing a workshop on "Grammar Engineering" during ESSLLI'93 in Lisbon. Thanks to the help of the organizers of the European Summer School on Language, Logic and Information, this EAGLES workshop could take place in Lisbon, at the end of ESSLLI'93 on August 28-29th 1993.

3 Introduction: Goals of the workshop

A number of key researchers in the field were nominally invited to make a presentation or at least participate in thematic discussions. The number of participants was kept low primarily to foster fruitful exchanges and to promote discussion that could lead to concrete conclusions. In addition, budgetary considerations prevented offering travel subsidies, which also contributed both to the small scale of the event and to its being linked to another event at which most of the people we wanted to include would already be present.

The goals of the workshop were to provide key researchers in the field an opportunity for brain-storming on the definition of the concepts involved in the question of the linquistic adequacy of formalisms for natural language processing. We expected each participant both to draw on their experience to provide information on the way some of these concepts are already realized in actual grammar formalisms and to bring in their own opinions on the way these concepts should be realized in future formalisms.

Thus we asked participants to address the following theoretical key topics:

- How do we determine linguistic adequacy?

  Adequacy of the formalism is not the same as coverage of a grammar written in that formalism. How can we distinguish between *adequacy with respect to the language data* and *adequacy with respect to a linguistic theory*?

- From the point of view of the coverage of the grammars: are the existing formalisms for NLP *expressive* enough?

  Although the problem of testing is not central to our theme, since there is an EAGLES Working Group on "Evaluation and Assessment", we also have to adress the question of measuring adequacy.

- Are the formalisms committed to a particular theory? Up to what degree does the formalism of your choice allow you to *express generalizations*?

  For instance, levels for thematic roles (why don't some formalisms use them more, although they are expressible?); word order (how natural is it to treat a free word-order language?) ; binding principles (do the same principles allow one to treat different phenomena, or does one need separate accounts for them?).

- Taking the position that formalisms are indeed powerful enough, how can we reduce *complexity* in the face of problems which increase complexity? Can we process large grammars with such complex formalisms?
- From a processing point of view: How can *weights* be used? Are weights obtainable by training? Can we *induce grammars from input*?
- On the question of *degradation* of data: How do existing formalisms handle unforeseen input? What counts as ill-formed? Are there several levels of ill-formedness?
- Questions of *modularity*, *scalability* and *reusability*, both in theory and in practice.

Each of these topics can be articulated on two levels, as questions of competence vs. performance: adequacy at the level of formalisms and complexity of formalisms vs. the level of specific applications and adequacy from the point of view of processing.

List of Participants

The following people attended the workshop (those who gave a presentation are marked with a P):

Anne Abeillé (Paris-VIII) P
Sergio Balari (UA Barcelona)
Jochen Dörre (U Stuttgart)
Dominique Estival (ISSCO, Geneva)
Claire Gardent (U Amsterdam) P
John Griffith (U Tübingen) P
Esther König-Baumer (U Stuttgart) P
Mark von der Kraan (OTS Utrecht)
Hans-Ulrich Krieger (DFKI, Saarbrücken)
Lori Levin (CMU, Pittsburgh) P
Suresh Manandhar (U Edinburgh) P
Detmar Meurers (U Tübingen) P
Klaus Netter (DFKI, Saarbrücken)
Fernando Pereira (AT&T, New Jersey)
Fabio Pianesi (IRST, Trento) P
Mats Rooth (U Stuttgart)
Herbert Ruessink (OTS Utrecht) P
Louisa Sadler (U. Essex)
Roland Seiffert (IBM Germany, Heidelberg)
Antonio Sanfilippo (Sharp European Lab, Oxford) P

4 Presentations

The format of the workshop was to have a small number of presentations, each not more than 20 minutes, on a topic leading to general discussion of some of the specific issues.

Following the order of the presentations during the workshop, we take them in turn in this section of the report. We first give the abstract sent by the speaker before the workshop, and then a transcription of the notes taken by another participant during both the presentation itself and the discussion following it. These notes were later transcribed by the organizers, but unfortunately, we did not manage to retrieve all of them. Moreover, since these notes were taken by different participants, their style is not always consistent and the reference to individuals is sometimes difficult. However, the point of including them is not to attribute statements to particular people, but to give a sense of the discussions and to show the arguments that were presented.

4.1 Herbert Ruessink (OTS Utrecht)
The Reusability of Grammatical Resources project

Abstract:

The LRE project "The Reusability of Grammatical Resources" is concerned with the extension of the basic ALEP formalism with"linguistic datatypes" to bridge the gap between the basic formalism and linguistic practice.

Notes:

Reusability is seen as the portability of grammars from one environment to another, in which different uses may be made of the resource. This is necessary for reasons of cost, both in starting again and also in modification of existing grammatical resources.

The idea is to provide a library of descriptive devices, datatypes, for linguists to use. Datatype libraries should have only minimal theoretical commitment - they should in principe be polytheoretic. Note: *all* the datatypes are not expected to be compatible with each other - rather the linguist is supposed to select for use a consistent subset.

Datatypes should be investigated and implemented either as extensions to ET6/1 or compiled out into the ALEP/ET6/1 core - (i.e. level 2 or 3). Level 2 extensions would be unproblematic in terms of implementation but it requires greater interaction between grammar writer and the core formalism. This requires the error and debug messages to talk to the user at the appropriate level of abstraction.

H.K. - What is a datatype?

H.R. - Anything from a tree to a global well-formedness constraint, etc....

F.P. - Is the idea: (1) a grammar formalism for grammars or (2) a programming language (processors for carrying out some task)?

H.R. - The idea is (1), but both aspects are addressed.

