From conceptual networks to relative clauses: engineering meaning-driven generation

Crit Cremers

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

March 2024

Abstract

The Delilah parser and generator for Dutch generates - among others - embedded and interpreted relative clauses of any complexity from strictly semantic input. It does so within a grammatical framework that links well-formed sentences to fully specified propositions in a logical calculus, on the basis of a phrasal lexicon. The propositional output can be evaluated against the propositional input. Relative clauses in Dutch show an intrinsic variety of discontinuity. The algorithm neither establishes nor presumes any functional relation between constituency and logical consequence. It is argued that this degree of grammatical freedom is essential for balancing form and interpretation in generation. Consequently, meaning-driven generation and meaning-oriented parsing do not mirror each other.

1. Relative clauses are grammatical islands

Relative clauses are embedded, subordinated sentences. They are interpreted as adnominal adjuncts and licensed by some anaphoric binding to a nominal constituent occurring in another sentence. In many languages, this tight connection is established by a representative of a closed class of functional pronouns, mostly occurring in dislocated position. These *relative pronouns* follow a double syntactic agenda: complying - to wit, agreeing - with the nominal constituent in the upper sentence and fulfilling some role in the embedded sentence. Here are some examples of relativization in Dutch; the relative pronouns and the nominal heads – the antecedents - are underlined, the relative clauses are bracketed.

(1) Wij kopen elk boek [dat hij schrijft]
we buy each book which he writes

(2) Wij kopen elk boek [waarin zijn verhalen staan]
we buy each book in-which his stories are

(3) Wij kopen elk boek [waar zijn verhalen in staan]
we buy each book which his stories in are

(4) Alle auteurs [bij wie depressiviteit centraal staat] worden goed verkocht
all authors for whom depressive-nature central is are well sold

(5) Mijn tante veracht menig uitgever [wiens boeken zij in de ramsj tegenkwam]
my aunt despises many-a publisher whose books she among the irregulars found

(6) Ik heb enkele uitgevers hiervoor gewaarschuwd, [die overigens niet reageerden]
I have some publisher this-for warned who by-the-way not reacted

(7) De uitgever heeft (daar) [waar dat mogelijk was] omstreden termen vervangen
the publisher has (there) where that possible was controversial terms replaced

(8) Dan moet de uitgever ?(datgene) [waar zij over twijfelt] corrigeren
Then should the editor rectify (all that) which she about doubts

There is no intrinsic thematic connection between the antecedent and the relative pronoun: their semantic embeddings are independent. In this respect, relative pronouns are anaphors: they borrow denotation from an antecedent but entertain a conceptual network that is independent of the antecedent and its clause. Furthermore, it is possible for the relative pronoun to absorb the antecedent (cf. (7), (8) above). In that case, the pronoun fulfils two thematic roles simultaneously; the antecedent's role in the higher sentence is assigned to universally or generically quantified but little specified entities, by default. This denotation must be carried by the relative pronoun These so-called free relatives, however, do not offer special problems for generation. In fact, they avoid *pied-piping*, the main source of syntactic and semantic complexity in relativization (cf. below and section 7). The remaining pronoun in free relatives, however, must be able to fulfil two distinct thematic roles simultaneously, which seems problematic in (8).

In Dutch, the relative pronoun is left dislocated itself and can *pied pipe* - drag along - the major constituent it is part of. The latter phenomenon is illustrated in (2), (4) and (5) above by the phrases *waarin*, *bij wie,* and *wiens boeken,* respectively. The left dislocation with or without pied-piping - to what some grammarians may call the specifier position - yields a non-local dependency, possibly crossing sentential domains, as below.

(9) De schrijver [*wiens* boeken$_i$ jij dacht dat _$_i$ uitverkocht waren] verkoopt slecht
the author whose books you thought that sold-out were sells poorly
'The author whose books you thought were sold out sells poorly'

(10) De dag [*die*$_j$ je wist dat _$_j$ zou komen], was ongemerkt voorbij gegaan
the day which you knew that would come had unnoticedly passed
'The day which you knew to be coming passed unnoticedly'

In-situ relativization is impossible; this is almost a negative universal, according to Šimek (2023). Furthermore, the clause introduced by the relative pronoun is a prototypical island: no structure sensitive grammatical process - except for non-reflexive anaphora - operates into, or out of, the right-hand domain of the pronoun. This insight stems from early generative grammar. Actually, the island status has been claimed for the combination of antecedent and relative clause, according to Ross' (1967) Complex NP Constraint. Yet, the relative clause itself may occur in right dislocation or extraposition within the boundaries of its antecedent's clause, as in (6).

The grammar of relative clauses in Dutch has been addressed extensively, though fragmented, by Haeseryn *et al.* (2012) in general terms and by Broekhuis *et al.* (2012-2019) in a generative framework. De Vries (2002) offers a comprehensive theoretical perspective on Dutch relativization. Their findings and analyses are not challenged here.

Syntactically, a relative clause complies with the general sentence structure – an insight also due to early generative grammar. In Dutch, its left periphery is invariably occupied by a relative pronoun or a constituent containing such pronoun. The pronoun itself may be multifunctional or even ambiguous, but its functions for relativization are limited and discernable.
The task of parsing and interpreting relative clauses is as complex as the task of parsing and interpreting *wh*-questions – questions related to an existential presupposition and introduced by a question word. In many languages, the classes of relative pronouns and of question words overlap. Almost all question words can function as relative pronouns, as children know. My four-year-old daughter used to relativize with the pronoun *welk(e)* - common in questions, but hardly ever heard in spoken Dutch relativization.

The parsing problem for relative clauses is slightly more complicated if the clause is not adjacent to its antecedent or if the antecedency is ambiguous:

(11) Toen hebben enkele vaders de kinderen opgehaald [*die* met de fiets waren gekomen]
then have some fathers the children collected who by the bike had come

Semantically, however, the meaning of a relative clause is always to be conjoined to the meaning of the antecedent. To assure coherent interpretation, all predications over an argument are in conjunction with each other, basically (cf. Reckman 2009). It is common practice to interpret nominal constituents as generalized quantifiers - higher order instantiations of a specific relation-in-extension between two predicates, the (nominal) restriction and the (verbal) scope (cf. Westerståhl 2019). This way, the quantifier in an antecedent heading a relative clause is modelled as a double abstraction over algebraically related one-place predicates.

(12) *generalized quantifier*
scheme: $\lambda\mathfrak{P}.\lambda\mathfrak{R}.\ Qx.\ \mathfrak{P}x \otimes \mathfrak{R}x$
for a determiner interpreted by the quantifier Qx – extensional type *<<et, et>, t>* :a relation between predicates
$\lambda\mathfrak{R}.\ Qx.\ Nx \otimes \mathfrak{R}x$
for a determiner phrase - extensional type *<et,t>* : a structure of predicates
examples:
$\lambda\mathfrak{P}.\lambda\mathfrak{R}.\ ALLx.\ \mathfrak{P}x \rightarrow \mathfrak{R}x\ (F)(P)$ *all principles fail*
$\lambda\mathfrak{P}.\lambda\mathfrak{I}.\ SMEx.\ \mathfrak{P}x\ \&\ \mathfrak{R}x\ (F)(P)$ *some principles fail*

Depending on the nature and the sentential embedding of that relation, the relative cause is interpreted within the restriction $\mathfrak{P}x$ or within the co-domain $\mathfrak{R}x$. In the first case - typically with stative quantifiers or quantifiers with a downward entailing restriction - the relative clause is said to be *restrictive* and conjoined to the antecedent's nominal core, in which case it inherits the logical properties of the nominal restriction – see (1)-(5) above. In the second case – typically with dynamic, referential quantification, as in (6) above and with names– the relative clause is called *appositional;* it is interpreted inside the co-domain of the quantifier to which it is linked, in conjunction with the quantifier's semantic image of the upper sentence holding the antecedent. In both cases, the antecedent's quantifier binds the crucial variable in the proposition representing the relative clause. Depending on the quantitative relation, relative clauses may be ambiguous between a restrictive and an appositional reading. In spoken language, this ambiguity is resolved

by prosody. So, the following schemes are available for a proposition *Sx* representing a relative clause, to be incorporated in an antecedent quantifier's frame:

(13) Qx. (Nx & Sx) ⊗ Vx e.g. *all [N professors [S who quit]] [TP are expelled]*
Qx. Nx ⊗ (Vx & Sx) e.g. *these [N professors, [S who are expelled]], [TP quit]*

If the canonical relation ⊗ for the two-place quantifier (12) is symmetric - e.g. disjunctive or conjunctive - the two schemes yield equivalent semantics. If not, the first scheme induces the restrictive reading, and the second the appositive one. The opposition between the readings, therefore, is semantically only relevant for quantifiers imposing an anti-symmetric relation between their restriction *Nx* and their nuclear scope *Vx* - regardless of the prosodic pattern imposed.

This standard approach to the semantics of relativization avoids a syntax-oriented discussion of a clause internal source for the antecedent along the lines of Kayne (1994), the so-called D-complement hypothesis. A generation task does not fare well by derivational complexity beyond need, anyway.

2. Relative clauses can be generated as grammatical islands

While parsing and interpreting relative clauses does not exceed the standard complexity of parsing and interpreting other sentences, the task of generating relative clauses from semantic specifications is far less transparent. Almost by definition, the logical structure of a proposition which interprets a sentence is less complex than the syntactic structure of any sentence raising that proposition. On every sentence, natural language syntax imposes an almost rigid, total ordering in two dimensions: constituency and linearity. This ordering is resource sensitive: any maneuver may affect validity and interpretability. The logical semantics of a sentence is only ordered by variable binding; this ordering is weak and partial. Generation, then, requires hypothesizing and *adding* structure, whereas interpretation-oriented parsing boils down to *untwining* strict linearization and discontinuity. Thus, generating syntactically complex sentences from logical propositions alone is an uphill journey, with no shortcuts available. The fact that relative clauses are grammatical islands has some advantages for generation, however. It implies that relative clauses can be generated by an independent procedure operating on a real subset of concepts and relations chosen from the input constraint. Both from a syntactic and from a semantic point of view, the generation of a relative clause is a closed shop - *salve* anaphora, which can invade into any domain.

