1 Technology and science

In the growing field of language technology, a reemerging issue is the proper place of linguistics—whether language technology should be based on linguistic insights, models and theories or not. An equally interesting question arises when we turn the question around and ask how language technology may and should influence linguistic theory and practice.

Let us first take one step to the side and consider the relationship between science and technology in other fields. Our first thought is that science is primary, while technology combines scientific insights from several fields and applies them to specific cases. The construction of suspension bridges, to take one example, applies Newtonian mechanics to calculate the optimal arcs of the deck and the bearing wires and the height of the towers and the suspender cables. At the same time the mass and strength of the materials used must be taken into consideration to make sure the bridge can hold itself and its load. In addition, it is mandatory to test a model of the bridge in a wind tunnel to see how well it will perform under varying conditions.

But the interaction between science and technology is not unidirectional. First, technological developments result in better tools which open opportunities for new scientific insights, as witnessed from Galileo’s telescope to the PET scanner. Second, in many cases the scientific results are not available beforehand. The quest for a technological solution to a certain task may initiate research leading to new scientific results. Today, we can witness this in the biotechnological fields. Third, methods from technology, in particular simulations, enter the sciences. Returning to the suspension bridge, there is no easy way to calculate the behavior of the bridge under different wind conditions. But it is possible to see what happens by simulations in a wind tunnel. Lately, computer simulations have supplemented physical simulations, and they are becoming accepted as scientific discovery procedures in several fields, including population dynamics, climate studies, quantum chemistry and nuclear physics.
In the sequel, we start by considering some experiences from one particular effort, the LOGON project, and relate them to the field of language technology as a whole. We will then discuss possible consequences for linguistics as a scientific enterprise.

2 The LOGON project

2.1 Machine translation

The focus of the LOGON project is machine translation from Norwegian to English of texts from the hiking domain (Oepen, Dyvik, Lønning, Velldal et al. 2004, LOGON web page). The project, which is supported by the Norwegian Research Council’s KUNSTI program for the advancement of language technology in Norway, lasts for roughly 4 years, involves three universities, Bergen, Trondheim and Oslo, and employs on average three researchers on the post doctoral level and 4 doctoral students in addition to 5 (part time) principal investigators.

Machine translation (MT) started as a research field already in 1949 when Warren Weaver (2003) circulated a memo proposing to use the emerging computer technology for automatic translation. A year later Yehoshua Bar-Hillel was employed as the first machine translation researcher at MIT. The field grew rapidly during the 1950s with active research groups in several countries and various international conferences. The earliest approaches to MT were later classified as the direct approach. Somewhat simplified, the basic idea is to use a dictionary to find the proper translation of each word. In addition, the source language input is partly analyzed morphologically to identify features like number and case, and the target language output is rearranged to get the word order correct.

It was soon observed that this approach was insufficient and that a deeper understanding of the linguistic structures of the two languages was required. At about the same time, Noam Chomsky (1957) introduced and argued for generative grammars in linguistics; finite mechanisms sufficient for generating an unlimited number of different sentences. Generative grammars, in particular the non-transformational variants, came to play a large role in the development of MT and later in computational linguistics in general. Chomsky’s concern was not machine translation, but linguistics and psychology. Apparently, this is an example of a technology (MT) applying an available theory (generative grammar), but it is tempting to speculate whether it was more than a coincidence that generative grammar was first conceived at MIT, where there was already active MT research.
In the next approach to MT, the source language sentence is translated into a semantic representation language, a lingua franca or *interlingua*, and a target language sentence is generated from this representation. Not only a morphological, but also syntactic and semantic analyses of the source sentence are applied in the construction of the interlingua representation. The problem for this approach is to construct a universal interlingua in which the semantic content of any human language can be represented. Different languages use different means to express meaning and to classify the world.

