Laboratory Set-Up of Database Semantics

Roland Hausser

Universität Erlangen-Nürnberg (em.)

Abstract

The analysis of natural language in today's linguistics, analytic philosophy, and computer science is either (i) agent-based data-driven or (ii) sign-based substitution-driven. A sign-based ontology has the apparent advantage that it obviates any need for an interface component with sensors for vision and audition, and actuators for manipulation and vocalization. In an age when artificial vision, audition, manipulation, locomotion, and computers did not exist, this was a necessity. The question is how to adjust today's language research to the age of computers and artificial intelligence by changing to an agent-based data-driven[1] ontology?

keywords: time-linearity, type transparency, computational complexity, successful communication, content, declarative specification, operational implementation, fragment of a natural language, upscaling cycles

1 Early Times

The absence of computers did not stop the grammarians of the ancient and recent past from contributing essential notions representing important insights, such as accusative, active, adjective, agglutination, agreement, allomorph, analytic, aorist, clause, comparation, conjunction, dative, determiner, domain, event, function, future, genitive, imperfect, inflection, isolating, medium, modifier, morpheme, morphology, nominative, noun, object, passive, perfect, phrase, pragmatics, predicate, pronoun, proposition, range, relation, semantics, subclause, subject, syntactic mood, syntax, synthetic, tense, and verbal mood. Without these notions most of modern linguistics would be unthinkable.

A recent attempt to bring language science into modern times was Chomsky's Generative Grammar for characterizing the innate universal structure of natural language (nativism). Rewrite rules generate constituent structures from the S node (for sentence or start) by repeated substitution, resulting in phrase structure trees which are defined in terms of "dominance" and "precedence." By adding a transformation component to a context-free phrase structure base, the computational complexity of Transformational Grammar increased from polynomial to undecidable (Peters and Ritchie 1973).

Chomsky emphasized repeatedly that Generative Grammar was "not intended" for modeling communication: "To avoid what has been a continuing misunderstanding, it is perhaps worth while to reiterate that a generative grammar is not a model for a speaker or a hearer." (Chomsky 1965, p. 9). Yet, as shown by

[1] A well-known example of an early data-driven programming language is AWK (Aho, Kernighan, and Weinberger 1977, 1988).

the analogy with anatomy, it is unlikely that a supposedly innate universal model of natural language would be without a speak mode, a hear mode, and a transfer channel, especially in language acquisition.

In consequence, many linguists moved (or returned) from nativism to the study of large data and statistics (Church&Mercer 1993). However, statistics alone is not sufficient for building a talking robot. In analogy, if the Martians came to Earth and modeled cars statistically, the cars would never run. Instead, the Martians would have to chose a car in good running condition, take it apart, study the parts and the functional flow, and reconstruct the mechanisms of the motor, the wheels, the breaks, the transmission, etc., until the reassembled vehicle would run again as before.

2 Study of the Language Signs

A truly classic pioneer of modern linguistics was the Swiss linguist de Saussure (1857–1913), who formulated the most important properties of the natural language signs as the *premiere principe*, l'arbitraire de signe,[2] and the *second principe*, caractère linéaire du signifiant.[3] These principles are as valid today as when they were first proposed.

Regarding the second principle, de Saussure continues in good humor:

> Ce principe est évident, mais il semble qu'on ait toujours négligé de l'énoncer, sans doute parce qu'on l'a trouvé trop simple; cependent il est fondamental et les conséquences en sont incalculables; son importance est égale à celle de la première loi. Tout le mécanisme de la langue en dépend.[4]

Ignoring time-linearity is one of those aberrations which are so frequent in the history of science and which often take several centuries to be rectified.

The first attempt at combining time-linearity with detailed grammatical analysis and efficient computation was NEWCAT'86. Still sign-based, i.e., without a distinction between the speak and the hear mode, it presents time-linear derivations for 221 examples of German and 114 examples of English, programmed in Lisp and published with the source code.

2.1 NEWCAT parse of Fido dug the bone up. (CoL 3.3.4)

```
* (z Fido dug the bone up \.)

    Linear Analysis:

    *START
    1
       (N-H) FIDO
       (N A UP V) DUG
    *NOM+FVERB
    2
```

```
   (A UP V) FIDO DUG
   (GQ) THE
*FVERB+MAIN
3
   (GQ UP V) FIDO DUG THE
   (S-H) BONE
*DET+NOUN
4
   (UP V) FIDO DUG THE BONE
   (UP NP) UP
*FVERB+MAIN
5
   (V) FIDO DUG THE BONE UP
   (V DECL) .
*CMPLT
6
   (DECL) FIDO DUG THE BONE UP .
```

The grammatical analysis is a formatted *trace* of the computational operations. Each numbered derivation step consists of a sentence start, e.g., `(A UP V) FIDO DUG`, the next word `(GQ) THE`, the rule name `*FVERB+MAIN`, a number (here `3`), and the resulting output `(GQ UP V) FIDO DUG THE`, which redoubles as the input to the next derivation step. As a direct reflection of the computational application of the grammar rules, tracing is the ultimate form of *type transparency* (Berwick and Weinberg 1984). Computational tracing as the exclusive method of grammatical analysis is used in all subsequent work of what became DBS.