It is relevant to stress here that the concepts in input and output need not to be primitive. A concept can denote an operational object from any independent, semantically relevant library or archive. Concepts may denote spaces in a structured ontology, lemmas in an encyclopedia, or multidimensional vectors. In Delilah's grammar, a concept is just a name for an object like these. It is assumed to carry along all its computable properties, which can be addressed at any moment in the derivation or generation. So, the meaning of a sentence is constructed as a term over the field of relations between the concepts addressed in that sentence. It instantiates *secondary semantics*, the model-invariant semantics of consequence (Kracht 2003), or the sentence's *logical*

form (LF). In Delilah's grammar, that semantic term amounts to a conjunction of (disjunctions of) standardized two-place relations under first-order quantification and with negation. The disjunctions result from negating a conjunction, whereas a disjunction in the input constraint gives rise to a disjunction of full sentential propositions. (14) below specifies the scheme of a standard flat logical form.

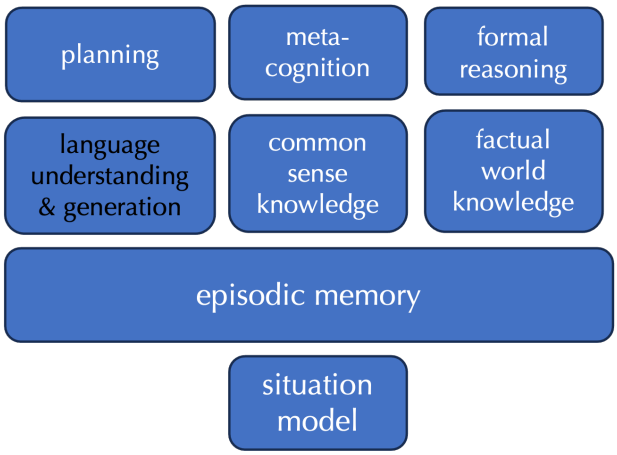
(14) p_1 ....& [q_1 ....∨ q_m] ... & p_n
where each p_i *holds a relation* $r_i(t_{2i}, t_{2i+1})$ *and*
each q_i *holds a binary relation* $r_i(t_{2i}, t_{2i+1})$ *or its negation* $\neg r_i(t_{2i}, t_{2i+1})$ *and*
each t_{2i} *is a bound variable* X⊗ *indexed for quantification, polarity and scope, and*
each t_{2i+1} *is a bound variable* X⊗ *or a conceptual constant*

In this paper, I describe an algorithm in the Delilah parser and generator for Dutch (Cremers, Hijzelendoorn and Reckman 2014, Reckman 2009, https://delilah.leidenuniv.nl). In one of its modes, the automaton produces Dutch sentences, the meaning of which entertains a computable and logical relation to a given, initial proposition. As such, Delilah is a **v**ery **s**mall but **m**eaningful **l**anguage **m**odel, a VSMLM. These days, one is almost bound to explain why a language system is not a derivative of a **l**arge **l**anguage **m**odel, an LLM. The main difference – besides scale, of course - is the focus on *propositional meaning* and the modular organization of *grammatical knowledge*. The VSMLM Delilah lives on explicit, computable, objectivized, operational and exportable meanings as inalienable attributes of sentences. In order to generate sentences from meaning, we must assume meaning to be something more substantial than a probability or an undetermined phase: the proposition raised by a sentence may be largely underspecified, but it is not undetermined. Delilah leans on the assumption that in language communication, the proposition is immanent or emerging rather than conjectured or circumstantial. This assumption is rooted in the observation that the predicative structure of any sentence hardly ever gives rise to confusion. If propositional confusion arises, we have a case of looming miscommunication or blatant underspecification, rather than a case of being trapped by the genuine vagueness of language. We may consistently question any aspect of a sentence *iff (*and *because*) we agree on its proposition, and precisely to that extent. In present LLMs, however, propositional meaning is an undefined accidental phenomenon at best, beyond grasp or grammar. In these models, propositional meaning is neither explicit nor computable nor objectivized or quantized. That is no surprise. Propositional aspects of meaning are unlikely to be learnable from data other than profound propositional annotations, for an unannotated string of words bears little information as to its propositional meaning. The comprehensive annotation system necessary for deep semantic learning, however, is not yet available in our days (but see Abzianidze *et al.* (2023) for a slightly more positive assessment of this state of affairs), despite important initiatives like meaning banks. Semantic annotation is certainly not incorporated in modern LLMs - which highlights their overwhelming impact. Paradoxically, deep semantic annotation for learning purposes requires a sophisticated level of grammatical analysis, whereas, in fact, avoiding grammar seems to be the main drive behind these gigantic enterprises.

While identifying linguistic meaning as an essential part of the language enterprise, a VSMLM effort fits well in Tom Dietterich's design of the next generation artificial intelligence systems. Its components are listed below (from Dietterich 2023). The author argues convincingly that for an AI system to be accountable, linguistic knowledge should be available and operational,

independent of, but in open relation with, other cognitive components. In my view, this implies that meaning-driven grammar is called for, while propositions are traded along the system.

(15) *Dietterich's next generation language/ai model*



In the following sections, the relation between sentences and propositions in the Delilah automaton is presented against the background of the idea that syntax and semantics are non-bijective in natural language grammar. The nature of the asymmetry in generating relative clauses is elaborated on. In sections 8, 9 and 10, finally, the design of the algorithm which creates well-formed relative clauses under semantic control is illustrated and discussed.

3. Constituents and entailments do not project each other

Delilah operates a rigid categorial unification grammar that accounts for both syntactic linearization and an underspecified semantic representation. The grammar is described in Cremers, Hijzelendoorn and Reckman (2014). The relevance of underspecification at some stage of semantic derivation is clarified by Bunt (2008). The semantic representation produced by the unification grammar is an unconverted, implicitly typed higher-order lambda term. Its post-derivational conversion into a logical proposition is subject to constraints, accounting for domain-specific and construction-specific conditions like island-hood. These constraints compare to PTQ's post-derivational rules, the *meaning postulates* (Montague 1972): they restrain conversion and composition on base of grammatical considerations. Such a protocol can also be found in the $=_q$ specifications of Minimal Recursion Semantics (Copestake *et al.* 1999): instructions how to apply semantic terms post-derivationally. The unification grammar lives on a detailed phrasal lexicon or *constructicon*, providing graphs which specify all relevant syntactic, semantic, and morphological properties of a phrase. At the end, the Delilah parser assigns a family of fully specified, logical propositions to any Dutch sentence covered by its grammar, in a multi-stage procedure exploiting the unification of complex symbols. The Delilah generator aims at mapping propositions of this very type onto well-formed Dutch sentences that provably entail the initial proposition. The input constraint feeding into to the generator is a proposition with one of three origins:

- it is the (partial) result of a parsing task, *e.g.* a semantic consequence of the LF of a parsed sentence, or

- it is the conclusion of some reasoning over LFs assigned to text, or
- it represents some extra-linguistic state of affairs.

The input's origin is not relevant for the generation procedure as such, although a non-linguistic origin may be expected to be less specific with respect to semantic dimensions like tense and aspect. It is a major challenge for the generation algorithm, however, to deal with a reasonable degree of underspecification. In Dietterich's modular architecture of future AI (15), linguistic precision may not be decisive for propositions to be traded. As an upper limit of underspecification, we impose the restriction that the input constraint consists of at least two small clauses, *i.e.* two relations from a closed set of linguistically relevant predicates, hinted at in the description of flat LF (14). For propositions start to be linguistically meaningful - translatable into sentences more informative than 'something happens' or 'there is something' - as soon as at least one (bound) variable occurs twice: one occurrence introducing a concept and another occurrence relating the concept to something else. So the minimal input to a successful generation procedure is a proposition like (16), in some standard calculus, or like (17), in the flat notation introduced in (14):

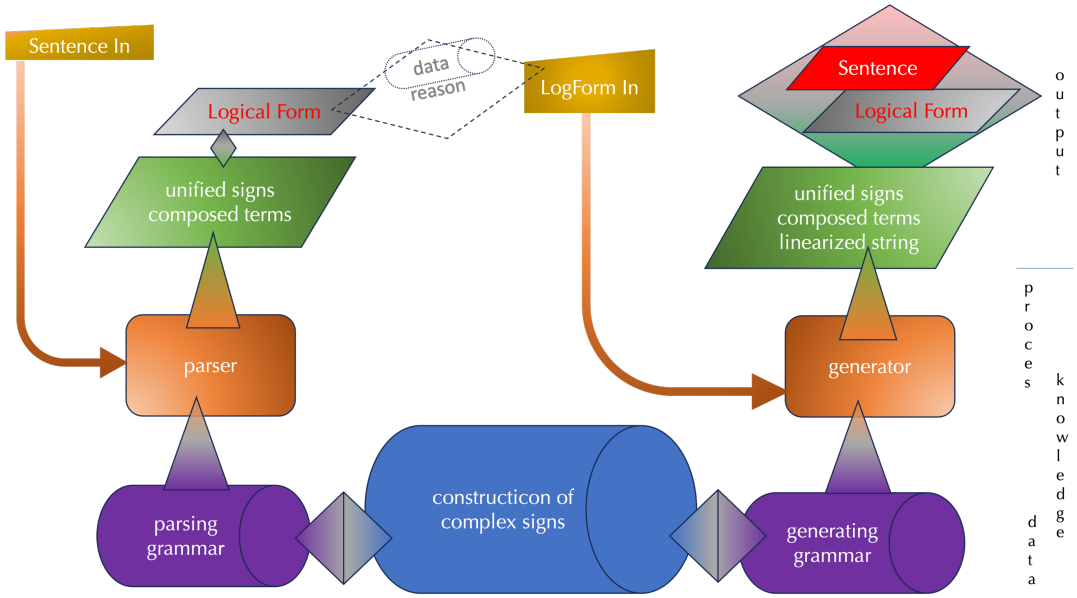
(16) $Qx.\ r_i(x_m, a) \odot r_j(b, x_m)$
*(*a, b *random terms compliant with* r_i *and* r_j *resp.,* Q*x a quantifier,* ⊙ a connective*)*

(17) $r_i(X^{\otimes}{}_m, a)\ \&\ r_j(b, X^{\otimes}{}_m)$

The requirement that at least one variable occurs twice, reflects the minimal connectivity of a proposition representing the meaning of a sentence. Under a neo-davidsonian regime for natural language meaning representation - all predicates are introduced by a dedicated quantifier - even the smallest sentence is semantically represented by a proposition of type (16) or (17), or an extension of these. Since entailments are only verifiable as natural language sentences, they too must stem from minimally connected logical forms, consisting of at least two conjoined small clauses with at least one variable occurring twice, as indicated above.

