1

Exploiting logical forms

Crit Cremers and Hilke Reckman
Leiden University Centre for Linguistics (LUCL)

Abstract

This paper presents a semantic setup for Dutch on the basis of deep processing. The parser and generator Delilah computes a system of logical forms that is both semantically adequate, and instrumental in processing tasks like disambiguation and inference. The logical forms are derivationally related but differ as to the level of specification and exploitability. The semantic setup is new, and is likely to be the first computed, fully specified semantics for Dutch. One of the logical forms introduces a new way of compiling out semantic dependencies. The resulting system is discussed at the crossroad of logical semantics and computational linguistics.

1.1 Logical form and grammar

Logical form we take to be that level of linguistic analysis at which lexical concepts, inferential semantics and information structure interact. The required analysis is formal, in the sense that it should account for the intersubjective and — thus — systematic aspects of sentential and lexical meaning. In particular, logical form is a level of grammatical representation at which semantic consequences can be computed — a view already expressed by Higginbotham (1985). Thus, logical form is one of the ultimate targets of linguistic analysis and the representation of what makes natural language a unique module of human cognition. Typically, logical form emanates from deep processing; there are no shallow ways to semantic

Proceedings of the 18th Meeting of Computational Linguistics in the Netherlands, pp. 5–20
Edited by: Suzan Verberne, Hans van Halteren, Peter-Arno Coppen.

precision.

The claim that logical form is formal by necessity, does not imply that it is bound to comply with first-order predicate logic. Predicate logic does not entertain a privileged relation to semantic interpretation: the set of meanings of a natural language is neither a subset nor a superset of the well-formed and interpretable propositions of predicate logic or their complement. It may be a helpful tool in describing certain aspects of the relations between concepts and operators in natural language; in our view, however, it is neither the target nor the anchor of natural language semantics. Logical form must be casted as a representation for which some adequate notion of semantic consequence (entailment) can be defined.

Overviewing modern grammar, it makes sense to state that the relation between logical form and syntactic structure defines the arena. That is not a trivial observation: though linguistics ranks among the elder sciences in the world, it took millennia before the proper balance between form and interpretation was questioned from a grammatical perspective. In recent times, Bertrand Russell (1949) has argued that logical form is not homomorphic to syntactic structure. Generative grammar split on the question with which aspects of meaning grammar could afford to deal (Seuren 1998, ch. 7). In the same period, Montague (1973) defined logical form by compositional interpretation, but correlated it to a pseudo-syntax. Pursuing this semantic perspective, categorial grammar (CG) started to take syntax seriously and to produce interpretations by derivation, deduction or unification (Moortgat 1997, Morrill 1994, Steedman 1996, Carpenter 1998). CG's strong generative capacity, however, challenges the boundaries of linguistic relevance. At the same time, modern grammar theories appear to converge at some formal link between structure and interpretation. The compositional nature of the main semantic configurations is by now undisputed (Heim and Kratzer 1998), that is, if we agree on the lexicon being the only source of semantic wisdom and on meaning being computable at all.

In computational linguistics, logical form is not a very common module of language processing systems. Approaches that avoid to incorporate explicit grammar, deliberately refrain from logical form. Many scholars working on such systems seem to share the scepticism about the computability of meaning that some theoretical linguists entertain. These 'agnostic' strategies govern the field, in our days. As a matter of fact, for each language that is targeted by computational efforts, the number of systems doing semantic analysis is quite restricted. Among these, computation of logical form for inference is rare. Notorious icons here are the HPSG LinGO enterprise (Copestake and Flickinger 2000), the grammars involved in the Verbmobil project (Wahlster 2000), the PARC XLE parser (Maxwell and Kaplan 1993), and DRT-related approaches, like (Bos 2001) and (Bos 2004). Non of these deals with Dutch.

Memory-based language learning as established in Daelemans and van den Bosch (2005), however, does not seem to target propositional interpretation or any other semantic level, till now. Undoubtedly, the problem for these learning approaches is twofold: there is hardly any semantic tagging from which propositional semantics can be induced — there is nothing to be stored or learned — and if it existed, the scarceness of data might be overwhelming with regard to the subtlety that propositional interpretation for inference calls for. The text of the STEVIN program to stimulate the infrastructure for Dutch computation assigns high priority to semantic research but only refers to lexical semantic tagging (Nederlandse Taalunie 2004). Inference — the core business of computational semantics according to Bos (2004) — is not addressed.