Like NEWCAT'86, *Computation of Language* (CoL 1989) is still sign-based, but expands the time-linear NEWCAT approach to computational complexity analysis. For example, the formal language $a^kb^kc^k$ is context-sensitive in the PSG hierarchy and parses in exponential time, but is a C1 language in the LAG hierarchy and parses in linear time[5] (CoL 6.4.3, FoCL 10.2.2):

2.2 LA Grammar for $a^kb^kc^k$

LX $=_{def}$ {[a (a)], [b (b)], [c (c)]}
ST$_s$ $=_{def}$ {[(a) {r$_1$, r$_2$}]}
r$_1$: (X) (a) $\Rightarrow$ (aX) {r$_1$, r$_2$}

[2] 'First principle: arbitrariness of the sign.' It refers to the fact that different languages may use different surfaces, e.g., fauteuil, sessel, and poltrona, for the same kind of thing, here 'easy chair,' based on different *conventions* within the different language communities.

[3] 'Second principle: linear character of the sign.' It refers to the fact that language signs follow each other in a certain grammatical order. Changing the order results in a change of meaning or in ungrammaticality.

[4] 'This principle is obvious, but it seems that stating it explicitly has always been neglected, doubtlessly because it is considered too simple. It is, however, a fundamental principle and its consequences are incalculable. Its importance equals that of the first law. All the mechanisms of language depend on it.' De Saussure [1916](1972), p. 103.

[5] The term 'time-linear' refers to a grammatical derivation order while the term 'linear time' refers to a computational complexity degree.

r_2: (aX) (b) $\Rightarrow$ (Xb) $\{r_2, r_3\}$
r_3: (bX) (c) $\Rightarrow$ (X) $\{r_3\}$
$ST_F =_{def} \{[\varepsilon\ rp_3]\}$.

A lexical entry like [a(a)] in the set LX consists of a surface, here a, and a category, here (a). The set ST_s happens to contain only one start state, namely $\{[(a)\ \{r_1, r_2\}]\}$; this means that the first input must have the category (a), i.e., it must have the surface a, and that the rules applying to the first and the second input are limited by the rule package to r_1 and r_2. Rule r_1 adds an (a), r_2 subtracts an (a) and adds a (b), while r_3 subtracts a (b) from the category.

The rule package of r_1 is $\{r_1, r_2\}$, i.e., after r_1 has applied, r_1 and r_2 are tried on the next word, and accordingly for the rules packages of r_2 and r_3. The set ST_F contains only one final state, namely $\{[\varepsilon\ rp_3]\}$, i.e., the category must be empty (ε) and the currently activated rule package must be that of r_3.

Compared to the context-sensitive PSG (FoCL 8.3.7), the LAG is exceedingly plain. Furthermore, the LA Grammars for context-free a^kb^k (CoL 10.2.3) and for context-sensitive $a^kb^kc^k$ are in the same language class of DBS and the number of coefficients, as in $a^kb^kc^kd^k$, $a^kb^kc^kd^ke^k$, etc., has no effect on the linear complexity of their LA Grammars. Like the natural language analysis 2.1, an $a^kb^kc^k$ expression is analyzed as a formatted trace of the parse:

2.3 SAMPLE DERIVATION OF **aaabbbccc** WITH ACTIVE RULE COUNTER

```
* (z a a a b b b c c c)
;  1: Applying rules (RULE-1 RULE-2)
;  2: Applying rules (RULE-1 RULE-2)
;  3: Applying rules (RULE-1 RULE-2)
;  4: Applying rules (RULE-2 RULE-3)
;  5: Applying rules (RULE-2 RULE-3)
;  6: Applying rules (RULE-2 RULE-3)
;  7: Applying rules (RULE-3)
;  8: Applying rules (RULE-3)
; Number of rule applications: 14.

    *START-0
    1
       A (A)
       A (A)
    *RULE-1
    2
       A A (A A)
       A (A)
    *RULE-1
    3
       A A A (A A A)
       B (B)
    *RULE-2
    4
       A A A B (A A B)
       B (B)
```

```
*RULE-2
5
   A A A B B (A B B)
   B (B)
*RULE-2
6
   A A A B B B (B B B)
   C (C)
*RULE-3
7
   A A A B B B C (B B)
   C (C)
*RULE-3
8
   A A A B B B C C (B)
   C (C)
*RULE-3
9
   A A A B B B C C C (NIL)
```

Expressions which are not in the language, e.g., aaabbc, are analyzed to the point of the ungrammatical continuation, here aaabb+c, and rejected as such. While PSG derivations are substitution-driven by always starting with the same S symbol followed by random applications of rewrite rules (computing possible substitutions), LAG derivations are data-driven by processing the input surfaces one after the other (computing possible continuations). The LAG hierarchy is the first, and so far the only, complexity hierarchy which is orthogonal to the PSG hierarchy TCS'92):.