Overall, Delilah establishes a grammatical cycle of parsing, manipulating propositions, and generation: parsing complex Dutch sentences *into* logical form, reasoning *with* logical form and generating well-formed and interpretable sentences *from* logical form. Scheme (18) depicts this cycle. Intrinsically, any generated sentence comes with a full grammatical structure, and its LF can be evaluated against the input constraint. Since this check amounts to an occurrence check of small clauses in utterly flat propositions, the evaluation is straightforward.

(18) *the cycle of form and meaning*



The parser and the generator apply the same constructicon and equivalent but inverted grammars. Yet, parsing – trying to assign an interpretable formula to a given sentence – and generation – trying to verbalize a given proposition – are basically different processes. In Delilah, parsing and generation do not mirror each other. The two processes are not traceable to a common algorithmic core. This is neither accidental nor opportunistic. It implements the viewpoint that syntax and semantics have non-compliant systemic characteristics, and that good syntax and good semantics diverge by necessity. Good syntax provides all and only well formed and interpretable strings by means of constituent structures. Good semantics provides all and only real semantic consequences by means of a logic. These two tasks cannot be executed in parallel or in symmetrical translation, since constituents and entailments do not match, although the processes leading to linearization and semantic consequence can be lined up partially. Elsewhere, we argued at length in favor of this divergence (Cremers, Hijzelendoorn and Reckman 2014), referring to it as a coherent grammar's form of incompleteness. Here, I will just sum up the most salient aspects of this point of view:

- Syntax defines linearization by constituency, and semantics specifies meaning by providing logical consequences. Constituents, however, do not project entailments, and entailments do not project constituents. If constituents would project entailments, their combinatorics cannot consistently account for linearization. If the proposition is to project constituents, it cannot fully account for its semantic consequences.
- Syntax is conservative, constructive, and resource sensitive, best modelled by unification. Entailment-oriented semantic specification requires the full power of the lambda calculus, a type-zero device.
- Syntax defines a total order over constituents. Entailments only enjoy a partial order.

To illustrate these points, consider the syntax and semantics of *except*-clauses like

(19) The philosophers were drunk except for Socrates.

For this sentence, an elementary syntactic phrasing and a list of some compelling semantic consequences are given below; words introducing lexical concepts are in italics.

(20) [[the *philosophers*] [were *drunk* [except for *Socrates*]]]
(21) *Socrates* was not *drunk* (and)
Socrates was a *philosopher* (and)
Some *philosophers* were *drunk*
Not all *philosophers* were *drunk*

From this listing of simultaneously valid and therefore conjoined entailments, one can see that no proper constituent of the sentence projects any entailment by itself and that no entailment lives on a single proper constituent. Enriching syntax in such a way that a constituent may project an entailment (ultimately a sentence, too), however, makes linearization and constituency a void enterprise; it would imply, for example, deleting or inserting the syntactic counterparts of negations and quantifiers. In the same vein, there does not seem to be any reasonably restrictive class of logical-semantic characterizations of *except* which could pre-arrange all the semantic consequences of a sentence governed by it. A characterization like this should, for example, be sensitive to the (relative) position, the structure, and the function of the quantifiers targeted by *except*. The operator and its complement, however, can occur almost anywhere in a randomly complex sentence - notably both to the left and the right, both above and below that quantifier - without semantic consequences. Hence, the simple sentences (20)-(21) already reveal the algebraic mismatch between constituency and entailment. Even under conditions of structural completeness - every analysis comes with a full range of valid alternatives, as in Lambek categorial grammar (cf. Moortgat 1988) - it seems nearly impossible to find transparent homomorphisms linking constituents and entailments, as I will try to illustrate below. Nevertheless, both domains are covered by every language user's knowledge, and are therefore as real as can be. Comprehensive natural language processing must deal with both constituents and semantic consequences.

4. Meaning-driven generation is not compositional

This approach to the relation between syntax and semantics suggests a fundamental asymmetry in the processing of a natural language's grammar. Going from syntax to semantics in the realm of a constructicon, the algorithms select and lose information. Some constraints are exploited and subsequently left behind, *e.g.* left-right ordering, aspects of phrasal dominance, and agreement. Going from semantics to syntax, however, information and structure must be added: the partial ordering of a proposition is converted into the total ordering of syntax. The syntactic structure of a sentence is more delicate and more rigid than the structure of the logical object it covers, the proposition. In this vein, meaning-oriented *parsing* can be seen as a process of *increasing entropy*. That is natural business. Meaning-based *generation*, however, is a process of *decreasing entropy* – some structure must be injected, hypothesized, even enforced, as it can not be provided by the input proposition.

Seen this way, meaning-based generation is like blowing against the wind - it requires additional algorithmic resources beyond formal grammar and the information in the constructicon.

Meaning-driven generation requires careful procedures balancing between concepts and combinatorial categories – a theme elaborated on in section 9. Additionally, the asymmetry implies that meaning-driven generation is not deductive, whereas deduction in the Gentzen sequence format is the main proof procedure of formal grammars in the categorial spectrum (Van Benthem 1991). The asymmetry also indicates that generating a sentence from logical form is not deterministic. The outcome is easy to check against the input, however, because flat logical forms (as in (27)) are maximally transparent and fully specified locally.

This view on the delicate balance between syntax and semantics may seem to run counter to Montague's achievements, as in Montague (1972), and to the compositional tradition in their wake, as in multi-modal categorial grammar. No such flaw is intended, however, and here are three arguments to that effect.
(i) In Montagovian semantics, the eventual computation of semantic consequences over first order intensional formulas only partially depends on the derivation of the typed higher order terms underlying these formulas. To see why, let (22) be a well typed lambda term for a given sentence, reflecting its Montagovian derivation, and let (23) be the exhaustive conjunction of that sentence's *n* logical consequences such that no p_i entails p_j. The first can be seen as the *applicative*, the second as the *concatenative* representation of LF, in the terminology of Desclés (2004). The concatenative representation is, as I see it, the one that feeds into the non-linguistic modules of Dietterich's AI-model (15).

(22) $L^1(L^2(L^3, ...), L^m)$
(23) p_1 & ...& p_k ... & p_n

For the lambda-free conjunction (23) to be compositionally derived in its turn, there must be a mapping $f(L^i)$ to subsets P_k of the set of logical consequences assembled in the conjunction. In section 7 it is conjectured that such a mapping would be very hard to construct, and if constructible, it would violate grammatical adequacy. As a matter of fact, stored higher-order terms as in (22) tend to underspecify logical consequence, for the scope of modal and quantificational operators is not fully determined in an unconverted term, whereas their scope is a crucial factor in the computation of entailment. Moreover, any functional relation between specific constituents contributing unconverted lambda terms and specific entailments emanating from converted lambda terms vaporizes with the spurious derivational power (cf. the type lifting of Hendriks 1993) that is needed to establish a coherent term like (22) and the inevitable application of post-derivational meaning postulates, as in Montague (1983).
(ii) Compositional syntax - syntax with a typological eye on logical form - may be quite sloppy. PTQ's treatment of relativization, for example, avoids syntax, as it analyses *such that*-paraphrases instead of left dislocation relativization. To arrive at correct linearization, compositional treatments of relativization - as provided *e.g.* by Morrill (2017) following Morrill and Valentin (2010) - heavily lean on upgrading combinatoric power, with little or no impact on semantics. The present generation-oriented algorithm is dedicated to relativization and comparable forms of complex predication, the linearization effects of which stay within the boundaries of Delilah's restrictive and rigid standard syntax of Dutch. Nevertheless, the underlying grammar also exploits multimodal combinatorics - conditionalized composition - and may simulate Morrill's calculus, but without enlarging its expressive potential.

(iii) Compositional semantics is a parsing derivative. Even in deductive multimodal categorial grammar, lambda terms and phrases are immediately connected in a derivation. There is no 'level' of independent inspection, analysis, or account of semantic terms. The semantic formulas are not identified without string combinatorics. The envisaged process which respects compositionality is parsing, not generation. In generation, however, combinatory properties other than relations between concepts are not given but hypothesized. Thus, classic compositionality does not appear to open any window on generation from logical structure - it is designed to assign meanings to strings, not to serve meaning-driven generation of strings. Compositional grammar in the Montagovian tradition is a one-way track. In addition, it seems hopeless to try and find a topological transformation from (23) to (22).

The relation between derivation and entailments in generation is also addressed in section 8.

5. Logical form has many faces

In the Delilah grammar, syntactic structure is as elaborate as is necessary to control constituency, linearization, agreement, morphology, and unification of semantic terms. Semantic structure, on the other hand, is as flat as possible. Basically, the semantic representation of a sentence is a conjunction of disjunctions of standardized small clauses specifying the building blocks of presuppositions and entailments. The complex sentence (24) is assigned the rigid categorial constituent structure (25), as well as the standard logical form (26) and its ultimate derivative, the flat logical form (27). Relative pronouns have a twofold combinatorial category, one to serve the relative clause and one to connect the clause with the antecedent, as proposed by Cremers (2004) - cf. constituents 6-6, 5-11 and 3-11 in (25) below. The logical forms arise from conversions which are prepared, but not directed, by syntax. Below, both semantic representations are slightly shortened for clarity; conceptual constants are in boldface, and variables are in capital.