Computing logical form is the main objective of the Dutch language processor Delilah. Delilah (http://www.delilah.eu) parses and generates sentences by applying a rigid combinatory categorial grammar, by graph unifying extensive attribute-value matrices and by computing a fully specified semantic representation. The categorial grammar can be viewed as combining Combinatory Categorial Grammar (*e.g.* Steedman (1996)) and Multimodal Categorial Grammar (Moortgat 1997), in the spirit of Baldridge and Kruijff (2003). It was defined originally by Cremers (1993) and argued to be mildly context sensitive in Cremers (1999) and as such described in Cremers (2002). It does not, however, support hypothetical reasoning but entertains syntactically rigid derivations. Thus, it avoids spurious ambiguity at the cost of giving up direct composition of fully specified semantics. As a consequence, in complex symbols syntactic and semantic types may shift. In particular, quantifiers are arguments in syntax but functors in semantics. The grammar is lexicon-driven: all grammatically relevant specifications are stored in the lexicon and subsumed to lemmas. The lexicon specifies *extended lexical units* in the sense of Poß and van der Wouden (2005). In that respect and in the way the lexicon is defined, Delilah — like most lexicalist processing systems — embodies a Construction Grammar (*e.g.* Croft (2001)). The product of parsing and generation is a comprehensive attribute-value matrix in HPSG-style, or a family of these matrices.

Semantically, a derivation in Delilah produces a value to a field in the matrix. This value is an underspecified structured storage of lambda-terms, dubbed Stored Logical Form. A separate algorithm converts these terms into a fully specified Flat Logical Form. In the remainder of this paper, the different functions of these levels of semantic representation in processing Dutch will be discussed.

1.2 Logical form in Delilah

Delilah computes three related levels of logical form: Stored Logical Form (SLF), Normal Logical form (NLF) and Flat Logical Form (FLF).

SLF is *derivational*: it is constructed by applying rules of grammar, which induces unification of complex symbols. It is *compositional* in that it expresses and reflects important aspects of the grammatical structure, and by being built derivationally. Moreover, it is *underspecified*. Important features of the interpretation are not made explicit at SLF. SLF underlies the construal of both NLF and FLF. Specialized algorithms translate SLF in NLF and FLF post-derivationally. Consequently, NLF and FLF are no longer fully compositional in the strong sense: not every aspect of these logical forms is functionally related to the grammatical structure. NLF and FLF both specify all definable features of the interpretation. They differ in the ways of specifying semantic properties. In NLF, matters of scope and semantic dependency are encoded globally and implicitely, as in standard predicate logic. In FLF, scope and semantic dependency are compiled out and made explicit at local levels.

Underspecification tends to be seen as a felicitous feature of computed logical form. It can even be the base of inferential semantics, as was argued in Bos (2001). Bunt (2007) states that full specification has been proven to be intractable. We are particulary interested in underspecification of semantic scope. Like Bunt, Ebert (2005) argues that specifying full scope leads to a combinatorial explosion. A sentence like

A politician *can* fool *most* voters on *most* issues *most* of the time

with five (italicized) operators would lead to 5! analyses. In our view, this may be an artifact of the semantic formalism but not a property of the interpretation. For interpretation, scope is only relevant to determine which terms are (or could be) semantically dependent upon the interpretation of other terms, *i.e.* to find out where to skolemnize dynamic quantifiers. Knowledge of language tells us that there are several types of lexical and local restrictions on semantic dependency in this sense. For example, variables bound by definite quantifiers are for their valuation not dependent on any other variable. Indefinite quantifiers do not influence the valuation of each others' variables. Dynamic quantifiers do not scope out of strong islands. In Delilah, fully specified logical form is derived from underspecified SLF by an algorithm that exploits this type of grammatical knowledge. A comparable strategy is proposed in Koller and Thater (2006). Consequently, NLF and FLF are protected against spurious redundancy while being constructed, by incorporating grammatical knowledge.

Alshawi (1992) explicitly addresses the question why one should entertain multiple levels of logical form. He claims the Quasi Logical Form of the Core Language Engine to be a single level of semantic representation, with additional resolution procedures providing values for free variables. In that sense, our three logical forms also provide one single representation that is constructed in several layers. We claim that these different layers can serve distinct functions. We certainly do not claim that the logical forms arise from different semantic modes. There is one process of logical form construal unfolding at different stages.

1.3 Stored Logical Form

1.3.1 Construal

Each lexical phrase is endowed with a template — an HPSG-style attribute-value matrix — that specifies a structure *Store+Body* as the value of the lf-feature *semantics*. The *Body* is a lambda term, representing the canonical meaning of the phrase. The *Store* contains slots for the logical forms of dependent phrases: a verb stores the lf of its arguments, a preposition the lf of its complement, *etc*. The storage reflects the combinatoric patterns specified in the syntax. Thus, the store contains lambda terms for all dependent constituents, and not just for scope-sensitive operators, as was proposed by Cooper (1983) and for early Montague grammar. For each slot in a *Store*, the variable it operates on in the *Body* is specified. Here is a general scheme for a *Store+Body* structure in a lexical template; all markers are variables, except for the predicates and relations in *Store*.

(1) *Store*: $\{$ DependentLF$_1$#X$_1$, ..., DependentLF$_n$#X$_n$ $\}$
Body: operator$_1$^ ... operator$_q$^relation$_1$(X$_i$, X$_j$) & ... & relation$_p$(X$_k$, X$_m$)

The variables in the *Store* that stand for the logical forms of dependent constituents — *DependentLF$_i$* in (1) — are linked inside the template to these logical forms $(1 \leq i, j, k, m \leq n)$. In general, they are instantiated by *Store+Body* structures themselves in the process of unification. As the outcome of this graph unification, the template of the computed phrase provides a *Store+Body* structure that contains all relevant logical forms of the subphrases involved but underspecifies the interaction of operators. The *relations* in (1) are, in general, constants specified in the template itself.