The *transfer* approach tries to take this into consideration. In *semantic transfer* the source language sentence is analyzed syntactically and semantically, resulting in a monolingual semantic representation. This representation is transferred to a similar representation for the target language from which the target sentence is generated. The same semantic representation of the source language sentence may be used for translation into several different languages, while the transfer step is fine-tuned to a particular language pair. In contrast, in a *syntactic transfer* approach the transfer step is done between syntactic representations instead.

### 2.2 Semantically based transfer

The semantic representations in LOGON are expressed in what is called Minimal Recursion Semantics (MRS) (Copestake, Flickinger, Pollard & Sag to appear). The formula (1b) is an MRS representation of sentence (1a).

\[(1) \text{a. Hver fjellfører skimtet ei hytte.} \]

\[b. \langle h0, \{h1:hver(x,h2,h3), h4:fjellfører(x), h0:skimte(e5,x,y), h9:en(y,h6,h7), h8:hytte(y)\}, \{h2 =_q h4, h6 =_q h8\}\rangle\]

The basic units are the elementary predications (EP), e.g., *skimte* \((e5,x,y)\). This represents a formula in predicate logic, with \(x\) and \(y\) as variables. The argument \(e5\) is a Davidsonian event variable which may be modified by an adverb or a prepositional phrase. While it is a goal in logical representations of semantics to get the scope relations right, MRS allows for scope underspecification. Thus the MRS (1b) can be further specified to the MRS (2a) which represents the logical formula (2b), as well as to an MRS corresponding to formula (2c). The last part of the representation is a set of restrictions on scope, here \(h2 =_q h4\) and \(h6 =_q h8\). The first one expresses that \(h4\) must equal \(h2\) except that some quantifiers may intervene. Together with some general principles, the equations assure that (1b) cannot represent other formulas than these two.
One possible translation of this sentence is shown in (3a) with the associated MRS in (3b).

(3) a. Every mountain guide saw a cabin dimly.

b. \( \langle h_0, \{ h_1: \text{every}(x, h_2, h_3), h_4: \text{compound}(e_9, x, z), h_{10}: \text{udefq}(z, h_{11}, h_{12}), h_{13}: \text{mountain}(z), h_4: \text{guide}(x), h_0: \text{see}(e_5, x, y), h_9: a(y, h_6, h_7), h_8: \text{cabin}(y), h_0: \text{dim}(e_{12}, e_5) \}, \{ h_2 = q h_4, h_6 = q h_8, h_{11} = q h_{13} \} \rangle \)

We observe that there are many one-to-one correspondences between words in sentence (3a) and sentence (1a), and recognize them as correspondences between EPs in (3b) and (1b). There are also some mismatches. The morphological compound *fjellfører* is expanded to *mountain guide*, and the simple verb *skimte* is translated to the complex *see... dimly*.

Such mismatches are harder to handle in an interlingua system. Since the interlingua is in principle intended to work with many different languages, it has to anticipate the possible variations and contain a rich enough inventory of expressions. Moreover, the mapping between a language and the interlingua must be constructed independently of the other language. For example, whether *skimte* should be decomposed in the interlingua has to be decided independently of its translation in English. In the transfer system, on the other hand, the transfer module is particularly constructed to handle the mismatches between two specific languages.

In addition, many properties which are unique to one of the two languages, like idiosyncratically chosen prepositions, do not show up in the MRS representation at all. They are taken care of by the monolingual parsing and generation modules between the surface realization and the MRS. This is an advantage over a purely syntactically based transfer system.

Minimal recursion semantics was originally developed in the context of Head-Driven Phrase Structure Grammars (HPSG), a strongly lexicalized unification-based grammar formalism (Pollard & Sag 1994). In the modern form of HPSG, all feature structures are typed and respect type constraints, and unification is between typed feature structures (cf. e.g. Sag, Wasow & Bender 2003). For the English generation, LOGON has adopted and developed further the large, publicly available, HPSG-based English resource grammar (ERG) implemented in the LKB system (Flickinger 2002, ERG web page).
For analysis, LOGON has made use of and extended an existing grammar for Norwegian, NorGram, developed within the framework of Lexical-Functional Grammar (LFG) and implemented in the XLE system from PARC (Dyvik 2000). The slightly unconventional choice of using both LFG and HPSG was partly practically motivated; we wanted to reuse ERG and NorGram, and partly theoretically motivated; we wanted to use semantic transfer and abstract away from the particular grammar formalism. A first challenge was to combine MRS semantics and Lexical-Functional Grammar. This was done successfully by the use of LFG’s projection mechanism (Oepen et al. 2004).