F.P. - The trouble is that in (1) you commit yourself as closely as possible, whereas in (2) you want to keep options as open as possible... There would seem to be a confusion of aims here, since the use of a DCG itself has nothing to do with a grammar and description/communication about grammars, and does not necessarily impede discussion.

S.M. - As many ideas as possible are being taken from constraint logic programming.

D.E./A.A. - How does this actually help a practical grammar writer? And what about reusability?

S.M. - Well, the trouble is, otherwise, the user ends up discussing the *encoding* of the construct.

F.P. - Yes, the notion of abstract types is important, but shouldn't one just provide a language which permits people to define their own datatypes. So, isn't what we need better programming languages rather than formalisms?

K.N. - But having things like typed AVLs means you can put together your own linguistic descriptions without being necessarily tied to HPSG, LFG etc.

H.R. - They have looked at HPSG, LFG, CG and GB, but these will only get partial coverage. They started with inventorizing the dataypes from these theories and then they selected trees, variable inequalities, relational dependences, implicational constraints, global well-formedness conditions. To do GB you would need constraints as trees, needing at least most of First Order Logic. HPSG requires a set-union operation and they don't yet have a clear idea for this.

H.R. - Actually what they mean by datatype is rather 'descriptive statement'.

There was some discussion about whether one would have general constraint resolution (so as to do HPSG on the basis of constraints/principles rather than CFG rules). FP pointed out it would be rather inappropriate to try to do HPSG by principles in a formalism which is essentially CLE-based.

F.P. - Took issue with the notion that the grammar writer should be isolated from the processing, but examples were given of how one might want to hide e.g. list processing from the grammarian. But anyway, abstract datatypes will do just that for you anyway.

4.2 Fabio Pianesi (IRST) Modularity in a TFS system

Abstract:

The reusability of existing linguistic resources, in particular grammars and lexicons, is one of the main obstacles on the road to effective linguistic engineering. To put it more simply: suppose we are to develop a new grammar and that we want to reuse parts of existing grammars. What computational resources are needed that can ease the task of extracting and combining grammar parts to yield a usable one? Among the questions that are in need of clarification there are the following: Is any part of an existing grammar able to be reused, or are there restrictions that must be met for the result to be well-formed (according to the underlying grammatical formalism) ? How can we actually carve a grammatical module out of an existing grammar and use it to form a new one ? How can we guarantee that consistency is maintained when merging of grammatical modules, each from a different origin, is performed ? In this talk we will report on work on these topics that is ongoing in the GRAAL-EUREKA project, the main purpose being not to propose final solutions but, rather, to highlight problems and stimulate discussion. Our considerations will be developed within the typed feature logics (Carpenter, 1992 (**?**)) framework; an attempt will be made at pointing at the contribution that concepts and notions drawn from algebraic specification languages can give to address at least some of the above open issues.

Notes: (Q: question; A: answer; C: comment)

Q. Should we translate TFS grammars into object-style specification language?

A. No, this is an analysis of how to carve out sub-grammars and bring them into other grammars.

Q. Are there existing systems in which sub-grammars can be defined and reused?

A. For instance, you may want to reuse a lexicon from HPSG to CCG.

Q. Isn't the problem that this addresses caused by the heavy use of inheritance (as opposed to rule-based systems)?

A. No, this is an attempt to understand how to carve sub-grammars in general.

C. The notion of modularity here is very different from the one in linguistics, and this approach seems to require linguistic non-modularity.

A. This kind of modularity is relatively independent from the rest of the grammar. It is not a separation into abstract principles, it is like that of software engineering, i.e. reusing.

C. This could be used in developing large grammarss and restricting interactions between parts.

A. It can be used to keep track of dependencies between modules and of modules on parameters.

C. Have you considered using existing programming language modular systems, e.g. SML?

A. Yes. Some people have looked at C++.

C. GROAN...

Q. Do you agree that there could be a conflict between the linguistic type of modularity and algebraic modularity?

A. Not necessarily. They are independent.

C. But sub-grammars will not be extractable because they arise from linguistic modules interactions.

C. Reusability of linguistic modules depends on commonality of theoretical choices.

C. Linguistic modules do not correspond to "descriptive" modules.

C. Algebraic modularity is really not interesting for GB grammars. What this is good for is boring morphosyntax and subcategorization.

C. Higher-order modules could represent linguistic modularity.

C. Grammar writers may not be able to think in such abstract terms, they may be more comfortable with descriptive modules.

C. One may get more reusability by giving up unitary theories and taking copies for new modules. Changing them may be better than trying to maintain full and theoretically neat factoring.

4.3 Suresh Manandhar (U Edinburgh) Expressivity of current formalisms

Abstract:

Within AI based knowledge representation systems, frame languages such as KL-ONE are widely employed for representation of domain specific knowledge. Frame languages are very closely related to feature logics. One direction for extending current feature logic based formalisms is to develop integrated knowledge representation languages that support both constraint based grammars and pragmatic knowledge representation. This I believe is an appropriate area for extending current typed feature logic based formalisms.

Secondly, but not unrelated to the previous point, typed feature logic based formalisms should be extended to support NL semantics. At this stage, not a lot is known about what extensions are desirable for supporting NL semantics. Some possible avenues for extending typed feature formalisms in order to support NL semantics would be:

a. Better representation of semantic structures.

   For instance, the meaning of the sentence *every donkey sneezed* would be represented in something like:

```
quant: [det: forall,
        ind: [var: x,
              rest: [ reln: donkey,
                      inst: x]]],
scope: [reln: sneeze,
        sneezer: x]]
```

   In what sense does such a representation capture the intended meaning? For instance, does the above representation imply that the set of objects denoted by *donkey* is a subset of the set of objects denoted by *sneeze*?

b. Can/should there be a consistency checking mechanism for semantic structures so that:

   (john likes mary) & sem(john does not like mary)

   could cause some kind of inconsistency?

c. Should formalisms support representation of semantic theories such as situation theory?

d. And discourse representation theory?