The store may contain elements that do not correspond to syntactic constituents. In particular, the store of a lexical verb will contain a quantifier that introduces the event characterisation of the verb. This is a typical choice that reflects the level of morphological decomposition fixed in the lexicon and the nature of the morpho-syntactic interface.

For a very simple sentence like *Every man works* the *Store+Body* structure after unifying the relevant templates for *every*, *man* and *works* looks like this:

(2) *Store:* $\{$ $\{$*Store*: $\{$*Store*: $\{\emptyset$, *Body*: λx. *man*(x)$\}$#Q$\}$
Body: λP.$\forall$y. Q(y) $\rightarrow$ P(y)$\}$#S,
$\exists$z. event(z, *work*)#E $\}$
Body: agentof(E,S) & attime(E,T) & time(T,present)

The conversions of the stored lambda terms into a fully specified representation is discussed in the next section about FLF. Here it is only relevant to note that representations for the quantifier, the noun, the verb and the inflectional element occur in a structured and labeled way in the sentential SLF: they are explicitly linked to constituents in the grammatical analysis. In (3), part of the lexical specification of the verb *zag* ‘saw’, as in *Elke man zag Henk werken* ‘every man saw Henk working’, is represented (slightly edited). Links between SLF and the other components of the analysis are underlined.

(3)
```
|ID:A+B
|HEAD:|CONCEPT:see
|     |PHON:ziet
|     |QLF:see
|     |SYNSEM:|ETYPE:state ....
|     |       |PERSON:or([2, 3])
|     |       |TENSEOP:at-pres
|     |       |VTYPE:semi_aux
|PHON:C
|PHONDATA:lijnop(ziet, A+B, [arg(left(1),0,D),
          arg(left(11),wh,E), arg(right(1),9,F)], C)
|SLF:{{[G&(B+H)#I,
      {{[J$ K&(B+L)#M], [], []},
      λN.∃O.quant(O, the) & property(M, O) & entails1(O, decr) & N
         & entails(O, incr)} &(B+L)#P,
      λQ.∃R.(quant(R, some) & see(R) & state(R) & entails1(R, incr)
         & Q & entails(R, incr)))&(A+B)#S], [], []},
      experiencer_of(S, I) & goal_of(S, T)& theme_of(S, P)
      & attime(S, K) & tense(S, pres)}
|SYNSEM:|CAT:s_vn
|       |CONTROL:controls(goal_of~[A+B, T], U~[B+L, T])
|       |PREDTYPE:nonerg
|       |TENSE:tensed ....
|TYPE:s_vn\0~[np^0#B+W, np^wh#B+H]/0~[vp^9#B+L]
|ARG:|ID:B+H
|    |PHON:E
|    |SLF:G
|    |SYNSEM:|CASE:nom
|    |       |CAT:np
|    |       |NUMBER:sing
|    |       |OBJ:subject_of(A+B)
|    |       |PERSON:or([2, 3])
|    |       |THETA:experiencer_of
|ARG:|ID:B+L
|    |PHON:F
|    |SLF:J
|    |SYNSEM:|CAT:vp
|    |       |EXSEM:X
|    |       |EXTTH:U~[B+L, T]
|    |       |THETA:theme_of
|ARG:|ID:B+W
|    |PHON:D
|    |SLF:X
|    |SYNSEM:|CASE:obliq
|    |       |CAT:np
|    |       |OBJ:dirobject_of(A+B)
|    |       |THETA:goal_of
```

In this example, it is clear that whatever plays an active role in SLF, is anchored to the grammatical analysis. In this sense too, SLF is compositional and derivational.

1.3.2 Application of SLF: disambiguation

As said, SLF reflects the morpho-syntactic complexity of the phrase: every constituent that contributes to the meaning of the whole phrase, is represented in the relevant *Store+Body* structure. For this reason, different SLFs of the same sentence correlate to different syntactic ways of composing a meaning for a sentence. This property of SLF can be used to determine to which extent *extended lexical units* or 'constructions' have contributed to an SLF and thus, which degree of lexical aggregation is represented by it. Normally, an interpretation with a high degree of lexical aggregation is to be preferred over an interpretation that was not or less based on extended lexical units. For the sentence *Nobody kicked the bucket till now* the reading *Till now nobody hit the bucket such that it fell* should have a considerably lower priority than the reading *Till now nobody died.* Comparing SLFs in this respect suffices in Delilah to select the most aggregated interpretation. Here is the relevant reasoning and measurement.