The transfer formalism has been developed and implemented specifically for the project. The transfer is done by rewriting rules. They are written in a type-based formalism and implemented in a system on top of the LKB. Finally, the three different components for analysis, transfer and generation are built together into one functional system. Technically, this is a fairly large and complex system gluing together several different processes running simultaneously in different programming languages.

2.3 Stochastic ranking

During the 1990s, two new approaches to machine translation appeared; statistical machine translation (SMT) and example based machine translation (EBMT). Instead of founding MT on linguistic models and hand-crafted rules, the two approaches start with large collections of translated texts and try to extract, or learn, regularities. When new text is to be translated, they exploit the similarities with previously encountered translations. More specifically, an SMT system tries to learn two types of probabilities during training: how likely a word \( w' \) is as the translation of a word \( w \) by counting how frequently \( w \) gets translated as \( w' \), and how likely a sequence of words is as a sentence in the target language. Later on, when given a sentence in the source language, the SMT system will, in principle, consider all possible strings of words in the target language and calculate the joint probability of it being a sentence and its words being translations of the words in the source sentence.

In many ways, SMT can be considered a return to the earliest, direct, word-based approach to MT. At the same time, by adding the stochastic component, it also introduces a new dimension compared to all earlier approaches. It is essential for an SMT system that it can consider huge amounts of translated text during training, and the computations involved are so resource-consuming that they would not have been possible twenty years ago. Pure SMT is an extreme representative for language technology without linguistic theory (Jelinek 2005). The initial results of SMT were quite promising, and by some evaluation mea-
sures, SMT seems to outperform traditional rule-based systems. At the same
time SMT also has clear limitations. A fair amount of the output produced is
ungrammatical or it totally misrepresents the content of the source sentence—

ceiling effects which cannot be overcome by pure word-based SMT.

In the LOGON project we try to combine insights from SMT with the rule-

based transfer system. For one Norwegian sentence, the transfer system may
produce 10, 20 or even several hundred different translations in English. A

string may be ambiguous and get several different analyses. During transfer,
there might be many different alternative translations for many words. And fi-
nally, one MRS may have many different realizations in English. So far we
have experimented with ranking the output on the basis of English text (Velldal
& Oepen 2005). The first results of these rankings are promising, and clearly
better than a random selection. During the remaining project period we plan to

apply ranking of the Norwegian analyses on the basis of a tree bank for Nor-
wegian to get the most likely grammatical structure as input to the translation
process. Finally, we should calculate the probability of the output sentence as a

translation of the input sentence. But to get this step right, we will need large
amounts of translated text as training material, and we will probably not reach
that stage within the remaining project period.

2.4 Profiling and regression testing

Constructing large-coverage grammars, like NorGram and ERG, is a stepwise

and gradual process. The language engineer continually extends the grammar to
increase the coverage. But this may lead to overgeneration and false analyses.
After a while the grammar becomes so big and complex that it is impossible for
her to oversee all the consequences of her changes. She has to test the grammar
empirically. To do this efficiently, in LOGON we apply a method of system-
atic regression testing (Oepen & Flickinger 1998) together with an advanced
programming package, [incr tsdb()] (available from Delphin web page). A test
suite is a set of sentences with optional annotations. After a round of changes
to the grammar, all the sentences in the test suite are parsed in a batch process
and the results recorded and stored in a database. By comparing the results of a
batch run to earlier results in the database, the language engineer gets immedi-
ate feedback to the changes to the grammar. She immediately sees which new
analyses have appeared and which old analyses have disappeared.