(24) Ik laat elke jongen met wie jij niet hebt kunnen werken, een boek lezen
I let every boy with whom you not have been-able-to work a book read
'I have every boy with whom you could not work, read a book'

(25) *constituent structure of (24)*

```
1-14:s\wh~[]/1~[]  [ik,laat,elke,jongen,met,wie,... werken,een,boek,lezen]
   1-1:np\0~[]/0~[]  [ik]
   2-14:s\0~[np^wh]/1~[]  [laat,elke,jongen,met,wie,... een,boek,lezen]
      2-11:s\0~[np^wh]/1~[vp^6]  [laat,elke,jongen,met,wie,jij,...werken]
         2-2:s\0~[np^wh]/0~[np^0,vp^6]  [laat]
         3-11:np\0~[]/1~[] [elke,jongen,met,wie,jij,niet,hebt ... werken]
            3-3:np\0~[]/0~[n^0]  [elke]
            4-11:n\1~[]/1~[]  [jongen,met,wie,jij,niet,hebt,kunnen,werken]
               4-4:n\0~[]/0~[]  [jongen]
               5-11:n\0~[n^0]/1~[]  [met,wie,jij,niet,hebt,kunnen,werken]
                  5-6:n\0~[n^0]/1~[s_vn^1]  [met,wie]
                     5-6:n\0~[n^0]/0~[s_vn^1]*s_vn\0~[]/1~[s_vn^1][met,wie]
                        5-5:s_vn\0~[]/0~[np^0,s_vn^1]  [met]
                        6-6:n\0~[n^0]/0~[s_vn^1]*np\0~[]/0~[]  [wie]
                  7-11:s_vn\1~[]/1~[]  [jij,niet,hebt,kunnen,werken]
                     7-7:np\0~[]/0~[]  [jij]
                     8-11:s_vn\0~[np^0]/1~[]  [niet,hebt,kunnen,werken]
```

```
                        8-8:s_vn\0~[]/0~[s_vn^8]  [niet]
                        9-11:s_vn\0~[np^0]/1~[]  [hebt,kunnen,werken]
                           9-9:s_vn\0~[np^0]/0~[vp^59]  [hebt]
                           10-11:vp\0~[]/1~[]  [kunnen,werken]
                              10-10:vp\0~[]/0~[vp^9]  [kunnen]
                              11-11:vp\0~[]/0~[]  [werken]
      12-14:vp\1~[]/0~[]  [een,boek,lezen]
         12-13:np\0~[]/1~[]  [een,boek]
            12-12:np\0~[]/0~[n^0]  [een]
            13-13:n\0~[]/0~[]  [boek]
         14-14:vp\0~[np^0]/0~[]  [lezen]
```

(26) *standard logical form of (24)*

```
∀A:state(A,male) ∧ state(A,young) ∧ ∀B:state(B,possible) →
ιC:proposition(C) ∧ ∃D:event(D,work) ∧ agent_of(D,y) ∧ attime(D,..) ∧
evidence(C,possible) ∧ ∃E:relation(E,with)  ∧ attime(E,..) ∧
[¬(theme_of(B,C)) ∨  ¬(attime(B,..)) ∨ ¬(aspect(B,perf)) ∨
¬(tense(B,pres)) ∨ ¬(theme_of(E,B)) ∨  ¬(goal_of(B,A))] → ∃F:event(F,let) ∧
ι(G):property(G) ∧ ∃(H):event(H,read) ∧ ∃J:state(J,book) ∧ agent_of(H,A) ∧
theme_of(H,J) ∧ attime(H,..)) ∧ agent_of(F,i) ∧ goal_of(F,A) ∧
theme_of(F,G) ∧ attime(F,..) ∧ tense(F,pres)
```

(27) *flat logical form of (24), derived from (26)*

```
state(A:∀:↓:[],male) & state(A:∀:↓:[],young) & state(B:∀:↓:[],possible) &
proposition(C:ι:↓:[A,B]) & event(D:∃:↑:[A,B,C],work) &
agent_of(D:∃:↑:[A,B,C],y) & attime(D:∃:↑:[A,B,C],..) &
evidence(C:ι:↑:[],possible) & relation(E:∃:↑:[A],with) &
attime(E:∃:↑:[A],..) & [¬(theme_of(B:∀:↑:[],C:ι:↑:[])) ∨
¬(attime(B:∀:↑:[],..)) ∨ ¬(aspect(B:∀:↑:[],perf)) ∨
¬(tense(B:∀:↑:[],pres)) ∨ ¬(goal_of(B:∀:↑:[],A:∀:↓:[])) ∨
¬(theme_of(E:∃:↑:[A], B:∀:↑:[]))]  & event(F:∃:↑:[A],let) &
property(G:ι:↓:[A]) & event(H:∃:↑:[A,G],read) & state(J:∃:↑:[A,G],book) &
agent_of(H:∃:↑:[A,G],A:∀:incr:[]) & theme_of(H:∃:↑:[A,G],J:∃:↑:[A,G]) &
attime(H:∃:↑:[A,G],..) & agent_of(F:∃:↑:[A],i) &
goal_of(F:∃:↑:[A],A:∀:↑:[]) & theme_of(F:∃:↑:[A],G:ι:↑:[]) &
attime(F:∃:↑:[A],..) & tense(F:∃:↑:[A],pres)
```

The constituent structure embodies a double rigid hierarchy of containment and linearity. Both logical forms – flat logical form in particular – are largely unordered but connected by a network of variables (the capitals in the formulas). Flat logical form (27) (see Reckman 2009) compiles out the logical dependencies raised by logical form (26) and specifies the polarity of each predication explicitly. In a flat logical form like (27), the fourth component of a variable specification quadruple *<Variable : quantifier : monotony : list>* marks those variables the valuation of Variable depends on. This reflects quantifier scope, which is otherwise absent in this representation. Note that in standard logical form, the interdependency of bound variables is the main ordering force. Once this dependency is compiled out, the order between clauses connected by symmetrical operators becomes futile; the linear order of the sentence, at least, does not impact it.

The flat logical form is construed as a conjunction of disjunctions of small clauses expressing standardized semantic properties and relations, as defined in (14). Sentential negation gives rise to a disjunction of negated small clauses, representing the scope of the nonveridical operator; this

is marked by square brackets in boldface. (Though this provision is not yet implemented in Delilah's present package, lexical and structural ambiguity could also give rise to disjunctive clauses: a meta-disjunction of local ambiguities of any kind, one of which is contextually chosen to establish a reading. Furthermore, it introduces intensional domains, labelling them as a property or a proposition, with an additional bound variable. The packed lambda term resulting from unification of the respective lexical lambda terms and underlying the logical form, however, is too complex and too untransparent to print here - it comprises a hierarchy of 21 lambda operators and over 40 variables. Its complexity is a good reason to assume that this stored (or packed) object is not the target of interpretation; for an additional argument to this vein, see Haruta *et al.* (2022). The compositionally packed pre-conversion lambda term represents merely a stage in the interpretation procedure - it is an egg, not yet a chicken.

In parsing, the task of assigning a semantic representation starts with the unification of semantic terms specified in the constructicon. Subsequently, structurally and typologically controlled lambda conversion is applied to the aggregated terms. This yields (a family of) conjunctive propositional formulas like (26) or (27) in a logic for which semantic consequence is defined. In this process, the syntactic structure evaporates, and entropy increases.

As stated before, generation from logical form involves a completely different task: given an intrinsically connected propositional formula with a partial order, find those well-formed structured sentences which, when parsed, entertain a logical form entailing the initial formula. The generated proposition must be at least as specific as the input constraint, without introducing concepts that are not available in the constraint. This process involves the imposition of a total order and thus a decrease of entropy. Going from a relatively chaotic network to a class of well-structured sentences calls for strategies different from those suited to go in the opposite direction. This is particularly true if both moves are planned within the same constructional framework. A common grammatical base for parsing and generation is an important prerequisite for coherency and control in a natural language automaton. For verbalizing and testing logic entailments, we need a generator that transforms formal propositions into sentences which we can evaluate empirically and experimentally. For natural language interaction, as well as for *e.g.* experimental models of acquisition (Shakouri *in prep.*), we need parsing and generation algorithms, the yields of which can be evaluated by humans. For meaning-driven computational linguistics to become an empirical branch of science, we had better establish trajectories from sentences to meanings to sentences, as sentences are the theatre of meaning and the only expressions which human beings silently agree upon - semantically.

The remainder of this paper discusses an algorithm that generates highly structured, Dutch relative clauses from flat logic within the lexical and constructional frame of the parse (24) - (27).

6. Dutch relative pronouns have distinct distributions

Relative clauses in Dutch are postpositional and enjoy SOV word order. The complementizer position is absorbed by the obligatory relative pronoun - cf. (29). The major constituent which this pronoun raises or is embedded in, occurs mandatory and exclusively in the leftmost (specifier) position of the sentence. Some relative pronouns, however, allow for preposition stranding - cf.

(32). The interpretation of the relative clause as appositional or restrictive does not correlate with any syntactic difference, neither structurally nor positionally. Most relative pronouns can absorb the antecedent, yielding free relative clauses, with positional properties equal to those of an explicit antecedent and interpreted as a definite quantifier. Of course, free relatives cannot be extraposed.