It is by now widely acknowledged — and it has never been denied, by all we know — that the lexicon of a natural language processing system not only specifies single words. It must also — and maybe even predominantly — contain phrases or phrase structures that have a specialized syntax and/or semantics. At present, scholars proclaiming Construction Grammar (Croft 2001) stress the central role of non-atomic construal in natural language, up to rejecting an interesting level of syntactic combinatorics. In formal grammar and computational linguistics, however, applicants of HPSG, Categorial Grammar and Tree Adjoining Grammar always seem to have accounted for considerably more involved structures than just atomic units. Poß and van der Wouden (2005), for example, make clear that there is a huge variety of ways in which words and structures cluster to build complexes with specialized syntax or semantics. In general, lexical units that limit lexical selection or syntactic transparency will come with a meaning to which not every proper part contributes according to its proper class. Here are just some examples from Dutch:

- *hebben* with a DP can mean *possess*, among others; together with mass nouns like *honger* and *dorst* it expresses in all its morpho-syntactic varieties the property of being hungry (thirsty); the verb itself hardly contributes to this meaning; the construal *to possess hungriness (thirstiness)* cannot be banned from being parsed;

- prepositional phrases may introduce adjunctive modifiers, but very often verbs select certain prepositions to express specialized meanings, e.g. *werken op iemands DP*, meaning *affect somebody's DP*, where the DP introduces a psychological concept; parsing cannot be witheld from an adjunctive construal, but the selected meaning deserves priority;

- intransitive verbs V can be part of the so-called Dutch *way*-construction, expressing the meaning of moving to a certain position by V-ing, e.g. *to laugh oneself a way/path to DP*; the characteristic DP with the *way*-type NP does not play a role in the semantics.

In all these cases, the *Store+Body* structure representing the meaning of the extended lexical unit, is simpler than the SLF resulting from a parse that does not account for the lexicalized meaning. In particular the store will contain less distinct lambda-terms. Crucially, we do not presuppose that the semantic structure of an extended lexical unit is simpler in any logical or arithmetical sense. All we claim is that the store of an extended lexical unit will show less internal structure than its distributed counterpart. This follows from the construal of SLF. Thus, a simple measurement on the stores of SLFs may select the simplest and thus highly aggregated and most 'normal' reading.

Here is an example. Any sentence containing phrases of the type *honger hebben* will come with two analyses: one reflecting the reading 'be hungry', the other reflecting the reading 'possess hungriness'. The latter SLF will contain a store where the lambda-term for 'hungriness' is specified. The former SLF will have an empty store instead, as the lambda-term for 'to be hungry', whatever its logical complexity is, will not be stored but specified in a body field of SLF. Computing the aggregated complexity of the stores and comparing them, will identify the former SLF as the simpler one. Since this SLF is part of a full analysis, we can select this analysis as the preferred interpretation. The case is illustrated for the sentence *Ik heb honger* 'I possess hungriness' c.q. 'I am hungry', with simplified representations of the *Store+Body* structures for the sake of transparency. The preferred reading has a simpler store than the dispreferred one.

(4) SLF 1 :{ { *ik*:*x* }, `be_hungry(x)` }
SLF 2 :{ { *ik*:*x*, *hungryness*:*y* }, `possess( x, y )` }

Delilah produces all readings permitted by the grammar. It defines for each SLF the structural complexity, *i.e.* the degree of embedding of *Store+Body* structures. For every acceptable parse, it computes a number that expresses the ratio between the structural complexity of the SLF and the total number of elements specified in the stores. This ratio marks the semantic complexity of the SLF. Parses can be ordered according to these ratios. Moreover, it computes the total number of predicates or small clauses in the bodies of the SLFs. This number is used as a secondary marker for semantic complexity, but does not necessarily discriminate between lexical units and semantically composed phrases.

Delilah is trimmed to pop up the simplest reading for a sentence by exploiting the underspecification of SLF. This method is utterly reliable, because it anchors in lexical specifications. In the analysis of the sentence *Sommige kinderen hadden weinig honger* ('some children were a little bit hungry') the *possess*-reading will be suppressed. This is because the lexicon contains an extended lexical unit relating *honger* and *hebben*, the SLF of which is simpler than the SLF constructed from independent lemmas for *to have* and *hungriness*. The SLF for the whole sentence is constructed by sheer unification (*cf.* Reckman (to appear)). Therefore, the complexity of the lexically specified SLFs is conserved in the derivation.

1.4 Flat Logical Form

1.4.1 Construal

Each SLF that passes the complexity test, is submitted to a post-derivational algorithm. This algorithm performs two tasks: it converts the SLF into a coherent formula and it compiles the additional semantic information resulting from this conversion onto each variable.

The first step involves β-reduction of implicitly typed lambda-terms, explicitly specifying scope dependencies between semantic operators like quantifiers, modalities and negation. This conversion is sensitive to scopal variation of operators, to the semantic nature of operators and to structural restrictions on scope imposed by islands or other intervention effects. In this step, a good deal of a semantic theory can be effectuated. The nature of the semantic theory in Delilah is not at stake here, however, as the formalism does not impose constraints on the theory that steers the conversion. It is noteworthy, though, that in Delilah we have chosen to represent all quantifiers by descriptions, thus integrating first and higher order forms of quantification for inference.