Notes:

Starting from the standard semantic representation of *Every donkey sneezes* in HPSG, Manandhar raises the question whether such an encoding is really desirable. It uses feature structures as mere data structures to encode semantics. This “data structure approach” requires a subsequent interpretation of the resulting semantic structures by a totally different, independent formalism.

Fundamental observation: the encoding does not imply:
“[donkey]I $\subseteq$ [sneezes]I”

This situation can be changed if (generalized) quantifiers are made an integral part of the constraint language and are given the desired interpretation:

- $\forall$ (donkey, sneeze) for *Every donkey sneezes*
- $\exists$ (donkey, sneeze) *A donkey sneezes*
- M(donkey, sneezes) *Most donkeys sneeze*
- F(donkey, sneezes) *Few donkeys sneeze*

etc., where these formulas are thought of as feature terms denoting the relevant sets of objects (donkeys and such).

Plausible equivalences/inference rules then would be e.g.:

- $\forall$ (d, s) consistent iff d $\Rightarrow$ s consistent
- $\exists$ (d, s) consistent iff d $\sqcap$ s consistent
- M(d, s) $\sqcap$ F(donkey, s) $\equiv \perp$

Having done this, we indeed get: $\forall$ (d, s) implies $[d]I \subseteq [s]I$

Manandhar calls this approach "meaning oriented representation" (as opposed to the "data structure approach" above). The central question to the linguists is now: *Do you want to put semantic reasoning directly into feature logic?* And if yes: *What form of consistency checking do you want?* (certainly not: "inconsistency always leads to failure").

Manandhar points out that his expectation for a workshop like this is that linguists make explicit their requirements of formalisms. There are a lot of possible extensions to current constraint-based formalisms. But the designers/implementers usually are not in the position to decide which ones are the most urgent ones or are unwanted at all. He then presents a list and raises the question to the audience: *Are these desirable extensions?*

- abstraction/application (λ-terms)
- generalized quantifiers
- set operations (add to a set, remove, membership)
- DRT support
- situation theory support ("support" relation, consistency checking for "support" relation)

Fernando Pereira argues that there is a minimum set of extensions required to adequately build semantic representation (at least he demands to have them).

- λ-abstraction
- function application
- scoped constants

Implementing these requires higher-order unification (but in a slightly specialized and simplified version).

4.4 Detmar Meurers (U Tübingen)

Abstract:

The idea of the talk is to report on my experience using two typed feature logic systems, TFS and CUF, to implement Carl Pollard's HPSG proposal for German (Pollard 1990 (**?**)). The two systems used incorporate very different architectures, and the linguist clearly has to base his/her encoding of the grammar on the choices made by the designers of the system. It's these choices I want to illustrate and show what kind of encoding this leads to in the case of HPSG. These choices include (for example) whether the system distinguishes between the constraint language and its relational extension, or what role the type hierarchy and the appropriatenes conditions play and how they are defined.

Notes:

Implementing HPSG in TFS and CUF made clear what some of the choice points for linguistic formalisms are:

A. the Signature can be:
   1. a Type Hierarchy:
      - standard: (strong/weak/none) partitioning; (closed/open/mixed) world.
      - axiomatic
      - special types: extensional types (with and w/out attribute) polymorphic types
   2. Appropriateness Conditions

B. Expressing constraints:
   1. as relations different from type relations
   2. no such difference

TFS has made the choices given in A.1., with a *standard, weak* partitioning and recursion in the Type Hierarchy.

CUF has made the choices given in A.1., with an *axiomatic* partitioning and recursion in the predicate definitions.

Linear precedence statements are hard to encode in TFS because of the lack of negation in the constraints. There can be no negation of feature terms.

(F. Pereira: Object-oriented inheritance is based on a very naive view of the world.)

4.5 Esther Koenig-Baumer (U Stuttgart) A study of grammar engineering in the typed feature-language CUF

Abstract:

Starting out from a few minor rules of thumb which help to keep a grammar tractable by exploiting the facilities of a typed, recursive, feature-based grammar formalisms like CUF, I want to describe some of the dimensions of design decisions which are available to the grammar writer and how they are influenced by the basic assumptions of the linguistic theory one uses and by the division of labor between the grammar proper and the grammar interpreter.

The study is based on two small, but non-trivial grammars for German, which involve e.g. head-movement. One grammar is written in HPSG-style, the other is a categorial grammar.

One claim is that both the conciseness of the grammar and the processing efficiency are enhanced if parts of what are called 'principles' in HPSG are delegated to specialized grammar interpreters (traditionally called 'parsers' or 'generators') instead of relying on a direct interpretation of the grammar by the CUF-machinery. This amounts to saying that a vanilla typed feature language gives too little support for the grammar encoding and that we should take away some of the responsibilities from the grammar writer to the grammar interpreter developer (who well might be the same person). The second claim is that these positive effects are increased if one pursues a strong lexicalization of grammar, since on the processing side the search for a derivation is bounded by the size of the input, and on the grammar encoding side there are the usual conceptual benefits of object-oriented approaches.

Notes:

Implementing both a lexicalist categorial grammar and an HPSG grammar for German in CUF.

4.6 Anne Abeillé (Paris-VIII)
Lexical Sensitivity in HPSG and LTAG (A look at extraction phenomena)

Abstract:

Topics addressed:

- Are the formalisms committed to particular linguistic analyses?

- How can our formalisms express lexical sensitivity to syntactic phenomena?

The discussion will focus on extraction phenomena (and UDC), especially in French, and will contrast a TAG treatment and an HPSG treatment.