(28) De docent [die jouw scriptie heeft beoordeeld] was overspannen
the professor who your thesis has graded, was overstressed
'The professor who graded your thesis, was overstressed'
(29) * De docent [die heeft jouw scriptie beoordeeld] was overspannen
the professor who has your thesis graded, was overstressed
(30) De docent [bij wie jij bent afgestudeerd] is ontslagen
the professor [with whom you have graduated] has-been fired
The professor with whom you graduated, has been fired
(31) * De docent [jij bij wie bent afgestudeerd] is ontslagen
the professor [you with whom have graduated] has-been fired
(32) De docent [waar jij bij bent afgestudeerd] is ontslagen
the professor [whom you with have graduated] has-been fired

In Dutch, relative pronouns come in a dazzling variation, most of them agreeing with the antecedent in one of more features (or absorbing it). All wh-question pronouns are also relatives, but not the other way around. The pronouns *die* and *dat* are exclusively relatives and determiners, agreeing with an obligatory antecedent in number and gender. They only occur as direct arguments to verbs. The relative pronoun *wie* is highly sensitive to conceptual properties of the antecedent and an argument to prepositions exclusively, without facilitating preposition stranding. Its neutral counterpart *wat* is singular only and goes with -- or is limited to, according to some speakers - non-nominal antecedents; it occurs only as a full argument to a verb. The relative pronoun *waar* is both an adjunct and a left occurring nominal argument to a preposition; it does not agree with the antecedent, although some speakers prefer a non-human or even non-animate antecedent. *Waar* as an argument is the prototypical facultative preposition strander. In spoken Dutch, the pronoun *welk* (variants: *welke*, *dewelke*, *hetwelk(e), welks*) is increasingly old-fashioned as a relative pronoun, occurring as a determiner reintroducing the antecedent or as a full nominal argument to a verb or a preposition. It agrees with the antecedent in number and gender but can barely introduce free relatives. Nevertheless, *welk* and its derivates may be viable alternatives to *waar* as a non-stranding prepositional argument.

(33) De schilderijen op welke /waarop /waar het museum al een bod _/_/op heeft ontvangen, worden onderhands verkocht
the paintings for which the museum already a bid has received are privately sold

Just like before-mentioned *wat*, the pronoun *hetgeen* can head appositional relative clauses with a non-nominal antecedent:

(34) De auto zou gestolen zijn, *wat* de politie evenwel ontkende
'The car was said to be stolen, which the police denied, however'

(35) Toch heeft geen minister deze mails gearchiveerd, *hetgeen* in strijd is met de regelgeving
'Yet, no minister archived these mails, which runs counter to the rules'

The sheer existence of this type of relativization, by the way, calls for an event-based semantics, as is adopted in Delilah's semantics.

The relative determiners *wiens* and *wier* ('whose') express a relation between their antecedent and their nominal argument, and occur exclusively as pied-pipers (see (2), (4), and (5)). They agree in number and sex (rather than grammatical gender), while being restricted to human or animate antecedents according to some; they allow free relativization, though not easily:

(36) Wiens brood men eet, wiens woord men spreekt
whose bread one eats, whose word one speaks
'(Every)one speaks after the one who feeds him
(37) ?* Wiens brood men eet, moet de kwaliteit ervan waarborgen
whose bread one eats, must the quality of-it guarantee

Besides *waar,* there are other adjunctive pro-forms which can head a relative clause, with or without (standard) antecedent, and as a free relative, with or without expletive and extraposition:

(38) Hoe zij dat oploste was indrukwekkend
how she it solved was impressive
'It was impressive how she solved it'
(39) Het was indrukwekkend hoe zij dat oploste
it was impressive how she solved it

None of the combinatorial properties of relative pronouns as described above, is referred to in a logical form underlying generation, as that logical form is meant to be language independent. The input constraint on generation may stem from logical manipulation of contextual meanings, may represent non-language data, or can represent the meaning of a sentence in another language. None of these sources is likely to provide combinatorial clues for Dutch syntax or lexicalization. The issue is also discussed in section 8.

7. Pied-piping in relative clauses is non-deterministic

Whenever a relative pronoun is not a major argument in the relative clause - a direct argument in the sentence's verbal complex - its quest for the leftmost position in the sentence may require a constituent it is part of, to move along. This phenomenon, *pied-piping*, is also prominent in wh-questions. It complicates the construction of a relative clause from logical form considerably. Just like the combinatorial properties of wh-pronouns mentioned in the preceding section, we cannot expect pied-piping specifications to be derivable from the input constraint - a logical form - because we cannot expect constituency to be univocally encoded in LF. Pied-piping affects nominal and prepositional constituents - the specification of which is simply out of LF's range. But pied-piping lives on constituency. When it is obligatory or inevitable as well as when it is facultative, weighing the constituent structure is crucial in determining which constituent is to

occupy the leftmost position in the sentential structure. To illustrate this point, consider the following sentences.

(40) * De politicus wie jij nooit artikelen over wou lezen, wordt niet herkozen
the politician whom you never articles about wanted-to read is not re-elected
(41) ... over wie jij nooit artikelen wou lezen, ...
about whom you never articles wanted-to read, ...
(42) .?... artikelen over wie jij nooit wou lezen, ...
(43) *... wie jij artikelen over hebt vernietigd, ...
who you articles about have destroyed
(44) ? ... over wie jij artikelen hebt vernietigd, ...
(45) *? ... artikelen over wie jij hebt vernietigd, ...
(46) De partij waar jij nooit artikelen over wou lezen, is nauwelijks corrupt
the party which you never articles about wanted-to read is hardly corrupt
(47) ... waarover jij nooit artikelen wou lezen, ...
(48) .?... artikelen waarover jij nooit wou lezen, ...
(49) .?.. waar jij artikelen over hebt vernietigd, ...
(50) .?.. waarover jij artikelen hebt vernietigd, ...
(51) *? ... artikelen waarover jij hebt vernietigd, ...

In (40)-(45), the relative pronoun *wie* is shown to require pied-piping and to disallow preposition stranding. The opposition between (41) and (44), however, shows that the verb imposing the sentence's argument structure may be relevant in determining to which phrase pied-piping can or must apply. In (46)-(51), the relative pronoun *waar* is shown to occur with and without preposition stranding, even leaving the preposition inside a nominal constituent, as in (49). In Dutch, pied-piping of the maximal nominal constituent containing the wh-element, is not preferred, as demonstrated by the stylistically heavily flawed (45) and (51) - the verb's argument structure plays no role here. De Vries (2006) dubs this *heavy pied-piping* and observes that it works out more smoothly with prepositional constituents than with nominal ones.

The data above reveal that relativization and questioning in Dutch can cause nominal and prepositional constituents to become discontinuous: a wh-element at the left periphery of a sentence may have to be interpreted as part of a nominal constituent at its far right, and a part of a nominal constituent may occur at its far left, in the sentence's specifier position. Generating relative clauses under semantic constraints must keep the balance here. In this vein, pied-piping is an operational grammar's litmus test; keeping track of meaning implies an account of pied-piping. Contemporary large language models, though, will handle pied-piping without any hint to meaning.

The following cases of pied-piping must be accounted for in a grammar of Dutch.

A. *The wh-element is the argument of an adverbial pre- or postposition*. Pied-piping is obligatory or facultative, depending on the status of the wh-element.

De docent *[Sr [PP bij* wie *] hij* e *logeerde]*...
* ... *[Sr [DP* wie*] hij [PP bij* e*] logeerde]*...
Het bedrijf *[Sr [PP* waar*-voor] hij* e *werkte]*...
... *[Sr [DP* waar*] hij [PP* e *voor] werkte]*...

B. *The wh-element is part of an adnominal prepositional adjunct*. Pied-piping is obligatory or facultative, depending on the wh-element. If pied-piping comes up, usually the prepositional phrase is affected, but the dominating nominal phrase can go along too, especially in appositional relative clauses.

De kunstenaar [Sr [PP over wie *] hij [DP colleges e] had bijgewoond], ...*
the artist about whom he classes had attended
**De kunstenaar [Sr [DP* wie *] hij [DP colleges over* e *] had bijgewoond, ...*
Elke auto [Sr [PP waar-*voor] hij [DP geen vergunning* e *] kon betalen], ...*
every car for which he no permission could pay
Elke auto [Sr [DP waar *] hij [DP geen vergunning voor* e *] kon betalen], ...*
? Die soldaten ..., [Sr [DP de uitrusting voor wie *] trouwens incompleet was]*
those soldiers the equipment for whom by-the-way incomplete was

C. *The wh-element is realized as a definite determiner*. Pied-piping of a relevant covering constituent is obligatory. The relation introduced by this determiner, can also be introduced by a preposition or a possessive pronoun, with different pied-piping options (cf. De Vries 2006). Here too, prepositional phrases pied-pipe more smoothly; judgements as to grammaticality and style may vary.

De dichter [Sr [PP om toegang tot [DP wiens *lessen]] hij had* e *gevraagd], stierf*
the poet for access to whose lessons he had asked died
? ... [Sr [PP om toegang tot [DP wie z'n *lessen]] hij had* e *gevraagd]...*
? ... [Sr [DP toegang tot [DP wie *z'n lessen]] hij had* e *gevraagd]...*
... [Sr [PP om toegang tot [DP de lessen van wie *]] hij had* e *gevraagd]...*
... [Sr [PP tot [DP wiens *lessen]] hij (om) toegang* e *had gevraagd]...*
? ... [Sr [PP tot [DP wie z'n *lessen]] hij (om) [toegang* e] *had gevraagd]...*
? ... [Sr [PP tot de lessen van wie] *hij om toegang* e *had gevraagd]...*
? ... [Sr [DP wiens *lessen] hij (om) toegang tot* e *] had gevraagd]...*
** ... [Sr [DP* die/wie z'n *lessen] hij (om) toegang* e *tot* e *had gevraagd]...*

D. *The wh-element is a direct argument of a main verb* (finite or under auxiliary command). Pied-piping of a dominating - necessarily verbal - non-infinitival constituent is not possible

Elk boek [Sr [NP dat *] jij* e *niet hebt gelezen]...*
each book which you not have read
**Elk boek [Sr [VP* dat *gelezen] jij niet hebt* e *]...*

E. *The wh-element is an argument of (a preposition in) a non-finite verbal constituent*. Pied-piping (of the verbal constituent) is facultative and is facilitated by prepositions and r-relatives

Het bedrijf [Sr [S waar-*voor te werken] zij ons afraadde* e *]...*
the company for which to work she us advised-against
... [Sr [S voor hetwelk *te werken] zij ons afraadde* e *]...*
? ... [Sr [PP waar-*voor] zij ons afraadde* e *te werken]...*
? ... [Sr [S dat *te benaderen] zij ons aanbeval* e *]...*
which to approach she us recommended
? ... [S r [S hetwelk *te benaderen] zij ons aanbeval* e *]...*
... [S r [DP dat *] zij ons aanbeval* e *te benaderen], ...*

F. *The relative clause has a non-nominal antecedent*. Depending on the choice of the pronoun, light pied-piping is either obligatory or possible. Heavy pied-piping, though, is hardly viable.