In the second step the information spelled out by the conversion is specified onto each occurrence of each variable by a compilation protocol that turns the semantic operators obsolete. After applying this protocol, each variable is locally sugared with information as to

- its entailment property
- the quantificational regime it is bound to
- the variables its instantiation is dependent upon.

The entailment property indicates whether the predicate of which the variable is an argument, allows for upward, downward or no entailment with respect to the variable. The specification of 'governing' variables indicates whether a variable is referentially independent or must be valuated by a choice function; the notion of variables y_i governing a variable x thus amounts to the introduction of a choice function $f(y_1, \dots y_n)$ for x.

Here is an example. The FLF for *Every man works* is produced from an SLF like (2). The outcome of the conversion step can be compared to a standard specified logical form like (5). In (6) then, is the corresponding FLF compiled from it. Each variable is represented by a 4-tuple *variable* + *entailment index* + *quantifier index* + *governing variables*. Semantic constants are italicized, logical constants and standardized relations are in bold face.

(5) $\forall$**y** *man*(y) $\rightarrow$ $\exists$**z.** **event**(z, *work*) & **agentof**(z,y) &
attime(z,t) & **time**(t, *present*)

(6) *man*(y+**down**+**every**+[]) &
event(z+**up**+**some**+[y], *work*) &
agentof(z+**up**+**some**+[y], y+**up**+**every**+[]) &
attime(z+**up**+**some**+[y], t+**up**+**some**+[y] , *present*)

The information encoded on the variables is compiled from an intermediate representation where lambda-terms are fully converted and scopes are specified. The index with respect to entailment—in (6): *up* and *down*—specifies the local monotony properties of the binding quantifier in the relevant domain, according to its definition as a generalised quantifier. For simple, lexical determiners this is straightforward, as these properties are lexically defined. For complex determiners like 'at least n but not more than m' they have to be computed from the composition; in many of these cases no monotony will be detected. This calculus is discussed in Reckman (to appear).

The index with respect to the binding quantifier can be complex, as the quantifier itself is complex. Yet, the intention is to classify also complex quantifiers in such a way that *e.g.* existential impact of the quantifier can be derived immediately. Note that the entailment property of the variable *y* varies with the domain of its quantifier: in the restriction of the universal quantifier, the variable bound by it allows for downward entailment, in the nuclear scope it gives rise to upward entailment. Referential dependency does not vary with domains.

In case an SLF at the first step in its 'spell out' gives rise to real ambiguity, this ambiguity is by definition expressed in terms of scopal dependencies, since SLF is only underspecified for that. FLF takes the form of a single conjunction, representing the complete class of readings, where each conjunct is annotated for the reading(s) it is part of. A conjunct appears twice iff a variable occurring in it may or may not be referentially dependent. To be precise, SLF (2) will yield at the first conversion step two readings, differing with respect to quantifier scope, as in (7). (8) gives the FLF: the conjunction that represents the information of all readings of the SLF.

(7) ∀**y** *man*(y) → ∃**z**. **event**(z, *work*) & **agentof**(z,y) & **attime**(z,t) & **time**(t, *present*)

∃**z**. **event**(z, *work*) & ∀**y**. *man*(*y*) → **agentof**(z,y) & **attime**(z,t) & **time**(t, *present*)

(8) *man*(y+**down**+**every**+[]):[**1**,**2**] &
event(z+**up**+**some**+[y], *work*):[**1**] &
event(z+**up**+**some**+[], *work*):[**2**] &
agentof(z+**up**+**some**+[y], y+**up**+**every**+[]):[**1**] &
agentof(z+**up**+**some**+[], y+**up**+**every**+[]):[**2**] &
attime(z+**up**+**some**+[y], t+**up**+**some**+[] , *present*):[**1**] &
attime(z+**up**+**some**+[], t+**up**+**some**+[] , *present*):[**2**]

1.4.2 Application: inference

FLF is an operator-free conjunction. All logical information is specified as indices on variable occurrences. The representation is inferential: to decide whether a certain inference is possible, it suffices to linearly inspect an FLF and for each conjunct, to decide locally whether or not it gives rise to (part of) the hypothesis.

Suppose we have some model M, with standard definitions for predicates, or an ontology Ω, with a well-defined hierarchy of predicative concepts. Assume that in M or Ω one place predicates P↑ and P↓ are defined and ordered with respect to P as P↓ ≤ P and P ≤ P↑, respectively, where ≤ expresses semantic subsumption. Then, if P(x+_+_+_) occurs as a conjunct in an FLF, inferences like the following immediately hold (f_y denotes a choice function living on y):