In the LOGON project, we use this method in the development of each of
the different grammars and the transfer module. But we have also taken the
method one step further (Oepen, Dyvik, Flickinger, Lønning, Meurer & Rosén
2005). Not only may the changes to a grammar influence the performance of
that component, it may also influence other components. We have adapted the
method of profiling and the [incr tsdb()] system to the translation process as
a whole. We repeatedly batch translate test suites and compare the outcome
with earlier rounds. This is particularly important since the development of
the system involves about ten researchers at three different sites working on
different modules, including grammars and software for parsing, transfer and
generation.

The test material used in LOGON consists of some hand-constructed test
suites of between 100 and 300 sentences, one for simple grammatical construc-
tions and one for closed class vocabulary. In addition, we have selected a corpus
of about 3000 authentic Norwegian sentences from the hiking domain. We have
two or three different English reference translations of these sentences. This ref-
erence corpus is a useful guide for the writers of the transfer rules, even though
it is not a goal to reproduce the example translations in all cases. In addition, it
is used for continuous regression testing. Parts of the corpus are kept aside for
evaluation purposes.

3 Philosophy of linguistics

3.1 Holism

At first look, the LOGON project fits well with the idealistic view of the rela-
tionship between science and technology. Where the bridge engineer applies
theories from physics and provides specific lengths and other parameters, the
language engineer applies theoretical syntax and semantics and provides a lexi-
con of words and their properties. But the contribution from the language engi-
neer amounts to more than this. From theoretical syntax, she gets some general
principles, and she gets detailed analyses of some specific phenomena. Each
detailed analysis typically considers a limited set of data, data intended to il-
lustrate exactly these phenomena. But when the language engineer tries to glue
together these different detailed analyses, she often experiences that they do not
fit together. To get a working system, she has to make adjustments and cor-
rections and also invent analyses for several phenomena not considered in the
theoretical literature. In addition, the language engineer departs from the theo-
retical linguist with respect to the data on which she tests her grammar. Besides
carefully chosen, tricky (but typically short) test sentences, the language engi-
neer considers sentences from a corpus. And often a grammar which seems to
work on the shorter sentences gets problems on longer sentences where several
phenomena interact.

It is easier for the theoretical grammarian who considers a more limited set
of data to “verify” her proposal. The language engineer serves the job of an
experimental scientist testing the theoretical proposals under harder conditions and more realistic settings than what the theorists do. This confirms Quine’s (1961) thesis that one cannot test theoretical statements in isolation. What has to be tested are whole theories and systems. Rather than being a consumer of theoretical grammar, the language engineer adds to the field. In this respect, the use of computational implementations serves as a tool for advancing generative grammar. The computer serves a similar function as Galileo’s microscope, and could be named a “macroscope” for theoretical linguistics.

By taking this holistic approach, it not only becomes harder to “verify” a grammar than what is sometimes assumed in the theoretical literature, it also becomes harder to falsify individual grammar rules. Most often there is more than one way to fix a shortcoming in a grammar. For example, whether a particular PP should be considered an argument or adjoined to the verb, whether the word *down* in (4a) should be described as a particle, an adverb or a preposition with no complement, and what the role of *down* is in (4b), can only be determined by reference to the grammar as a whole.

\[
\begin{align*}
(4) & \ a. \ The \ Titanic \ went \ down. \\
& \ b. \ Sam \ went \ down \ to \ the \ supermarket.
\end{align*}
\]

### 3.2 Syntactic theorization

Quine (1961, p. 46) in his description of holism envisaged “our so-called knowledge or beliefs” as “a man-made fabric which impinges on experience only along the edges.” Statements closer to the periphery are most likely to be revised to accommodate observations. Statements further from the periphery are less prone to revision, but might get revised because of their logical interconnections to other statements. In a similar vain, Lakatos in his description of research programs talked about a protective belt of less firm statements that were apt to revision to accommodate observations and at the same time protect the core of the program from changes. Theories of generative grammar can be considered such research programs. Within these theories we can recognize several layers: from the most basic assumptions which cannot be changed without rejecting the theory, through intermediate principles which may be changed, but only rarely and after long discussions, to the surface layer of particular grammar rules and lexical entries which may be changed on a daily basis.