Hij is doorgereden, [Sr [PP waar-*voor] hij flink is* e *beboet]*
he has driven-on for which he considerably has-been fined

Zij hielp hem, [Sr [PP voor hetgeen*] men haar* e *rijkelijk heeft beloond]*
she helped him for which people her richly have rewarded
? Er is niet geanticipeerd [Sr [DP de schade waar-*door]* e *in de miljoenen loopt]*
there has-been not anticipated the damage by which in the millions runs

This overview of pied-piping might discourage grammatical estheticism. Some logically sound relativization schemes may be syntactically impossible in Dutch. The phenomenon is hardly systematic across languages, anyway. It is very well possible, though, that acceptability of (dis)continuity in relativization is gradient with all kinds of semantic properties of the phrases affected (cf. Kluender 1998). Moreover, the pied-piping phenomena interfere with island constraints and even with parasitic gaps (Morrill 2017). The linearization schemes are intricate and barely fed by the semantic constraint on a generation task. Syntactic opportunism, or better: consistently weighing the conceptual and the categorial agenda against each other while checking structural hypotheses, appears to be a preferable strategy for generating relative clauses. The grammar engineer can opt for avoiding relativization in certain cases, and finding alternatives in explicit conjunction, for example.

8. Generating relative clauses implies creating syntactic structure

As hinted at before, a formula φ potentially representing the meaning of a sentence, cannot possibly contain any clues as to the syntax of sentences carrying that meaning, for two main reasons:

- in the target language, certainly more than one sentence will project φ, or rather: if there is a sentence that means φ, there are many sentences with meaning φ, showing a considerable variety of lexical and syntactic features
- if φ represents the meaning of a sentence in one language, it represents the meaning of sentences in many distinct, if not all languages.

Each individual reason suffices to render syntactic or constructive clues in a propositional formula feeding into a generation process obsolete. The expectation that syntax emanates from logic, runs counter to the enterprise of grasping natural language semantics computationally. In Delilah, therefore, the formulas that constitute the parsing result and the input constraint to generation are taken to be completely free and independent of syntax. The formulas neither reflect nor project sentential structure.

Still, going from meaning to sentences is part of the empirical cycle in deep natural language processing. For human beings to evaluate a meaning, it must be phrased as an interpretable sentence. The first step in trying to associate a proposition with a sentence is unfolding and inspecting the network of relations between the concepts in the proposition. For each concept introduced, the relations it entertains with other concepts are collected. Then, the Delilah generation algorithm picks one of the concepts with the richest network. The constructicon is explored to find a predicative instantiation - finite or infinite - of that concept which seems apt for carrying the relational load imposed by the initial proposition. The chosen construction is used as the onset of an agenda to realize the constraint's other concepts and relations (Cremers 2004, Cremers, Hijzelendoorn and Reckman 2014). The process succeeds whenever a well-formed

sentence of the envisaged category is constructed, and its meaning has a well-defined logical relation to the input formula. Because generation operates against entropy, the meaning of the targeted sentence will be at least as specific as the formula the generation starts from. Thus, at least one of the meanings of the generated sentence – the semantic formulas assigned to it by parsing – should entail the input constraint, the initial proposition. Since entailment is supposed to be defined for the logic in which the formulas are cast and because their structure is utterly flat and conjunctive, this relation between input proposition and output proposition can be checked in a straightforward manner, algorithmically or by hand. This check is necessary, as meaning-based generation from logical form cannot be deterministic because of the lack of structural clues in the input. Generation from logical form is not a homomorphism.

Once a starting concept for the generation has been picked, two dimensions of the agenda have to be balanced in each subsequent step: one, the structural requirements regarding constituency and linearization, i.e. well-formedness, imposed by the construction-so-far, and two, the adoption of the conceptual net not yet addressed. Scheduling these two dimensions is the core business of the generating algorithm. They both introduce strict conditions on the sentence's building plan without offering decisive clues as to their unification or balancing.

Relative clauses present themselves as fulfilling the requirements of *complex predication*: a concept that entertains semantic, e.g., thematic, relations with two or more other concepts. For a proposition to be translatable into a sentence, every concept in the proposition must be related to at least one other concept. If at inspection of the proposition a concept turns out to play multiple thematic roles, the generating algorithm is bound to try and realize each of these relations properly within a well-formed phrase which projects a semantic term covering that relation and is connected to the other predicative complexes in which the concept occurs. Not every double thematic specification will result in a relative clause, of course, since complex predications may surface in several ways. The first step in the generation, therefore, is a grammatical evaluation of the relevant resources, resulting in a constructive strategy. This strategy can - and under some conditions: must - be relativization, but it may also invoke other forms of additional predication.

In the proposition, the multiple roles of the antecedent's concept are in conjunction with each other, as conjunction is the proposition's main constructor (cf. Reckman 2009). This complies with the undisputed practice to interpret a relative clause as an additional predication conjoined to the clause's nominal head, besides its role in a higher sentence (cf. (12)-(13)).

As soon as a concept is called for by the predicative structure-under-construction, its other predicative obligations must be evaluated. To extend the construction, absorption of the concept by a relative clause is an important option. Firstly, the algorithm checks the requirements imposed by the primary predication on the form and function of the antecedent. Subsequently, the full network defined by the secondary predication is isolated in a conceptual sub-agenda, aiming at the construction of an independent subordinated clause. The isolation of this agenda is warranted by the conjecture that relative clauses are syntactically and semantically islands, except for the outbound agreement of the relative pronoun. From the isolated network, the concept involved in the relativization is picked and constructed. This construction introduces a syntactic agenda that in combination with the isolated conceptual network is exploited towards clause formation, according to the regular backbone procedure for generation. Finally, the position for the relative

pronoun is filled with a form that matches both the likes of the antecedent and the structural and morphological requirements of the newly created clause, thus realizing the secondary predication. Again, this is a non-deterministic process, and it can possibly fail, for all kinds of reasons but mainly because of lexical gaps. In that case, another strategy for realizing the secondary predication is called for. If it succeeds, however, the resulting clause with a relative pronoun is properly inserted in the upper clause by unification. In the final semantic round, the relative clause's contribution to the semantics is tested as an integrated part of the semantics of the complete sentence. The logical relation of entailment between input constraint and the final sentence's propositional meaning can only be evaluated at the level of full specification, that is, post-derivationally. Of course, the conceptual agenda active in the generation procedure, is meant to prevent too drastic aberrations from the intended logical space, but it cannot act as an evaluation tool on the fly.

9. The generation procedure is steered by two agendas

Below, I'll give a summary of the incremental procedure leading from input logical form (52) to sentence (53). The input constraint can be cast in first order predicate-logical format but is always translated into the flat-logical format, to comply with Delilah's constructicon. The constraint does not have to specify every attribute of every variable for a sentence to be generated. In (52), monotony and dependency are largely underspecified. So are tense and aspect. Full specification of the input constraint is not necessary for the generation algorithm to do its job. The output proposition, however, represents the semantics of the sentence generated and consequently, is specified for every semantically relevant property of every concept. In general, the semantic specification of the input constraint is at the most as rich and detailed as the output proposition. For this reason, in a successful generation procedure the output proposition entails the input constraint - the output may be more restrictive than the input constraint. This relation also assures input of non-linguistic origin to be processed successfully.

(52) *input constraint: flat logical form*

```
experiencer_of(A:∃,C:ι:_:[]) & event(A:∃,see) & state(B:∃:_:[],car) &
state(C:ι,man) & theme_of(A:∃,B:∃:_:[]) & agent_of(K:∃, C:∀) &
event(K:∃,sing) & at_place(K:∃,B:∃:_:[]) & state(C:∀,woman)& tense(
A:∃,past)
```

(53) *sentence generated from (52)*

```
deze vent zag een auto waar iedere vrouw in zingt
this chap saw a   car  which every woman in sings
```

(54) *output proposition: flat logical form of (53)*

```
state(B:∃:↑:[],car) & state(C:∀:↓:[],woman) & event(D:∃:↑:[C],sing) &
agent_of(D:∃:↑:[C],C:∀: ↑:[]) & attime(D:∃:↑:[C], ..) &
tense(D:∃:↑:[C],pres) & theme_of(F:∃:↑:[],D:∃:↑:[C]) &
relation(F:∃:↑:[],at_place) & location_of(D:∃:↑:[C],B:∃:↑:[]) &
attime(F:∃:↑:[], ..) & state(A:∃:↑:[],see) & tense(A:∃:↑:[], past) &
state(H:ι:↔:[],man) & experiencer_of(A:∃:↑:[],H:ι:↔:[]) &
theme_of(A:∃:↑:[],B:∃:↑:[]) & attime(A:∃:↑:[],..)
```

The process of generating sentence (53) with proposition (54) from (52) as the opening constraint, given a phrasal lexicon and a combinatorial grammar, runs as follows.

(55) *generation algorithm for complex sentences in Delilah*

(a) The input constraint is compiled into a network of concepts and variables (see also (61) below). The concept *see* (variable A in (52)) is identified as the one with the largest network which is not 'governed' by other concepts, and is therefore taken as a starting point for the top-down generation of a sentence of category S. The choices of a starting concept and of the overall category are parametrized. (In the present network, the concepts *car* or *sing* could also have been chosen to start from; this would lead to sentences like *elke vrouw zingt in een auto die de man zag* 'every woman sings in a car that the man saw').

(b) From the lexicon, a verbal construction with concept *see* is chosen which offers a perspective of satisfying the needs from its network, like offering argument positions with suited thematic impact, and introducing a track to sentencehood – the designated target.

(c) The structural agenda introduced by the construction found in step (b) and the conceptual agenda found in (a) are explored to integrate other phrases. The balance between the two agenda's is parametrized throughot the procedure.