(9)	*from* φ & P(x+up+some+[]) & ψ	*infer*	$\exists$z.P(z)	*and*	$\exists$y.P$\uparrow$(y)
	from φ & P(x+up+some+[y]) & ψ	*infer*	$\exists f_y.P(f_y(x))$	*and*	$\exists f_y.P\uparrow(f_y(x))$
	from φ & P(x+down+no+[]) & ψ	*infer*	$\neg\exists$z.P(z)	*and*	$\neg\exists$z.P$\downarrow$ (z)
	from φ & P(x+down+every+[]) & ψ	*infer*	$\forall$z.P$\downarrow$(z)	*and*	$\exists$z.P $\downarrow$(z)
	from φ & P(x+up+every+[]) & ψ	*infer*	$\forall$z.P$\uparrow$(z)	*and*	$\exists$z.P $\uparrow$(z)
	from φ & P(x+down+many+[y]) & ψ	*infer*	$\exists f_y.P(f_y(x))$	*and*	$\exists f_y.P\downarrow(f_y(x))$

That is, for our sentence *Every man works* and well-defined concepts *person*, *young man* and *do something* in M or Ω, each of the following entailments can be made without any additional calculation:

(10) someone does something

every man does something
all young men work
some man does something
some man works

In the inferences with respect to the universal quantifier, we assume with Seuren (2006) that universal quantification in natural language should not be modeled with empty restrictions. Every inference machine, however, may implement its own theory on semantic entailments and logical consequence. That is not at stake here. The point is that FLF offers semantic representation at a level of specification that obsoletes deep inspection of the formula for the sake of automated inference.

Under these conditions, generalized entailment according to the definition of the RTE challenges (Dagan et al. 2005) can easily be defined for FLF.

(11) *generalized entailment*
A text T, represented in FLF entails hypothesis H represented in FLF *iff* H is a finite conjunction of clauses h_i and for each h_i there is a cover φ & p & ψ of T such that *p* entails h_i

Clearly, for each 'small clause' h_i the *p* it needs for entailment is given by its main predicate. We assume that each predicate is explicitly ordered—ontologically or lexically—to not more than a finite number of other predicates. Under that assumption, for each h_i only a linear inspection of T's representation suffices to find a possible antecedent to an entailment relation. If that possible antecedent is found, entailment is checked by inspecting the entailment tables. No serious computing is involved.

There is one complication. Once a partial hypothesis living on a variable X is established, other instances of X in the hypothesis can only be derived under the same set of dependency constraints. This requires some bookkeeping, but does not frustrate the computation. Below, a simple example is given of a one line text *Elke man zag Henk werken* and a hypothesis *Iemand zag iets* 'somebody saw something'. Each part of the hypothesis is inferred from the co-marked clause in the text's FLF. (We assume that *man* is subsumed under *person* and *property* under *thing*, temporal information is ignored.)

(12) FLF(*Elke man zag Henk werken*) =

```
△ man(A+decr+every+[]) &
□ property(B+decr+the+[]) &
  exists(C+incr+henk+[]) &
  work(D+incr+some+[B]) &
  event(D+incr+some+[B]) &
  agent_of(D+incr+some+[B],C+incr+henk+[]) &
  attime(D+incr+some+[B],E) &
◇ see(F+incr+some+[A]) &
♡ state(F+incr+some+[A]) &
♣ experiencer_of(F+incr+some+[A], A+incr+every+[]) &
  goal_of(F+incr+some+[A], C+incr+henk+[]) &
```

```
♠ theme_of(F+incr+some+[A], B+incr+the+[]) &
  attime(F+incr+some+[A], H) &
  tense(F+incr+some+[A], past)
```

FLF(*Iemand zag iets*) =

```
△ person(Z+incr+some+[]) &
□ thing(Y+incr+some+[]) &
♣ experiencer_of(X+incr+some+[], Z+incr+some+[]) &
♢ see(X+incr+some+[]) &
♡ state(X+incr+some+[]) &
♠ theme_of(X+incr+some+[], Y+incr+some+[])
```

On inspection of this type of inference, one can easily conclude that *partial entailment* is defined straightforwardly on the base of (11): a hypothesis <h_1,...h_n> is partially entailed by T if some h_i is entailed and some h_j is not. Partial entailment may induce a whole range of qualifications of the relation between a hypothesis and a text. In particular, an inference engine for FLF combined with an lf-driven generator can construct those subhypotheses that are entailed.

1.5 Normal Logical Form

In our approach, there is no special grammatical task for NLF. It is casted as enriched predicate logic, in the sense that it should be able to accommodate other than strict logical operators and quantifiers. For comparison, we give for a sentence like *Elke man zag Henk werken* SLF, FLF and NLF respectively.