Notes:

Lexical sensitivity in HPSG and TAG (LTAG). Both frameworks are lexicalist: most part of the grammar resides in the lexicon.

What is common to HPSG 2-3[1] and LTAG:
- no empty category
- filler-gap dependencies viewed as filler-head dependencies
- lexical rules (on predicates) for extraction of complements.

Differences between HPSG 2-3 and LTAG:

[1]HPSG 2-3 refers to Pollard & Sag (1994). (**?**)

HPSG:
- a Lexical Rule “moves” extracted element from COMP to SLASH list
- non-local SLASH feature stand between filler and all intermediate nodes.

LTAG:
- local encoding of filler-head dependency in supplementary elementary trees
- adjunction: insert matrix clause between filler and head

There is a sharp contrast between:
1) which table did you put <large NP> on _?
2) on which table did you put < large NP> _?

The use of traces (as in GB) is avoided in HPSG/LTAG for:
- questions,
- raising,
- passive.

Examples:

- Korean relativized verbs are treated in HPSG by a lexical rule giving the morphological marking and introducing a slashed argument. The same treatment is available in LTAG.
- The treatment of (optional) French stylistic inversion (*le livre que lira Marie*) is easy with a lexicalized treatment of extraction phenomena (i.e. using lexical rules).
- Extraction domain: Irish complementizer alternation

Further issues which can also be treated more easily in a lexical framework:
- extraction of idiom chunks
- adjunct extraction
- parasitic gaps

Esther: How do you handle movement in German? Do you need an (explicit) trace.

Klaus: The problem is with non head movement with adjuncts.

Anne: What processing ramifications do traces have?

Fernando: You cannot parse with gaps purely bottom up, you need some kind of top-down filtering. The problem disappears with lexicalized gaps.

4.7 Claire Gardent (U Amsterdam)
Discourse theory and Unification based grammars

Abstract:

Although unification based grammars have proved useful in developing sentence grammars, little attention has been paid to the problem of extending the unification based approach to discourse grammar. This talk surveys a number of implemented unification based NLP systems and discusses their respective adequacy for the development of a unification based discourse grammar intended to deal with such phenomena as intra and intersentential nominal anaphora, temporal anaphora and discourse parallelism. The survey is both theoretical (i.e. based on the documentation) and practical (i.e. based on practical experience) and concentrates on defining and assessing the following system properties: expressivity, usability, state of development, efficiency and extendability.

General points addressed: expressivity, theory commitment (can we express the relevant generalisations?) and practical usability of the available systems (i.e. practical efficiency) in relation to the development of a unification based discourse grammar.

Notes:

Claire Gardent presented a survey of six unification-based systems (ALE, ALEP-0, ELU, GDE, MIMO, PLEUK) and discussed their adequacy for the development of discourse grammars, both theoretical and practical.

- Parallelism: MSCD calculation.

- nominal anaphora $\rightarrow$ resolution involves the SYNT, SEM and DISC features.

- temporal anaphora $\rightarrow$ operation on lists.

On the theoretical side, there were the following issues:

- In order to be able to encode different kinds of anaphora resolution, Claire was of the opinion that it should be possible to attach program-like extensions to grammars, as it is possible to do in the ALE system.

- Claire's wishlist also included a token-identity test in contrast to unification, like the contrast between '$A == B$' and '$A = B$' in Prolog. Fernando pointed out that then declarativity would be lost.

Suresh warned that although ALE is based on typed feature structures, it is not possible to implement HPSG grammars in a direct manner. Instead ALE relies on a phrase-structure backbone.

On the practical side, there was a consensus on the fact that good debugging tools are needed, such as tracing facilities which give messages that go beyond "*unification failure* ....", and the ability to manipulate intermediate structures while tracing.

A discussion arose about the dissemination of systems. Klaus argued that for complex systems (with GUI, etc.) this is difficult because then you need to allocate resources for support, maintenance and documentation. Fernando stressed the advantages of spreading a system: feedback and accelerated development. This has proved to be the case in the programming language community.

Klaus disagreed, saying that there is a tendency in the computational linguistics community to heavily criticize systems that have been released (too) early, impeding the progress of such systems.

4.8 Antonio Sanfilippo (Sharp)
Expressing Thematic Information in a Unification-Based Grammar Formalism

Abstract:

It is now widely accepted that reference to thematic roles is crucial in expressing linguistic generalizations. In spite of this recognition, the treatment of thematic information in current unification-based grammar formalisms is somewhat disappointing. For example, both LFG and HPSG include reference to thematic roles in the lexical representation of predicates. In LFG, thematic roles are used in functional structure but are not provided with a semantic interpretation. Moreover, LFG's practice to define thematic roles as a small set of discrete notions is inadequate to cope with the complexity of predicates' lexical semantic structure. In HPSG, thematic roles are expressed in semantic representation but are conceived as verb-specific labels and encoded as attributes rather than types. Consequently, their functionality is limited, e.g. it is not possible to make thematic generalizations across argument roles and to structure the thematic domain into an inheritance lattice. In addition, thematic roles in HPSG are not referenced in the semantics of subcategorized arguments and are therefore not easily available during syntactic processing.

My proposal to redress these inadequacies is to express thematic roles as prototypical clusters of thematic entailments of verb meanings (Dowty 1991 (**?**)) within a neo-Davidsonian approach to verb semantics (Parsons 1990 (**?**)), and to reference them in the semantics of subcategorized arguments. I argue that clusters of thematic entailments are best represented as set of properties which provide a decompositional characterization of verb meanings and show how this treatment can be implemented using an HPSG-like grammar formalism.

Notes:

Lexicalization of the classification of thematic role types. Thematic roles are treated as *sorts*.

