Dutch from logic (and back)

Crit Cremers

Leiden University Centre for Linguistics
c.l.j.m.cremers@hum.leidenuniv.nl

1 A Generator

In this paper, we present a non-deterministic procedure to generate Dutch sentences with a predefined, fully specified formal meaning. The procedure is grafted on the Delilah parser and generator (http://www.delilah.eu). The input to the procedure is a formula in Flat Logical Form, a fully specified level of semantic representation ([7]). The formula contains only semantic information. The output is a well-formed Dutch sentence with a full grammatical representation, providing again a formula in Flat Logical Form. The logical relation between the input formula and the output formula can be computed.

The main characteristics of the procedure are:

- the input constraint is not biased towards the syntax or the lexicon;
- the generation procedure is non-deterministic, but finite;
- the result can be logically validated: input and output semantics are formulas in the same language

The paper describes the structure of Delilah's generator and the nature of Flat Logical Form. It specifies a method to relate Flat Logical Form to the lexicon and the (categorial) grammar by extracting semantic networks from it. These networks are shown to be able to steer lexical selection and grammatical unification. Finally, the logic of validating the result is explained and demonstrated.

The Delilah system entertains a generator which is driven by a multimodal combinatory categorial grammar of Dutch, dubbed Minimal Categorial Grammar (MCG) in [3]. Its categorization is rigid in that it does not exploit slash introduction – the combinatorial force of Lambek categorial grammars ([5]). The combinatorics of MCG are governed by a limited number of compositional modalities, not unlike the modalities proposed in [6] for Lambek categorial grammars and by [1] for Combinatory Categorial Grammar. The Delilah system applies MCG is also for deep parsing. Both in parsing and in generation, the grammar steers the unification of complex symbols. The unified graph is the main derivational result, apart from a derivational tree (when parsing) and a spell-out of logical forms. An underspecified semantic representation emanates from this unification ([4]).

The generator is hypothesis-driven: it tries to construct a well-formed and meaningful phrase of a given category, with a complete parse in the form of a unified graph representing a complex symbol. The generation procedure is strictly meaning driven without any structural preconditions, as in [2] and [9]. It proceeds by selecting appropriate phrases from the lexicon after inspecting an agenda and by testing their unification. The agenda is fed by the categories of phrases already selected, und updated after successful unification. The generation succeeds if the hypothesis can be checked, no item is left at the agenda and some non-empty structure has been created.

Basically, the algorithm tries to find templates and to unify them according to an agenda which is set by an initial hypothesis and updated by applying combinatory categorial rules. The agenda consists of two parts: *given*, corresponding with complex symbols already adopted, and *to_find*, corresponding to structures still to be checked. A succesful unification of complex symbols according to the agenda is the proper result of the procedure.

2 Flat Logical Form

The input to the generation procedure is a formula in Flat Logical Form (FLF). As an example of an FLF formula, see (1),representing *elke vrouw probeerde te slapen* 'each woman tried to sleep'. Variables are formatted as a quadruple *Variable* + *Monotone* + *Quantifier* + *Governors*. In this index, *Monotone* gives a value for upward or downward entailment for that variable with respect to its predicate, *Quantifier* identifies the binding regime and *Governors* is a (possibly empty) list of variables the valuation of which co-determines *Variable's* valuation.

(1)
```
state(S+↑+some+[A], woman)                          &
theme_of(S+↑+some+[A], A+↓+every+[])                &
event(B+↑+some+[A], try)                            &
property(C+↑+some+[A])                              &
event(D+↑+some+[C], sleep)                          &
experiencer_of(D+↑+some+[C], A+↑+every+[])          &
attime(D+↑+some+[C], E)                             &
agent_of(B+↑+some+[A], A+↑+every+[])                &
theme_of(B+↑+some+[A], C+↑+some+[])                 &
attime(B+↑+some+[A], F)                             &
tense(B+↑+some+[A], past).
```

The classifiers *state, event* and *property* normally come with variable arguments produced by context-dependent, but wide-scoped choice functions; this complication is here left out for the ease of explanation. Lexical concepts are arguments of classifiers and are italicized.