(13) SLF

```
{{[ {{[{{[], [], []}, λI.man(I)}$J&(9+11)#K], [], []}, λL.∃J.quant(J,
every) & K & entails1(J, decr) & L & entails(J, incr) }&(5+9)#M,
{{[{{[ {{[], [], []}, λN.∃O. exists(O) & quant(O, henk) & N &
entails(O, incr)}&(16+99)#P, λQ.∃R.quant(R, some) & work(R) &
event(R) & entails1(R, incr) & Q & entails(R, incr)&(5+16)#S],
[], []}, λT.agent_of [S, P] & attime(S, T))}$U&(5+16)#V], [], []},
λW.∃X.quant(X, the) & property(V, X) & entails1(X, decr) & W &
entails(X, incr)}&(5+16)#Y, λZ.∃A1.quant(A1, some) & see(A1) &
state(A1) & entails1(A1, incr) & Z & entails(A1, incr)&(1+5)#B1 ],
[], [] }, experiencer_of(B1, M) & goal_of(B1, P) & theme_of(B1, &
attime(B1, U) & tense(B1, past) }
```

(14) FLF

```
man(A+decr+every+[]) &
property(B+decr+the+[]).[
exists(C+incr+henk+[]) &
work(D+incr+some+[B]) &
event(D+incr+some+[B]) &
agent_of(D+incr+some+[B],C+incr+henk+[]) &
attime(D+incr+some+[B],E)] &
```

```
see(F+incr+some+[A]) &
state(F+incr+some+[A]) &
experiencer_of(F+incr+some+[A],  A+incr+every+[]) &
goal_of(F+incr+some+[A], C+incr+henk+[]) &
theme_of(F+incr+some+[A], B+incr+the+[]) &
attime(F+incr+some+[A], H) &
tense(F+incr+some+[A], past)
```

(15) NLF

```
quant(A,every).[man(A) → quant(B,the).[property(B).
[quant(C,henk).[quant(D,some).[work(D) & event(D) &
agent_of(D,C) & attime(D,E)]]] &
quant(F,some).[see(F) & state(F) &
experiencer_of(F,A) & goal_of(F,C) & theme_of(F,B) &
attime(F,E) & tense(F,past)]]]
```

Just like FLF, NLF is spelled out post-derivationally from SLF. Contrary to FLF, it is neither a conjunction, nor operator-free. Just like FLF, all semantic dependencies are unambiguously and fully encoded. It is structured by logical operators. NLF is best seen as the logic label of the semantic process. NLF or an impoverished version of it can be used for standard first-order theorem proving.

1.6 Discussion

Delilah computes, apart from a kind of standard full representation, two related but procedural and formally very distinct levels of semantic representation: underspecified, compositional SLF and fully specified, paracompositional FLF. SLF is produced as part of the graph unification which is the kernel of the parsing and generation procedure. Therefore, its construction does not complicate the derivation. Yet, on SLF measures can be defined that express essential semantic properties of the structure and thus can be exploited for selection of readings and reduction of ambiguity. We are still in the process of developing the best and most telling measurements, but it is clear that hardboiled criteria can be found and applied in reducing ambiguity there where it lives: in the semantic structure of a sentence. Of course, not all problems of selecting readings can be handled at SLF. Pure local polysemy can not be decided upon by inspecting SLF. But the delicate balance between levels of lexical aggregation can most certainly be tracked at this level.

Because of its sensitivity to constituent structure, SLF seems a good level for generation to take off. As a matter of fact, we are working on generation algorithms that are fed with SLF and that aim at producing sentences the FLF of which entails or is entailed by the FLF derived from the input SLF. In this sense the labour division between SLF and FLF may contribute to purely meaning-driven translation, as argued in Alshawi et al. (1991) and Copestake et al. (2005). As was argued convincingly in Rosetta (Rosetta 1994), machine translation on a semantic base — this is not a pleonasm nowadays — flourishes through compositionality. FLF approaches the representation of meaning in the spirit of Minimal Recursion

Semantics (MRS) as implemented in the English Resource Grammar (Flickinger et al. 2000, Copestake et al. 2005). The main point of convergence is that MRS and FLF present full semantics while avoiding syntactic complexity of the logical form. There are some differences, however. Since FLF is derived from SLF, all constraints on the interaction of semantic operators — like weak island conditions — are integrated in this derivation. The construal of FLF thus embodies an explicit theory on the syntax-semantics interface. Furthermore, FLF encodes all scopal and inferential information directly onto the logical form, rendering inference strictly local. Finally, in FLF neo-Davidsonian event structure is adopted, for constructional reasons. Extended lexical units may be such that the modification of elements that do not contribute to the meaning of the sentence, is to be interpreted as a modification at the semantic top level. For example, in the constructions of type *honger hebben* 'to be hungry' the mass noun *honger* may carry a determiner that conveys the quantification or degree of the state: *geen honger hebben* (lit: no hunger have) means 'to be not hungry' and *weinig honger hebben* (lit: little hunger have) means 'to be a little bit hungry'. Without quantification over states and events no systematic or even finite treatment of semi-transparent extended lexical units seems possible. In this respect, neo-Davidsonian event structure leads to normalisation and homogenisation of logical form (*c.f.* Reckman (to appear)).

References

Alshawi, H., D. Carter, M. Rayner, and B. Gambäck (1991), Translation by Quasi Logical Form Transfer, *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pp. 161–168.

Alshawi, H., editor (1992), *The Core Language Engine*, The MIT Press, Cambridge, MA, USA.

Baldridge, J. and G.-J. M. Kruijff (2003), Multi-modal Combinatory Categorial Grammar, *Proceedings of the 10th Annual Meeting of the European Association for Computational Linguistics*, pp. 211–218.