There are several different theories of generative grammar alive today, and two of them, LFG and HPSG, are applied in the LOGON project. The two share some basic assumptions. Both are aiming for broad coverage and computational tractability. They are lexicalized, and they use feature structures and
unification as basic ingredients. They depart, however, on the intermediate principles. In LFG, the principles include the strict separation between the different projections, the relative role of the constituent-structure and the functional-structure, and the way arguments are selected by the completeness and coherence principles. In HPSG, the intermediate principles include the sign as an integrated unit of phonological, syntactic and semantic features, taking care of both configurational and functional information, together with the principles for combining signs, like the head feature principle. In addition, HPSG uses types for expressing generalizations. Today, the two theories live more and more separate lives. In particular, they both have their own annual conferences.

As far as we know, LOGON is the first project to apply both theories in the same large computational system. And even though we apply them to two different languages, this gives us a unique opportunity for comparing them. So far, we see no reason for rejecting either of them on empirical grounds. It is also interesting that in spite of the different principles on the intermediate level, which make it impossible to translate grammar rules or lexical entries directly between the two frameworks, there are clear convergences on the surface level. For example, the bracketing of a string or whether a particular PP is an argument or an adjunct can be discussed across the two frameworks.

This convergence is on the one hand encouraging, but it also raises some concerns. What is the substance of the two theories? What is the theoretical status of the intermediate principles? Are there any ways to falsify them? To defend the grammatical theories, one could argue that the situation is not that different from other scientific fields, where basic and intermediate principles can be protected by adjusting the surface statements. But the situation in generative grammar seems more exceptional, as there are several different theories that neither communicate nor compete. In a way, the relationship between LFG and HPSG can be compared to the relationship between English and French, two different languages for talking about the same phenomena.

3.3 Semantics and translation

We will turn to the role formal semantic theories play in our technological MT project. Formal semantics tries to get the truth conditions for declarative sentences right, and to calculate them systematically. Though not a necessity, the truth conditions are often calculated by translating sentences into a logically perspicuous notation where formulas “wear their truth conditions on their sleeves”. An ambiguous sentence has to be translated into several different formulas which each represent a reading. Conversely, if two different sentences have the same meaning, they should be translated into formulas with the same
truth conditions, and it should be possible to deduce one from the other. More generally, the logical formalism should support inferences and make logical relationships between different sentences visible.

In the LOGON project we try to apply formal semantic representations in MT. But on our way from theory to technology, we also make some adaptations. We have chosen to use underspecified semantic representations. Semantic ambiguities which are not reflected in syntax are not spelled out when translating into MRS. They are kept underspecified in the semantic representations to translate more efficiently. The ambiguities may in many cases be preserved when translating into a closely related language. In cases where the ambiguities cannot be retained, we spell them out during transfer, rather than during analysis. As a result, the MRSs are constructed systematically from the syntax, but not compositionally in the traditional sense.

Like in syntax, the language engineer encounters phenomena which are not fully covered by any theoretical analysis. She has to fill in the details herself. She might also have to make adjustments to get different bits and pieces to work together. There is generally more than one way to solve a problem, and an analysis of a class of phenomena can only be evaluated in the context of the whole system. To take one example, the questions regarding how down should best be handled from example (4) is as much a question for semantics as it is for syntax.