FLF is designed for inference. Here is a, yet incomplete, set of inference rules. The predicates are represented as one-place, by schönfinkelization, for ease of exposition. The inference is given in standard predicate logic, for the same reason, but has an evident counterpart in FLF. In that representation, $P\uparrow$ and $P\downarrow$ represent a super- and a sub-predicate to P, respectively, according to a model or an ontology where $P\downarrow \leq P \leq P\uparrow$. The valuation of variables that are referentially dependent, is handled by wide- scope choice functions (cf. [8]).

(2) FLF logic

Premisse	*to infer*
φ & P(x+↑+Q+[]) & ψ	Qz.P(z) ∃y.P(y) ∃w.P↑(w)
φ & P(x+↑+Q+[$y_1..y_n$]) & ψ	∃f_y.P(f_y(x)) ∃f_y.P↑(f_y(x))
φ & P(x+↓+no+[]) & ψ	¬∃z.P(z) ¬∃z.P↓(z)

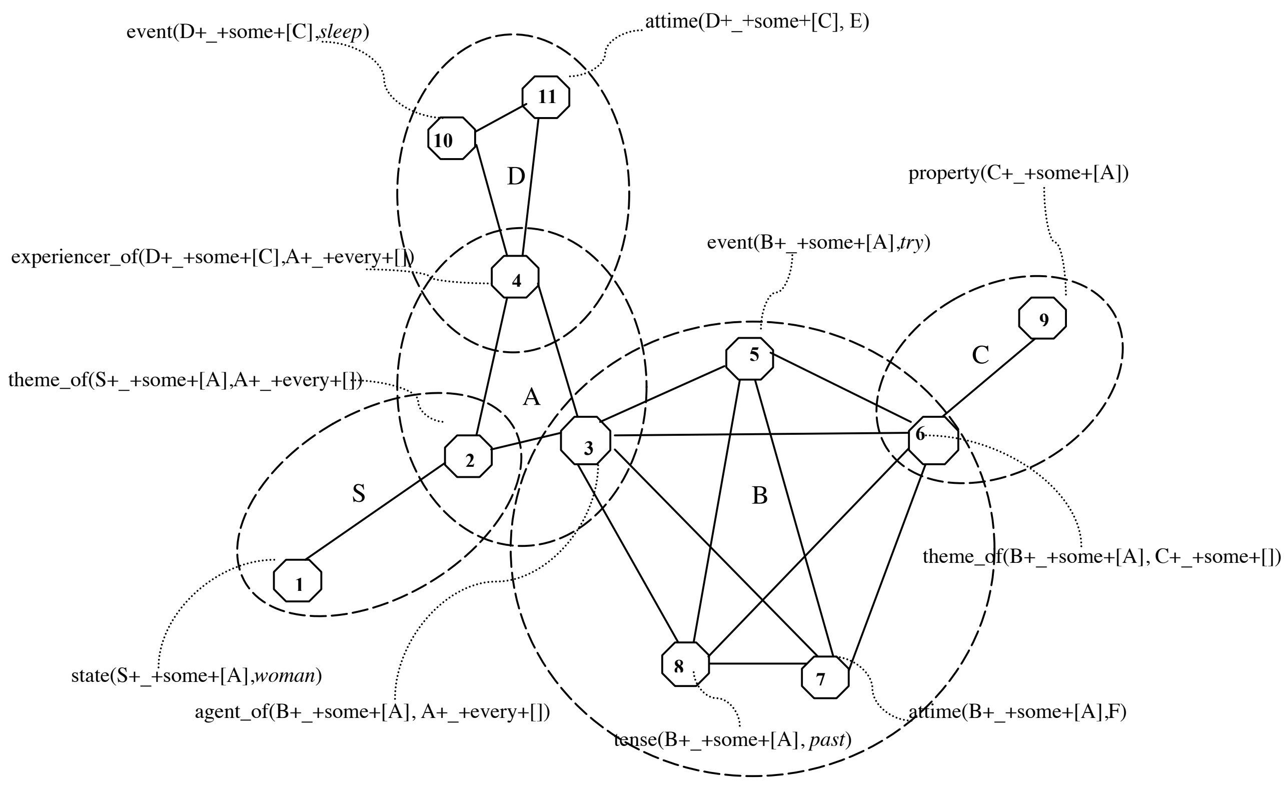
φ & P(x+↓+Q+[]) & ψ
(Q =/= no)
	Qz.P(z)
	Qy.P_(y)
	∃z.P(z)

φ & P(x+↓+Q+[$y_1..y_n$]) & ψ	∃f_y.P(f_y(x))

According to this table, the entailment $\mathfrak{I} \Longrightarrow_{FLF} \mathfrak{R}$ is defined iff both $\mathfrak{I}$ and $\mathfrak{R}$ are in FLF and every clause in $\mathfrak{R}$ can be inferred from $\mathfrak{I}$.