(d) To satisfy the *theme of*-relation for *see*, the concept of *car* (variable B) is called for at some stage in the generation. B's network appears to be complex: the concept is also involved in another thematic relation, to wit the location of *at place*. The complete network spanned by this concept, including the connected subnets of the concepts *sing* and *woman*, is put aside for relativization.

(e) In s*ee*'s frame, a viable construction for the concept *car* is sought, identified, and integrated into that frame by unification.

(f) In the subnetwork defined by *at place*, one of the richly connected concepts is selected to build a sentence from. In this case, *at place* itself qualifies. It is instantiated by a structural frame introducing subordinated-sentencehood.

(g) A dedicated agenda is defined and activated for controlling the generation of a subordinated sentence around a relative pronoun which agrees with the *car*-antecedent in a proper argument position. The agenda applies the general generation procedures hinted at in step (c).

(h) The constructed relative sentence is integrated into the upper sentence containing the antecedent. The semantic term computed for the embedded sentence is stored in the developing semantic term of the antecedent by unification; with this, it is integrated in the semantic term of the whole sentence.

(i) Whenever the procedure strands, it tracks back to the last point of choice.

This way, the double semantic and syntactic loyalty of the antecedent concept is accounted for. In the final proposition - the sentence's logical form - the relative clause is, of course, only discernable by tracing semantic nets. This illustrates the entropy gap between the highly structured sentence and the relatively flat underlying proposition .

Parsing the produced sentence yields the same proposition as generating it did. In this respect, generation and parsing converge linguistically, notwithstanding the divorce of syntax and

semantics. The entropic asymmetry is resolved by grammar-based engineering: early adoption of a fully specified complex symbol from the constructicon steers the generating agenda.

The very same procedure can lead to any of the following alternatives to (53), all associated with proposition (54), of course, though some will come with an even more specific proposition, entailing this one.

(56) *selection of sentences generated from (52), other than (53)*

(a) `enkele auto's waar in alle vrouwen zingen, had de man gezien`
a-few cars which in all women sing had the man seen

(b) `gezien had dit mannetje enkele auto's in welke elke vrouw zingt`

(c) `deze mannen zagen een auto waar iedere vrouw in zong`

(d) `door de ventjes waren enkele auto's waar elk meisje in zingt gezien`

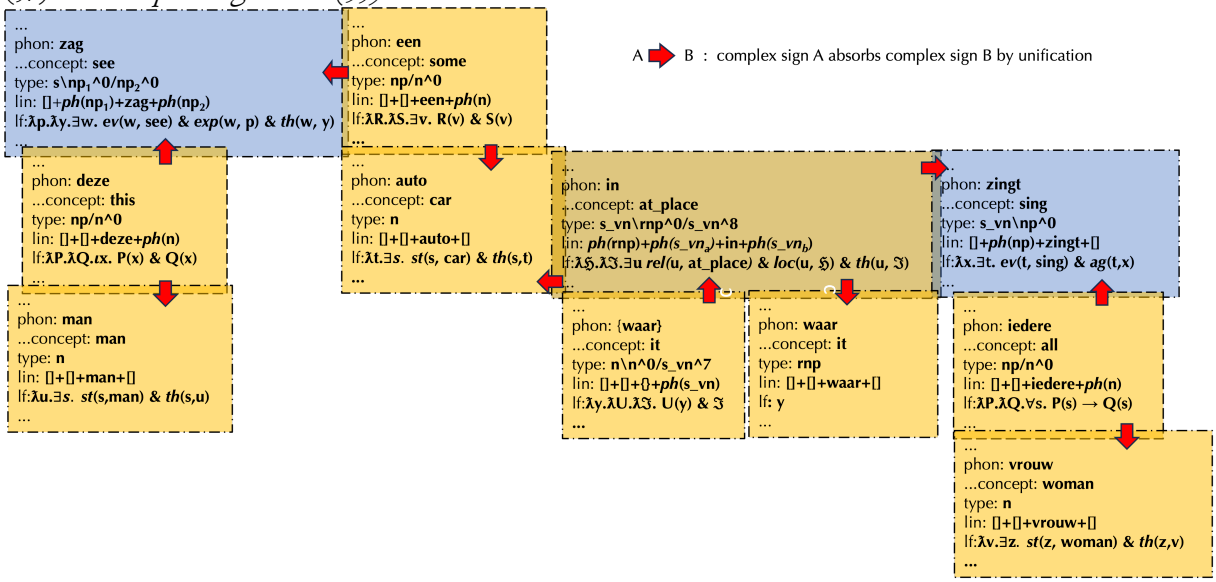
(e) `de mannen hadden een auto gezien waar alle vrouwen in zongen`

(f) `in de autootjes die de man had gezien, zongen alle vrouwen`
in the little-cars which the man had seen sang all women

The last of these sentences results from choosing another starting concept by different parameters, as hinted at in step (a) of algorithm (55). The algorithm promotes the starting concept to the main predicate in the construction-to-be.

Procedure (55) has some phases that can be clarified visually. Scheme (57) depicts the class of complex signs from the constructicon that underlies the sentence and the proposition to be generated. Here, the complex signs are represented only partially. A full sign is a structured term containing all grammatical information that is available for a certain phrase.

(57) *complex signs for (53)*



Some features of the generating grammar are relevant here. Firstly, the linearization term *lin* in each sign consists of four connected fields: *left_periphery + local_left + head + local_right*. These fields are addressed by the grammar. Furthermore, the relative pronoun *waar* comes with two linked combinatorial types: a function taking embedded sentences to its right and nouns to is left,

and a primitive type *rnp*, for the relatively fluid *r*-pronouns (Cremers 2004). The latter is to be absorbed by *in*, as an argument in the relative clause, which then is absorbed by the combinatorial function connecting that clause to its antecedent; this reflects the double loyalty ('upward' and 'downward') of relative pronouns.
As a particularity of Dutch word order, the linearizing modality for the right-hand, predicative argument of *in* inserts the stranded preposition to the left of that argument, in its left local field. The exact position of insertion is relatively free.

The order of unification is determined on the basis of the modal categorial grammar. After unification, the logical form has to be determined. Colour scheme (58) holds the typed function-argument arrangement of the lambda terms contributed by the different signs. It is followed by the full post-derivational, pre-compositional arrangement (59). Complex functors are in square brackets, and the colours match those in the global term (58) . Finally, (60) represents the fully converted and composed proposition in standard first order predicate-logical format, where the colours of the individual variables, operators and constants reveal their origin in the unconverted terms above.

(58) *post-derivational composition arrangement : complex signs*

```
een  ( {waar}     auto
       ( iedere   vrouw
                  ( in waar zingt )))
        ( deze  man  ziet )
```

(59) *post-derivational composition arrangement: lambda terms*

```
[λR.λS.∃v.R(v) & S(v)]. ( [λy.λU.λI. U(y) & I)]. ( [λP.λQ.∀s.P(s) → Q(s)].
(λv.∃z.st(z, woman) & th(z,v))
( [λH.λI.∃u. rel(u, at_place) & loc(u,H) & th(u,I)].
(λx.∃t. ev(t,sing) & ag(t, x))(y)))(λt.∃s.st(s,car) & th(s,t)))
( [λP.λQ.ιx.P(x) & Q(x)].
 (λp.λy.∃w. ev(w,see) & exp(w,p) & th(w,y)) (λu.∃r.st(r,man) & th(r,u)))
```

(60) *fully composed proposition, standard logical form*

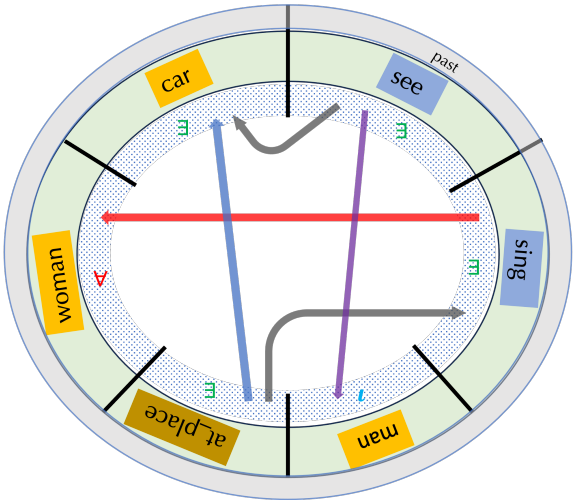
```
∃v.∃s.st(s,car) & th(s,v) & ∀s.∃z.st(z, woman) & th(z,s) →
  ∃u.rel(u,at_place) & loc(u,v) & th(u, ∃t.ev(t,sing) & ag(t,s) &
    ιx.∃r.st(r,man) & th(r,x) & ∃w. ev(w,see) & exp(w,x) & th(w,v)
```

From the colour tracks in these representations one can read that the logical structure of the final, computed, proposition (60) is not projected from the derivational arrangement (58). Every entailment of the proposition is a sub-proposition containing at least two occurrences of at least one variable: every entailment contains at least two linked variables linked. But every sub-proposition contains (coloured) traces of several lexical constructs. As argued before, there is no sign of a transparent relation between constituents and entailments, post-derivationally.