The linguistic data discussed includes anaphora, control and for morphosyntax, those generalizations which rely on θ-roles:

- resolving anaphora relies on a θ-roles hierarchy

- control clearly needs θ-role

- morpho-syntax: causative formation

Formal treatment:

- HPSG: θ-roles are attributes. They are not indexed in the syntax.
  Question:
  How to express generalizations?
- LFG: θ-roles are small sets of discrete concepts. They are used in functional structure.
  Questions:
  - what is the semantic interpretation?
  - criteria for defining θ-roles?
  – answer 1. (Dowty 1991): Definition in terms of entailment of presuppositions for a given argument.
  – answer 2. Neo-Davidsonian Semantics (Parson 1990): a θ-role is a relation between participants and events.

Follow-up to answer 2:

```
                         0



agent   loc     form     matter    inst     patient
 |      /       /        /         /         / \
 |     /       /        /         /         .....
l-cause  t-cause  m-cause  i-cause  l-change t-change m-change i-change
                                     / \
                                    /   \
                                    .....
```

- Predicates are classified in terms of the features organized in this hierarchy. Word classes are identified.
- Purpose of this classification: it provides lexical features for MT.
  Claim: it allows translation from one continuous expression to more than one discontinuous expressions.

Pereira: The use of the features to reach a θ-role is decomposed. It must involve reasoning – cannot rely simply on unification mechanisms...

Estival: It's still useful since it works well in the cases it works...

Levin: An alternative would be to map lexical items to "a meaning" which is mapped to the lexical items of the target language.

Sanfilippo: It is not 100% productive – it works in some cases, which can be useful, especially with phrasal verbs...

Pereira: You need deductive methods to take full advantage of the proposal.

4.9 Lori Levin (CMU)

4.10 Grammatical Function and Semantic Role Assignment in the ALICE-chan Japanese Instruction System (Lori Levin, Keiko Horiguchi, David Evans)

Abstract:

This topic illustrates how we have stuck to LFG in principle, but have had to deviate in practice to avoid ambiguity and handle illformed input.

ALICE-chan is a Computer-Assisted Language Instruction program for beginning to intermediate Japanese using NLP to help teachers author exercises and to process student input. Grammatical function assignment and semantic role assignment in ALICE-chan are consistent with LFG's lexical mapping theory and theory of non-configurational encoding. However, various practical considerations have led us to develop new procedures for grammatical function and semantic role assignment. These considerations include the ambiguity of Japanese case marking particles, the need to process illformed input, and the need to provide comprehensible feedback to human users.

Grammatical function and semantic role assignment are particularly challenging problems for ALICE-chan. This is because, in Japanese, the grammatical function of a phrase, and hence its semantic role, is largely determined by the particle (*ga, o, ni, e, de, kara,* etc.) that it is marked with, but particles are likely to be wrong in student input. Furthermore, each particle can mark several grammatical functions and many functions can be marked by multiple particles (e.g., *o, wa,* and *mo* for OBJ), resulting in a large amount of local ambiguity during parsing.

ALICE-chan handles these problems by postponing grammatical function/semantic role assignment. As each phrase is encountered in student input, a list of possible functions/roles is compiled based on (1) the particle (2) coarse selectional restrictions and (3) the function/role borne by the head of the phrase in a model answer pre-authored by a teacher. Final assignments are made after collecting the subcategorization information of the head of the sentence and the possible functions/roles of each phrase. Error messages are generated if the particles are not consistent with the function/role assignment. (The distinction between grammatical functions and semantic roles is somewhat blurred because of the need to use terminology that non-linguists can understand.)

4.11 Lexical Semantics in Mikrokosmos(Sergei Nirenburg and Lori Levin)

Abstract:

In addition to rule interaction issues, another research focus of the Mikrokosmos project is the integration of two approaches to lexical semantics that are usually considered to be incompatible. The approach to lexical semantics taken by researchers in syntactic theory is useful for identifying predicate-argument relations in syntactic structure, but is not suitable as the basis of an interlingua for machine translation. We explain the reason why in the talk. On the other hand, an approach to lexical semantics driven by ontological issues provides a good basis for an interlingua, but is non-optimal as a basis for identifing predicates and their arguments in particular languages. In this talk, we describe how the two approaches to lexical semantics are integrated in the Mikrokosmos lexicon.

Notes:

Most of the discussion merged issues from the two topics and revolved around the problems of MT lexicon design. In particular, for the question of how to treat translational divergences in MT, the solutions proposed involve both stating the differences lexically, and linking.

Another important question was: “How do you determine the ontology needed for semantic analyses?”

Problem of “frame” vs. “features”: representation for scope and embedding (e.g. *perhaps she must go.*)

4.12 John Griffith (U Tübingen)

Abstract:

Presentation of the DUSt (Disjunctive Unification System) feature structure unification algorithm. DUSt allows representations with unrestricted (named) disjunctions and tries to process them as efficiently as possible. Ambiguity often is encoded in NLP systems as disjunction and all formalisms allow disjunction in some way. However, the addition of disjunction to an otherwise efficiently processable formalism will cause processing to have exponential complexity. The disjunctions that are allowed in DUSt are also very rich in expressive power. They are "named" disjunctions, which means that a choice in one disjunction can affect the choices in another disjunction. Named disjunctions therefore allow linguistic generalizations to be made that are not expressible with only non-named disjunctions and they allow compact representations that are computationally attractive.

The presentation gives an introduction to named disjunctions and the DUSt representation in general, highlighting the kinds of generalizations that can be expressed in it. Concerning the algorithm for unification of feature structures with named disjunctions, the talk particularly addresses concerns for efficiency.

Notes:

The presentation discussed the use of disjunction for expressing linguistic generalizations, the ambiguities that it gives rise to, and the question of computational efficiency for treating normal and named disjunction.

Question: are named disjunctions desirable ?