Moreover, every FLF can be described as a connected graph, the small clauses being the vertices which are connected when they share a variable.

(3) the semantic graph of (1)


event(D+_+some+[C],*sleep*)
attime(D+_+some+[C], E)
10
11
D
property(C+_+some+[A])
experiencer_of(D+_+some+[C],A+_+every+[])
4
event(B+_+some+[A],*try*)
5
9
C
theme_of(S+_+some+[A],A+_+every+[])
A
3
6
2
S
B
theme_of(B+_+some+[A], C+_+some+[])
1
8
7
state(S+_+some+[A],*woman*)
agent_of(B+_+some+[A], A+_+every+[])
tense(B+_+some+[A], *past*)
attime(B+_+some+[A],F)

The sets contain clauses that share a variable. The union of these sets defines the lexical space for the generation procedure, to be described in the next section. Each set specifies the constraints on one lexical phrase. An FLF can only be verbalized into one sentence if it is connected in the sense described above and if every small clause (node) has *all* of its specified variables connected.

3 From meaning to form (and back)

The generation procedure is driven by a categorial hypothesis – a hypothesis as to the category of the phrase to be produced. The conceptual agenda strictly limits the freedom of the generator. Every concept is addressed exactly once in a successful generation procedure. Infinite looping is excluded under this simple cancellation agenda.

The procedure sketched above is essentially non-deterministic in at least two senses:

- the (structure of the) FLF is not determining the structure of the sentence;
- the output FLF may not match the input FLF according to a semantic standard.

FLF underdetermines not only its own verbalization – the syntax of sentences realizing that FLF. Because of this 'inverse underspecification', the generation procedure cannot fix all the characteristics of the produced semantics in the logical space in advance or *on the fly*. There are two reasons for the indeterminacy:

- FLF may itself contain less specifications than any verbalization would introduce;
- FLF cannot predict which logical dependencies between variables are blocked or enhanced by following a certain construction mode for the sentence.

The first aspect of semantic underspecification is evident: one cannot be sure that an FLF contains all the information that a full sentence will produce, as it may originate from other sources than language itself. Real-language complex symbols may introduce additional meanings to those mentioned in the conceptual agenda, *e.g.* by default specifications like tense on finite verbs. The concepts in the input are a subset of those in the output. Moreover, the input FLF may not specify semantic dependencies that are inherent to sentential construal. The following FLF, for example, gives rise to the generation of sentences meaning *Every man invites a woman* in a generic reading, but the FLF neither specifies tense nor scope.

The second incongruence between input and output FLF is due to the form-driven nature of sentence meaning. Whether or not a certain operator can scope over another, depends partly, if not mainly, on its syntactic embedding. For example, an operator embedded in a nominal construction has fewer scope options than an operator embedded in a non-nominal, but conceptually equivalent construction. In the same vain, intensional domains are not predictable. Generally, weak and strong islands of any sort are induced by syntax, and the syntax is underspecified, by definition and inevitably. Consequently, the generation procedure cannot be enriched with an additional agenda controlling possible scopal dependencies. Scope can only be checked or compared *post hoc*.

Since FLF – in fact, every purely semantic logical form – contains too little information to fully determine the generation procedure, generating from logic is a trial, by necessity. The outcome of the process can or must be checked against the input constraint. It is important to realize, however, that the input constraint and the output FLF may differ only in a limited number of ways. For example, the output may contain concepts that are not present in the input, but only if these concepts are introduced by default when applying certain complex symbols and if they passed the restrictions on unification imposed by the semantic networks. The output is far from being in free variation with the input.

As was argued above, it is unwise to check for strict identity or equivalence of input and output FLFs. But the analytical structure of FLF offers several options for a well-defined semantic relation to be imposed. Here are a few, for the InputFLF and OutputFLF:

- InputFLF is a (proper) sub-formula of OutputFLF;
- InputFLF and OutputFLF share a (proper) sub-formula containing predefined key-clauses;
- InputFLF and OutputFLF do not entail each other's denials.