Bos, J. (2001), DORIS 2001: Underspecification, Resolution and Inference for Discourse Representation Structures, *in* Blackburn, P. and M. Kohlhase, editors, *ICoS-3, Inference in Computational Semantics*, Buxton, UK, pp. 117–124.

Bos, J. (2004), Computational Semantics in Discourse: Underspecification, Resolution, and Inference, *Journal of Logic, Language and Information* **13** (2), pp. 139–157, Springer.

Bunt, H. (2007), Semantic underspecification: which technique for what purpose?, *in* Bunt, H. and R. Muskens, editors, *Computing Meaning Volume 3*, Springer, pp. 55–85.

Carpenter, B. (1998), *Type-Logical Semantics*, The MIT Press, Cambridge, MA, USA.

Cooper, R. (1983), *Quantification and Syntactic Theory*, Reidel, Dordrecht, the Netherlands.

Copestake, A. and D. Flickinger (2000), An Open Source Grammar Development Environment and Broad-coverage English Grammar Using HPSG, *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Greece.

Copestake, A., D. Flickinger, C. Pollard, and I.A. Sag (2005), Minimal Recursion Semantics: An Introduction, *Research on Language & Computation* **3** (4), pp. 281–332, Springer.

Cremers, C. (1993), *On Parsing Coordination Categorially*, PhD thesis, Leiden University, HIL dissertations.

Cremers, C. (1999), A Note on Categorial Grammar, Disharmony and Permutation, *Proceedings of the 9th conference on European chapter of the Association for Computational Linguistics*, pp. 273–274.

Cremers, Crit (2002), ('n) Betekenis Berekend, *Nederlandse Taalkunde* **7**, pp. 375–395.

Croft, W. (2001), *Radical Construction Grammar*, Oxford University Press, Oxford, UK.

Daelemans, W. and A. van den Bosch (2005), *Memory-Based Language Processing*, Studies in Natural Language Processing, Cambridge University Press.

Dagan, I., O. Glickman, and B. Magnini (2005), The PASCAL Recognising Textual Entailment Challenge, *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment* pp. 1–8, Springer, Southampton, UK.

Ebert, Chr. (2005), *Formal Investigations of Underspecified representations*, Phd thesis, University of London.

Flickinger, D., A. Copestake, and I.A. Sag (2000), HPSG Analysis of English, *in* Wahlster, W. and R. Karger, editors, *Verbmobil: Foundations of Speech-to-Speech Translation*, Springer, pp. 254–263.

Heim, I. and A. Kratzer (1998), *Semantics in Generative Grammar*, Blackwell Publishers, Oxford, UK.

Higginbotham, J. (1985), On Semantics, *Linguistic Inquiry* **16** (4), pp. 547–593.

Koller, A. and S. Thater (2006), An improved redundancy elimination algorithm for underspecified representations, *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, Sydney, Australia, pp. 409–416.

Maxwell, J. T. and R. M. Kaplan (1993), The interface between phrasal and functional constraints, *Computational Linguistics* **19** (4), pp. 571–590, The MIT Press, Cambridge, MA, USA.

Montague, R. (1973), The Proper Treatment of Quantification in Ordinary English, *Approaches to Natural Language* **49**, pp. 221–242.

Moortgat, M. (1997), Categorial Type Logics, *in* van Benthem, J. and A. ter Meulen, editors, *Handbook of Logic and Language*, Elsevier, Amsterdam and The MIT Press, Cambridge, MA, USA, pp. 93–177.

Morrill, G.V. (1994), *Type Logical Grammar*, Kluwer Academic Publishers, Boston, MA, USA.

Nederlandse Taalunie (2004), Vlaams-Nederlands meerjarenprogramma voor Nederlandstalige taal- en spraaktechnologie STEVIN Spraak- en Taaltechnologische Essentiële Voorzieningen In het Nederlands.

Poß, M. and T. van der Wouden (2005), Extended Lexical Units in Dutch, *in* van der Wouden, T., M. Poß, H. Reckman, and C. Cremers, editors, *Computational Linguistics in the Netherlands 2004*, LOT, Utrecht.

Reckman, H.G.B. (to appear), *Flat, not Shallow*, PhD thesis, Leiden University, LOT dissertations.

Rosetta, M.T. (1994), *Compositional Translation*, Kluwer, Dordrecht.

Russell, B. (1949), *The Philosophy of Logical Atomism* , University of Minnesota, Department of Philosophy, Repr. as Russell's Logical Atomism, Oxford: Fontana/Collins, 1972.

Seuren, P.A.M. (1998), *Western Linguistics. An Historical Introduction* , Blackwell Publishers.

Seuren, P.A.M. (2006), The natural logic of language and cognition, *Pragmatics: Quarterly Publication of the International Pragmatic Association* **16** (1), pp. 103–138.

Steedman, M. (1996), *Surface Structure and Interpretation*, The MIT Press, Cambridge, MA, USA.

Wahlster, W., editor (2000), *Verbmobil: Foundations of Speech-To-Speech Translation*, Springer.