Examples:

1. Inflectional morphology

```
[ MORPH [ STEM  2
          ENDING 3{e, st, t, en, et, en}
                                     d1
          FORM 2&3 ]
  SYN|LOC|HEAD|AGR  { [PERS 1st  ....} ....}  ]
                    d1  NUM sing]
```

This is of limited use: a weak form actually requires only atomic values.

2. Head Movement.

```
[ ....|VFORM  1{fin bse}
                   d1
  ....|VALENCE {[ SUBJ <2> [SUBCAT <2>]]
                            SUBCAT <3>]
        2:{ ([CASE nom] | <3>) ([CASE not(nom)] | BOTTOM)}
           d1
```

Here, there is no way to push the disjunction down to atomic values.

Note: sharing between disjuncts is not actually done, it is simply a notational device

Pereira: It all boils down to implications between equations and disequations (on paths)

3. for encoding Word Order:

```
    [SUBCAT    [FIRST { FS1   FS1   FS3}
                       d1
                REST   [FIRST { FS2   FS3   FS2}
                               d1
                        REST [FIRST { FS3   FS2   FS3}
                                      d1
                              REST   NIL]]]]
```

Equivalence with normal disjunction plus definite clause attachment:

```
[ F   { a   b}
       d1
  G   { c  d}]
       d1
```

is equivalent to:

```
[F   x
 G   y] , (x=a , y=c) ; (x=b , y=d)
```

Problem: it is possible to write (partially) meaningless specifications:

```
[f {    [g   { a   b}]   c}]
    d1       d1
[F x] , (x=[G y] , y=a) ; (x=c , y=b)
```

Named disjunctions that are ill-formed are easier to detect at run time than their definite clause equivalents with attachment.

Pereira: This is computationally intractable. It amounts to Boolean satisfiability (last example). We probably need specialized Boolean constraint solvers. The criticism that the definite clause format is too much Prolog-like and bound to the same limitations w.r.t. processing (sequential, first-come-first-evaluated mode) is not tenable. Simply change strategy.

Netter: He has used named disjunctions to separate the specification of a given item (e.g. determiners) from its actual morphological realization. Furthermore, it can be used to maintain ambiguities without solving them. There is one system in Saarbrücken that has named disjunctions.

General conclusions and advice:
- upon unification, choose first the path with the smaller amount of disjunction.
- avoid inheriting distributed disjunction.
- need global names for disjunctions.

5 General Discussions

In addition to the presentations and the individual discussions of specific issues after each of them, there were two extensive general discussions based on the set of questions that had been asked at the beginning and on additional topics. In particular, Jochen Dörre and Herbert Ruessink had proposed the following themes for discussion:

- weights and induction of grammars;
- ill-formed input;
- expressivity and efficiency issues;
- the relations between linguistic frameworks (comprising formalisms), linguistic theories and linguistic analyses of languages (i.e. grammars) on the one hand and computational linguistic formalisms and grammars on the other. This serves to get an idea of what we are to understand by "linguistic" in "linguistic adequacy" (these matters are also covered quite well in Shieber's presentation at the Linguistic Theory and Computer Applications Workshop, Whitelock et al. 1987 (**?**)). The requirements made of a formalism from a linguistic point of view are not compatible with the requirements made of a formalism from the viewpoint of current computational linguistic research and development. Linguistic adequacy does not mean catering to the needs of a linguistic framework, but catering to the needs of grammar development. This means being more on the line of adequacy with respect to linguistic data rather than linguistic theory.
- identifying the needs (with respect to a formalism) of grammatical development and separating the computational needs from the more linguistic needs (modularity, generalizations, general principles, etc.). Is linguistic adequacy the same as expressivity?
- How can the linguistic needs be realized? In which cases is it possible to measure the extent to which they are realized? Is it feasible (or necessary) to develop a single formalism that caters for different linguistic frameworks?
- To what extent are computational and linguistic needs diametrically opposed? What kinds of solutions could we think of to overcome this?

Hans Uszkoreit, who could not attend the workshop, had also submitted the following set of questions:

1. Which contemporary formalisms are actually used for large scale grammar development? What are the experiences?
2. Is large scale grammar engineering (GE) feasible? If not: will it be?
3. Is GE the appropriate way to get usable NL systems?
4. Should we rather concentrate on statistical methods?
5. Can only a combination of statistical methods and GE help?
6. Is GE necessary for theoretical linguistics?
7. Is the combination of theoretical linguistics and applied CL necessary for success in language technology?
8. Should formalisms implement linguistic theories or should they be more flexible since theories change all the time?

9. Where is the bottleneck in grammar engineering? What is missing?

   - better linguistic theories

   - appropriate linguistic analyses

   - analyzed linguistic data collections

   - more adequate formalisms? (totally different design? additional descriptive constructs?)

   - better implementations of existing formalisms? (better performance? better tools, i.e., editors, browsers, tracers, debuggers, data bases, diagnostics tools, consistency maintenance tools?)

   - better methods for grammar engineering? (modularization, organization of knowledge? consistency maintenance? testing, diagnosis by corpora? testing, diagnosis by annotated test suites? methods for distributed GE?)

   - different people?

10. Is the time ripe for large scale multi-lingual GE?

All these topics and specific questions were either taken up during the individual discussions or were addressed during the general discussions.

5.1 General Discussion: Transcripts of notes

(Saturday, 28/08/94)

K.N. - Given two weakly equivalent grammars, which one is more important?

F.P. - If the grammar writer is a programmer, efficiency and predictibility are most important. For a linguist, expresssivity is more important.

Descriptivity and adequacy do not have to be in contradiction with efficiency.

Phrase-Structure backbone as control structure.

D.E. - Does the grammar writer have a background in computer science or in linguistics? We should take that into account.