Checking a sub-formula property is simple, given FLF's conjunctivist structure. Moreover, it reflects a relatively liberal attitude towards the notion 'sentence meaning' – possibly too liberal. Though the reciprocal denial test is logically much heavier, but decidable on the basis of logic (2), it is even more liberal: accept the output if is does not run contrary against the input.

Taking into account the considerations given above with respect to the 'inverse underspecification', we would propose that the normal check would be as in

(4) Accept S with OutputFLF as a translation of InputFLF into Dutch iff OutputFLF entails InputFLF.

Informally, this means that the generated sentence is at least as specific as the input, or that a model for OutputFLF is also a model for InputFLF, but not necessarily the other way around. Again, it must be noted that the conceptual difference between InputFLF and OutputFLF will be very limited, given the restriction of the lexical resources to those induced by the semantic nets of InputFLF.

If a produced sentence cannot comply to (4), the generator can backtrack or start again. Backtracking has the advantage that all grammatical possibilities will show up, with a degree of efficiency that is determined by the structure of the grammar and the lexicon. A disadvantage of backtracking for the generation task may be that success may require quite a few trials if the source of the incongruence is in early choices. Of course, starting again may follow another track, but every control over the trials disappears.

Under both strategies, the proposed generation procedure guarantees definite qualifications of the result. This is a major advantage of meaning-driven generation with a semantic grammar.

4 Conclusion

In order to generate natural language from full logic, there needs to be no intrinsic relation between the semantic input constraint and the generating grammatical device. The input constraint only requires its concepts to be retrievable in the lexicon. It does not impose syntactic or morphological requirements; they are induced by the generator. Notwithstanding this flexibility, the correctness or effectiveness of the generation can be computed in a formal way, by exploring the logical relation between the input constraint and the output's logical form. But then, there is always Gauß meeting Wilhelm von Humboldt in the early days of the 19th century, according to Daniel Kehlman's *Die Vermessung der Welt;* Von Humboldt - a diplomat - starts masochistically.

...Er sei übrigens auch Forscher ! (...) Er untersuche alte Sprache.
Ach so, sagte Gauß.
Das, sagte der Diplomat, habe enttäuscht geklungen.
Sprachwissenschaft. Gauß wiegte den Kopf. Er woll ja keinem zu nahe treten.
Nein, nein. Er solle es ruhig sagen.
Gauß zuckte die Achseln. Das sei etwas für Leute, welche die Pedanterie zur Mathematik hätten, nicht jedoch die Intelligenz. Leute die sich ihre eigene notdürftige Logik erfänden.
Der Diplomat schwieg.

References

1. Baldridge, J., Kruijff, G.-J. M.: Multi-modal Combinatory Categorial Grammar, *Proceedings of the 10th Annual Meeting of the European Association for Computational Linguistics*, pp. 211 - 218. (2003)
2. Carroll, J., Copestake A., Flickinger, D., Poznànski, V.: An Efficient Chart Generator for (semi-)Lexicalist Grammars. In: Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG'99), pp. 86-95. (1999)
3. Cremers, C.: On Parsing Coordination Categorially. Leiden University, HIL dissertations. (1993)
4. Cremers, C., Reckman, H.: Exploiting logical forms. In: Verberne, S., Van Halteren, H., Coppen, P-A. (eds): Computational Linguistics in the Netherlands 2007. LOT. pp. 5 - 20. (2008)
5. Moortgat, M.: Categorial Investigations. Logical and Linguistic Aspects of the Lambek calculus. Foris, Dordrecht. (1988)
6. Moortgat, M.: Categorial Type Logics. In: Van Benthem, J., Ter Meulen, A. (eds): Handbook of Logic and Language. Elsevier, Amsterdam and The MIT Press, Cambridge, pp. 93 - 177. (1997)
7. Reckman, H.: Flat but not shallow. Towards flatter representations in deep semantic parsing for precise and feasible inferencing. LOT. (2009)
8. Winter, Y.: Flexibility Principles in Boolean Semantics. The MIT Press, Cambridge, MA, USA. (2001)
9. White, M. Baldridge, J.: Adapting Chart Realization to CCG.In: Proceedings Ninth European Workshop on Natural Language Generation. Budapest. (2003)