10. The generation procedure enhances semantic evaluation

In order to appreciate how flexible and creative the generation procedure must be to arrive at a meaningful linearized well-formed output, compare a simple donut-shaped visualization of the network addressed in (55)(a) above with the computed tree-representation of (56)(a):

(61) *network representation of (52): donut semantics*



see: existentially quantified event, with time *past*
sing: existentially quantified event, *theme_of* **at_place**
car: existentialy quantified state, *theme_of* **see**, *location of* **at_place**
woman: universally quantified state, *agent_of* sing
man: definitely quantified state, *experiencer_of* **see**
at_place: existentially quantified relation

(62) *generated syntactic tree of one of sentences (56)*

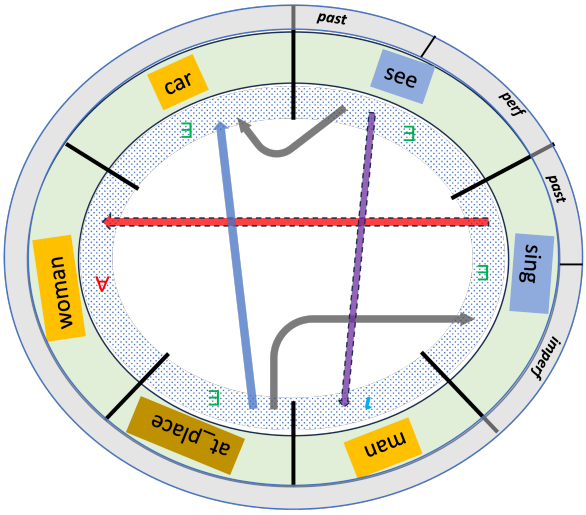
```
1-11+s\wh~[]/1~[]
[enkele,auto's,waar,in,alle,vrouwen,zongen,had,de,man,gezien]
   1-7+np\0~[]/1~[]  [enkele,auto's,waar,in,alle,vrouwen,zongen]
      1-1+np\0~[]/0~[n^0]  [enkele]
      2-7+n\1~[]/1~[]  [auto's,waar,in,alle,vrouwen,zongen]
         2-2+n\0~[]/0~[]  [auto's]
         3-7+n\0~[n^0]/1~[]  [waar,in,alle,vrouwen,zongen]
            3-7+n\0~[n^0]/0~[s_vn^1]*s_vn\1~[]/1~[]
[waar,in,alle,vrouwen,zongen]
               3-4+n\0~[n^0]/0~[s_vn^1]*s_vn\1~[]/0~[s_vn^1]  [waar,in]
                  3-3+n\0~[n^0]/0~[s_vn^1]*rnp\0~[]/0~[]  [waar]
                  4-4+s_vn\0~[rnp^0]/0~[s_vn^1]  [in]
               5-7+s_vn\1~[]/0~[]  [alle,vrouwen,zongen]
                  5-6+np\0~[]/1~[]  [alle,vrouwen]
                     5-5+np\0~[]/0~[n^0]  [alle]
                     6-6+n\0~[]/0~[]  [vrouwen]
                  7-7+s_vn\0~[np^0]/0~[]  [zongen]
   8-11+s\0~[np^wh]/1~[]  [had,de,man,gezien]
      8-10+s\0~[]/1~[vp_d^6]  [had,de,man]
         8-8+s\0~[]/0~[np^0,vp_d^6]  [had]
         9-10+np\0~[]/1~[]  [de,man]
            9-9+np\0~[]/0~[n^0]  [de]
            10-10+n\0~[]/0~[]  [man]
      11-11+vp_d\0~[np^wh]/0~[]  [gezien]
```

Although the representations are unanchored, some clear structural differences between (61) and (62) can be established. First of all, the concepts in the input graph are related to each other, but not ordered intrinsically. The words and phrases in the output string, on the other hand, entertain a strict and total order, in two dimensions. As a matter of fact, that ordering *is* the main result of a successful generation procedure, *salve* meaning. The acquired order is not derived from the input constraint, but computed by an operational grammar of Dutch, fed by an extensive constructicon. Secondly, the wellformedness and interpretability of the generated string partially depends on obligatory morphological and lexical specifications. We cannot, however, expect specifications like number, aspect, and tense to be systematically introduced by the input

constraint, for that constraint can have a non-linguistic origin: a database, an algorithm, etc. These attributes may be consequential for the resulting proposition. A Dutch noun is inevitably marked for number, and number may have semantical impact; a Dutch finite verb is inevitably marked for tense and aspect - essential aspects of propositional meaning. Not every concept in the input constraint, however, is specified that rigorously. Not even the word class realizing a concept is identified here - word classes and concepts are mutually independent. Thirdly, for each input constraint (52)/(61), algorithms like (55) produce a family of sentences which may not be homogeneous syntactically, but is homogeneous semantically, by construction.

As argued in sections 3 and 4, the generating algorithms do not define any function from propositions to sentences and they do not establish any functional relation between entailments and constituents. Nevertheless, although the generation process is hard to retrace, the generated proposition can be evaluated against the input constraint. To see how, compare the donut representation of the input (61) with that of its generated counterpart below:

(63) *network representation of logical form computed for (62): donut semantics*



see: existentially quantified event, in the scope of **man**, with time *past* and aspect *perfective*
sing: existentially quantified event, *theme_of* **at_place,** in the scope of **woman,** with time *past* and aspect *imperfective*
car: existentialy quantified state, *theme_of* **see,** *location of* **at_place**
woman: universally quantified state, *agent_of* sing
man: definitely quantified state, *experiencer_of* **see**
at_place: existentially quantified relation

Inspecting the two schemes - the input constraint and the generated meaning - one can notice that (61) is slightly less specific than (63). The latter's outer ring contains more semantic constants, and its inner plaiting shows a few more details. Both differences indicate that the input is underspecified with respect to a full-fledged Dutch sentence, where finite verbs inevitably come with tense and aspect and where quantificational dependency can affect reference. The remaining overlap between the two donuts, however, represents semantic equivalence. All connections between concepts are entailed, as none is obscured by being in the domain of an intensional operator; the entailment comprises, of course, tense specifications. Here are, in plain English (and to be generated in plain Dutch by Delilah), some of the semantic consequences of both logical forms, with tense and aspect according to donut (63) and assuming, with Seuren (2006), existential impact of universal quantification in natural language semantics.

(64) Something happened in a car
A car was seen

A woman sang
The man saw something
The man saw a car in which something happened
A woman was singing somewhere
Somewhere somebody sang
All young women singing
There was some singing in a car ...

The equivalence in the domain of semantic consequences establishes the possibility of generating a complex Dutch sentence (53) from an abstract logical form (52) by a meaning-driven but non-deterministic algorithm as in (55).
Thus, the algorithm generating relative clauses from logic input looks like a typical solution to a complex problem: it is hard to build, but relatively easy to check.

References

Abzianidze, L., L. Billynina and D. Paperno (2023). *Semantics and deep learning*. Lingbuzz/007736.

Broekhuis, H. et al. (2012-2019). *Syntax of Dutch. Volumes 1 – 6.* Amsterdam University Press.

Bunt, H. (1985). Semantic underspecification: which technique for which purpose? *In:* H. Bunt and R. Muskens (Eds). *Computing Meaning. Volume 3.* Springer, p. 55-85.

Copestake, A., D. Flickinger, I. Sag and C. Pollard (1999). *Minimal recursion semantics: An introduction.* www-csli.stanford.edu/~aac/newmrs.ps

Cremers, C. (2004). Modal merge and minimal move for dislocation and verb clustering. *Research on Language and Computation 2*, p. 87-103.

Cremers, C., M. Hijzelendoorn and H. Reckman (2014). *An Inquiry into the Computation of Meaning and the Incompleteness of Grammar.* Leiden University Press.

De Vries, M. (2002*). The syntax of relativization*. University of Amsterdam, PhD dissertation

De Vries, M. (2006). Possessive Relatives and (Heavy) Pied-Piping. *Journal of Comparative German Linguistics 9*, p 1-52.

Desclés, J-P. (2004). Combinatory Logic, Language and Cognitive Representations. *In:* P. Weingärtner (Ed.), *Alternative Logics. Do Sciences Need Them.* p. 115-148. Springer

Dietterich, T. (2023). *What's wrong with LLM and what we should be building instead.* Valgrai Scientific Council Forum. https://youtu.be/cEyHsMzbZBs.

Haruta, I., K. Mineshima, and D. Bekki (2022). Implementing Natural Language Inference for Comparatives. *Journal of Language Modelling 10:1*, p. 139-1912

Haeseryn, W. et al. (Eds) (2012). *Algemene Nederlandse Spraakkunst*. E-ANS Version 1.3. http://ans.ruhosting.nl/e-ans.

Hendriks, H. (1993). *Studied Flexibility*. PhD dissertation, University of Amsterdam.

Honcoop, M. (1998). *Dynamic excursions on weak islands*. PhD dissertation, Leiden University

Kayne, R. (1994). *The Antisymmetry of Syntax*. MIT Press.

Kluender, R. (1998). On the distinction between strong and weak islands: A processing perspective. *In:* P. Culicover & L. McNally (Eds.), *The limits of syntax*. Syntax and semantics (Vol. 29, pp. 241–279). San Diego: Academic Press.

Kracht, M. (2003). *The Mathematics of Language*. Mouton de Gruyter.

Montague, R. (1972). The Proper Treatment of Quantification in Ordinary English. In: D. Davidson and G. Harman (Eds). *Semantics of Natural Languages*. Reidel, 1972.

Moortgat, M. (1988). *Categorial investigations: Logical and linguistic aspects of the Lambek Calculus*. Foris.

Moortgat, M. (1997). Categorial type logics. *In:* J. van Benthem & A. ter Meulen (Eds.), *Handbook of logic and language,* p. 93–177. Elsevier.

Morrill, G. (2017). Grammar logicised: relativisation. *Linguistics and Philosophy 40:2.* p. 119-163

Morrill, G. and O. Valentin (2010). Displacement Calculus. *Linguistic Analysis 36.* p. 167-192. https://arxiv.org/pdf/1004.4181.pdf.

Reckman, H. (2009). *Flat but not shallow*. PhD dissertation, Leiden University.

Ross, J.R. (1967). *Constraints on variables in syntax.* PhD dissertation, MIT.

Seuren, P.A.M. (2006). The natural logic of language and cognition. *Pragmatics 16:1*, p. 103-138.

Shakouri, D. (*in prep*). *The Mother Daughter Machine*. PhD dissertation, Leiden University.

Šimek, R. (2023). *From interrogatives to relatives: A comprehensive account of wh-constructions.* Lingbuzz/007142.

Van Benthem, J.F.A.K. *Language in Action. Categories, Lambdas and Dynamic Logic*. North Holland Elsevier, 1991.

Westerståhl, D. (2019). 'Generalized Quantifiers'. *In:* E.N. Zalta (ed.), *The Stanford Encyclopedia of Philosophy*. https://plato.stanford.edu/ archives/win2019/ entries/generalized-quantifiers/.