S.B. - Shift from the logical problem of language acquisition to the question of complexity of processing. Then, efficiency must be taken into account in the design of grammars. Efficiency is not to be determined solely by the performance of an implementation.

A.A. - People are not that efficient anyway: misunderstandings don't get full parses. People are efficient at living, not processing.

A.S. - Linguists use the tools produced by computer scientists. So there has to be some interaction.

Fe.P. - Successful systems have involved close collaboration. How can the design be decided upon in advance?

L.L. - agreed. You can't expect anticipating the issues that will come up before working on a large system.

K.N. - In our experience, we didn't need to change the formalism during implementation, only low-level aspects.

D.M./K.N. - High-level programming languages: are they of any use other than for linguists to express their ideas, just like they would on paper?

Fa.P. - Whether the same formalism is used by linguists and grammar writers is irrelevant. The important point is to provide a system that is useful for NLP.

What is exactly the process of grammar writing? Until we know more about it, it is difficult to make a priori judgements about the adequacy of a formalism for that process rather than for processing itself.

Fe.P. - Look at what other sciences are doing: mathematics, physics use simulators to help them figure out consequences of their hypotheses.

K.N. - Grammar learning will become useful and thinkable only when our processing tools become nore efficient.

Fe.P. - Multiple inheritance is needed, it should be added to ALEP.

M.K. - It can be done with the bit-vector encoding of Aït-Kaci.[2]

(Sunday, 29/08/94) (Question numbers refer to H.U.'s questionnaire).

M.K. - About adequacy of form vs. coverage: why is large coverage not enough?

D.E. - For instance, SYSTRAN has a large coverage but no adequate formalism (old Fortran code).

A.A. - Descriptive adequacy is the criterion.

K.N. - That would mean that SYSTRAN is linguistically adequate.

F.P. - It is adequate for its task: string rewriting.

K.N. - For application independence, reusability has to be the criterion, since any formalism can be tailored towards a specific application.

Question 1: What does "contemporary" mean, if grammar development takes at least 5 years?

A.A. - TAGs, String Grammar Project

.... - GDE, Xerox, GPSG, LFG, ...

M.K. - Why is size important for adequacy?

F.P. - Future relevance cannot be determined at present.

Question 2: Different question, about large grammars of the past: What lessons are there to learn from them?

Question 3:

F.P. - Sure, grammar engineering must exist. It must be done, and has been done. The question is: "Was it fun?"

A.S. - Automatic derivation of lexicons with many ambiguities is one of the major problems of grammar development and engineering. More constraints in the lexicon and problem of sense disambiguation.

[2] Aït-Kaci et al. (1989). (**?**)

U.K. - Disjunctions?

F.P. - Constant factor of disjunction/ambiguity weighting is relevant.

D.E. - Does that mean that the answer to Q.5 is “no”?

Question 5:

K.N. - Grammar engineering is not just writing nice grammars in a neat formalism, but also concerns other aspects of processing.

F.P. - From the CLE experience,[3] the more you put in the lexicon the better it is. Also, there is nothing as efficient as context-free parsing.

S.M. - Maybe this is just because we know more about context-free grammars.

F.P. - Large scale grammar engineering is feasible; whether the tools we envisage are appropriate is an open question.

L.L. - We must distinguish knowledge acquisition, preferences for processing and subsymbolic NLP.

K.N. - The distribution of small vocabulary in speech is fairly even, whereas the samples to be found in syntax/semantics are too small to compute relevant distributions.

L.L. - We’ll never get large scale grammars without automatic knowledge acquisition, because of the application dependency of grammars.

Discussion about “low level approach” vs. “high level approach”, and “dumb/intelligent” approaches and interfaces.

A.A. - Maybe not all NL applications are linguistic applications.

K.N. - Right: text processing for instance.

M.R. - Spelling correction is at the borderline. It’s a combination of computational linguistics and computer science.

Question 8:

M.R. - Is grammar engineering useful for computational linguistics?

Fe.P. - The homogenity of frameworks depends on the number of linguists working in the framework, not on the preciseness of the formalization.

Fa.P. - Grammar engineering is not necessary for theoretical linguists.

M.R. - There is a huge gap between the theoreticians and the people in marketing. MT is galaxies away from what marketing people are promising in their brochures.

F.P. - Any theoretician in physics, etc., needs computers. It has to be the same in language, which is too complex to think about without tools..

M.R. - What are the tools used by both theoretical computational linguistics and grammar engineering?

K.N. - The analogy is the computational linguist using computers and grammar engineering.

[3] Alshawi (1992). (**?**).

Question 10:

K.N. - Formalisms have already moved far enough. Still, some things are missing (e.g. sets for implementing HPSG).

F.P. - Rather than e.g. trying to implement HPSG, an alternative could be high level programming languages which allow sufficient abstraction, e.g. LIFE (not necessarily the right one).

Differences: modularization, data abstraction, predictability, and control mechanisms.

A.S. - Too general, we need linguistic (language) environments. Is FO logic sufficient?

S.M. - No, we need more general, theory-neutral languages to figure out the necessary primitives.

S.B. - Theory neutrality is impossible.

Question 11:

L.L. - A big problem is the lack of understanding of the data. Robustness and grasp of the data are inseparable. For instance, there is currently no lexicon with adequate subcategorication information for treating unrestricted text.

A.S. - We need richer lexical knowledge bases!

L.L. - It is not clear what there is out there in terms of corpora.

D.E. - We don't need better implementations of existing formalisms, but a better understanding of data.

A.A. - We need both, current implementations are unsatisfactory. We also need core lexicons and core grammars.

F.P. - We need more work on test and diagnostics, annotated corpora and test suites, and better lexical resources, rather than implementations going on and on.

S.M. - We need better interaction with Logic Programming and Knowledge Representation people.