In addition, we see one more degree of freedom. The semantic representations may be chosen close to the truth-conditional interpretation and further from the surface form of the natural language, or closer to the surface form and further from the interpretation. To take a simplified example, if one thinks the interpretation of the simple past tense is that the event took place sometime before the utterance event, any of the following might be ways to represent this in logic, where (5a) is quite close to language and (5c) close to the interpretation.

\[
\begin{align*}
(5) \ a. \ & \text{run} (PAST,kim) \\
& \text{b. } P(\text{run}(kim)) \\
& \text{c. } \exists t'(t' < t_0 \land \text{run}(kim,t'))
\end{align*}
\]

The translation process constrains the semantics somewhat, however. In theory it is perfectly possible to make different choices for the two languages. In practice, however, there is much to gain from choosing as similar representations in the two languages as possible. For example, it seems rather unnecessary to write transfer rules for transforming something like (5c) into something like (5a), even though it would be possible.
3.4 Simulations

Are there any similarities between our experiences regarding language technology enriching linguistics and how simulation techniques have entered into other sciences? David Marr’s (1982) view on methodology has had a large impact on artificial intelligence and computer science. In solving what he calls an information processing problem one has to separate the questions about what is processed and why from the questions about how it is processed. The answers to the how-questions can be further split between an abstract representation and algorithm, on the one hand, and the actual hardware and software realization, on the other hand. Marr (1982) also pointed out that Chomsky’s then current syntactic theory with its transformations should be taken to be a what-theory and not a how-theory, and related this to the distinction between competence and performance in linguistics. In a lesser known earlier paper, Marr (1977) drew an additional distinction, between what he called a type 1 and a type 2 theory. Type 1 theories are the “how and what” theories described above. Type 2 theories occur “when a problem is solved by the simultaneous action of a considerable number of processes, whose interaction is its own simplest description.” It is a goal to construct type 1 theories as they add to our understanding, but not all phenomena may have good type 1 theories. In particular, he pointed out that “[v]iewed in this light, it becomes entirely possible that there may exist no Type 1 theory of English syntax of the type that transformational grammar attempts to define”.

In the LOGON project, we have assumed from the outset that syntax has a type 1 theory. Both LFG and HPSG are type 1 theories where a particular grammar is a declarative what-level description, telling what is grammatical and what is to be computed on the how-level. On the other hand, our empirical approach to grammar development with repeated batch testing and stepwise extensions bears some resemblance to the testing of a (model of a) bridge in a wind tunnel. This is a type of simulation. Does this parallel indicate that our grammars are simulations of type 2 rather than type 1 theories? Not directly, since what is simulated in the wind tunnel is the behavior of the bridge under various conditions, while the bridge itself is a constructed artifact. Similarly, the grammar itself is constructed and not (the outcome of) a simulation. The similarity to the bridge shows first and foremost how radically underdetermined the grammatical model is by the observations.

A striking property when a competence grammar is turned into a computational grammar is the large amount of ambiguity, both in parsing and generation. One string of 15–20 words can easily get several hundred different analyses. Still, a human who hears or reads this sentence will most often only get one or
a few different readings. Our use of statistical techniques to rank the outcomes of the parsing, generation and translation, is an attempt to approximate what the human does. Marr pointed out that for a type 2 theory to be interesting, it must show good performance since that is the only way it can show its value. Thus, if our rankings are successful, i.e., correlate well with the preferences of a human, they might be classified as successful type 2 theories. But to get anything like a type 1 theory of how humans choose the best or most likely analysis, we would need more. We would need a theory of what it is the human prefers.

4 Prospects

Computer systems for writing and for parsing linguistically motivated grammars have existed for about 25 years. But it is only during recent years that the formalisms have reached sufficient maturity, the computers sufficient power, and the grammars sufficient coverage for applying the systems on real texts. This has resulted in a new tool for grammar writing and testing, bringing a new dimension to theoretical syntax and semantics. This will gradually alter our views on the status of grammars and semantics. We might also eventually have to rethink our views on the relationship between competence and performance, but by now it is still open where this will lead us.

Acknowledgement

Kjell Johan might be best known for his work on theoretical linguistics: semantics, pragmatics and syntax. But he has also given valuable contributions to several other areas including computational linguistics, corpus linguistics and the methodology and philosophy of linguistics. This article is an attempt to reflect on some connections between these themes.

I would like to thank all the members of the LOGON consortium for the collaboration and discussions which form the background of this paper, and in particular Dan Flickinger and Stephan Oepen, as well as the editors of this volume, for helpful comments.

References