F.P. - There can be no serious progress if a well-defined boundary is drawn between linguists and computer scientists.

Funding for developing formalisms must go together with implementing a large scale grammar and testing it.

5.2 Results of the Questionaire

The questionnaire written by Hans Uszkoreit was distributed to the participants, and we asked each of them to fill in and to return it to us, so that the results could be compiled and included in a report to be distributed to anyone who has either attended the workshop or expressed an interest in it. Unfortunately, very few people have actually done so, and we can only give a summary of the results of the discussions themeselves. This is what we do now, taking each point in turn.

1. A variety of formalisms are in use today for GE, among them:
   GDE: GPSG; XTAGs: TAG; String Grammar Project; Xerox: LFG; HPSG.

2. Large scale GE is clearly possible. Large grammars have , however, been writen in non-contemporary formalisms. The question is whether the appropriate means and tools have been available for efficient and adequate development. One of the major problems of large scale GE is automatic acquisition of lexicons, in particular the treatment of ambiguities.

   The question of the adequacy of a formalism cannot be simply correlated with the size of grammars that have been written in that formalism. Large grammars can be written with a particular application in mind in a formalism which may be quite inadequate for other applications. If adequacy of description was the criterion, a system such as SYSTRAN could be considered as linguistically adequete. However, one of the criterion for determining the adequacy of a formalism could be the issue of re-usability and application independence.

3. GE not only involves writing large grammars in a nice formalism, but efficient processing is equally necessary. The reason why context free grammars are the ones most efficiently processed at present is presumably due to the fact that we know most about such grammars. People are mostly trained to write CF grammars ("There's nothing as fast as context-free parsing." Fernando Pereira) and as their experience conditions the way they write grammar rules, grammars which have at least a context-free backbone will be written in a more efficient way.

4. We have to distinguish between genuine sub-symbolic NLP from the use of statistical methods for special tasks, such as knowledge acquisition, for determining preferences or weighting. For these tasks, statistical methods are also clearly necessary in a traditional scenario of grammar development.

   Large scale grammars will necessarily require automatic knowledge acquisition. The type of grammar required is application dependent.

5. GE *must incorporate* statistical methods, distributional data and weights.

   We must also remember that not all NL applications are linguistic applications, and a successful NL application may also have to incorporate other databases.

6. No, but it may help (cf. the experience of GPSG and HPSG). The homogeneity of a framework depends more on the number of linguists working in the framework rather than on the preciseness of the formalization. GE is not necessary for theoretical linguistics, in fact there is still a wide gap between developments in theoretical linguistics and actual applications. However, any theoretician, whether working in physics, mathematics etc., needs computers. In exactly the same way, theoretical linguists need computational tools and methods, because language is too complex to be described without computational methods.

7. Of course, we need a combination of theoretical linguistics and applied CL, but descriptive linguistics might actually be even more useful.

8. Nearly all current formalisms have already moved far enough. What is required now are more abstract high level programming languages which allow sufficient abstraction rather than the implementation of specific linguistic theories.

9. Better implementations of formalisms have to be complemented by better grasp of the data. More effort should be spent on annotated corpora, test and diagnosis, and on lexical resources.

   The important issues are modularization, data abstraction, predictability and control mechanisms. What is needed are more general and (as far as possible) theory neutral languages, which provide the necessary primitives for description.

   There should not be a strict separation between computer scientists and theoretical linguists, but there must be better interaction with Logic Programming.

   Funding for developing formalisms must go together with implementing large scale grammars.

10. It should be...

6 Conclusion

Even though the questionnaire was not as successful a tool for getting information as we had hoped, the workshop itself provided valuable input to the work of the subgroup. Especially helpful was the focussed discussion of the precompiled list of questions. We knew in advance that we could not hope to get complete answers to all these questions, but the result of those discussions already provides a very good idea of the directions in which researchers in the field are looking for answers.

The meeting was also important for increasing the visibility of the EAGLES work among the users of linguistic formalisms. Their requirements are essential for the recommendations of the Working Group, and some concrete suggestions as to the best way for research to proceed have now emerged.

References

[1] Aït-Kaci, H., R. Boyer, P. Lincoln & R. Nasr (1989). “Efficient Implementation of Lattice Operations” *ACM Transactions on Programming Languages and Systems*. Vol.11. No.1.

[2] Alshawi, H. ed. (1992). *The Core Language Engine.* ACL-MIT Press Series in Natural Language Processing.

[3] Backofen, Rolf, Hans-Ulrich Krieger, Stephen P. Spackman, Hans Uszkoreit eds. (1993). *Report of the EAGLES Workshop on Implemented Formalisms at DFKI, Saarbrücken, March 1-3, 1993.* DFKI, Saarbrücken.

[4] Carpenter, B. (1992b). *ALE: The Attribute Logic Engine User's Guide,* Laboratory for Computational Lingustics, Carnegie Mellon University.

[5] Dowty, D. (1991). “Thematic Proto Roles and Argument Selection”. *Language* 67, pp.547–619.

[6] Parsons, T. (1990). *Events in the Semantics of English: a Study in Subatomic Semantics.* Cambridge: MIT Press

[7] Pollard C. (1990). “On Head Non Movement.” *Proceedings of the Symposium on Discontinuous Constituency.* Institute for Language Technology and Artificial Intelligence, Tilburg, The Netherlands, 25-27 Jan 1990.

[8] Pollard, C. & I.A. Sag (1994). *Head-Driven Phrase Structure Grammar.* CSLI Series. Chicago: University of Chicago Press.

[9] Whitelock, P., M.M. Wood, H.L. Somers, R. Johnson & P. Bennett, eds. (1987). *Linguistic Theory and Computer Applications.* London: Academic Press Limited.