3 Using Successful Communication for the Laboratory Set-Up

In face to face dialogue, the hearer's interpretation begins with the speaker's first word. From there, the hearer follows the sequence of incoming surfaces incrementally, with the speaker at least one word ahead. In indirect communication based on writing or recorded message, there is no limit on the speaker's lead.

This could be taken as a reason for starting the scientific analysis of natural language communication with the speak mode. However, there is a more important aspect to the distinction between the two modes, namely the difference in the respective input and output: the speak mode takes a cognitive content as input and produces an external surface as output, while the hear mode takes an external surface as input and produces a cognitive content as output.[6]

For the scientific investigation of natural language communication, the hear mode has the practical advantage of concretely given external input, i.e., the raw data of the language-dependent surfaces. They have no meaning or grammatical properties

[6] It is not possible to use the same software for runing "upward" in the speak mode (from content to surface) and "downward" for the hear mode (from surface to content) because input and output are of different kinds. DBS inferencing, in contrast, allows *inductive* (forward) and *abductive* (backward) use because both directions take the same kind of input and produce the same kind of output.

whatsoever (no reification in DBS), but they are measurable by natural science and interpretable by automatic speech recognition (asr) or optical character recognition (ocr). The input to the speak mode, in contrast, is agent-internal cognitive content which can only be inferred.

Therefore DBS starts the computational reconstruction of natural language communication with the hear mode's first step, namely automatic word form recognition of the raw surface input by means of computational pattern matching. The output of the hear mode is an agent-internal, purely cognitive structure: it derives the literal meaning$_1$ (PoP-1, FoCL 4.3.3) of the input surface as an agent-internal content.

In order for communication to be successful, the following condition must be fulfilled:

3.1 MINIMAL CONDITION FOR COMMUNICATION TO BE SUCCESSFUL

> The meaning$_1$ content used as input by the speak mode and the meaning$_1$ content derived as output in the hear mode must be the same.

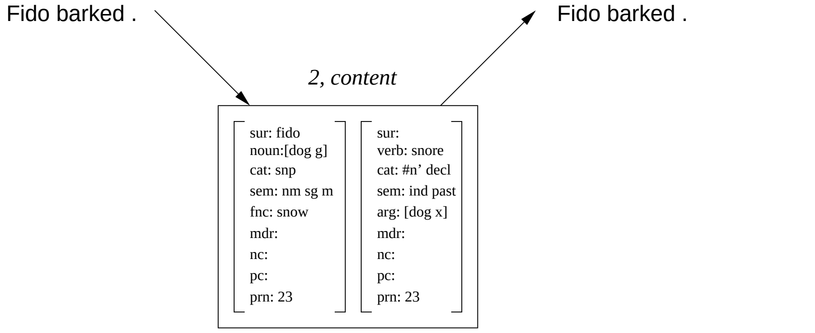
This criterion is the pivot of the DBS *laboratory set-up*:

3.2 DEFINITION OF THE DBS LABORATORY SET-UP

- The content automatically derived as output in the hear mode is reused systematically as the input to the automatic speak mode derivation.
- The content of a given example surface is correct if, and only if, the hear mode's input surface equals the speak mode's output surface.

The laboratory set-up provides a fully automatic, clear and simple method of verification. It requires that (i) the grammatical details of the speak mode suffice for the associated hear mode to automatically derive the speaker's content and (ii) that the grammatical details of hear mode content suffice for the associated speak mode derivation to automatically produce the hearer's surface.

3.3 LABORATORY SET-UP: FROM HEAR MODE TO SPEAK MODE



The DBS laboratory set-up produces semantically well-motivated content as input for the speak mode. It is the first and so far the only method of linguistic analysis by which the hear and the speak mode benefit each other.

The laboratory set-up is based on switching off inferencing, temporarily limiting the think-speak mode to traversing meaning$_1$ content and producing literal surface representations in the natural language of choice ('narrative speak mode'). This resembles a sign-based approach in that it excludes pragmatics, but differs in that it has an explicit notion of content and includes the reconstruction of the speak and the hear mode. When the direction from speaker to hearer outside the laboratory set-up is re-established and inferencing for non-literal use is switched back on, the speak mode (deductive) may realize inference content as language-dependent surfaces and the hear mode (abductive) may interpret the surfaces as inference content – data-driven, without any need for additional software.

4 From Operational Implementation to Declarative Specification

NEWCAT and CoL take a complete expression as input (holistic loading) and process it symbol by symbol in left-associative[7] order. FoCL ET SEQ., in contrast, supply (i) individual 'next words' separately by automatic word form recognition (incremental loading), (ii) intertwine each hear mode operation application with a next word look-up, and (iii) define the operations to integrate the 'next word' into the current 'sentence start.' This came with a change from the ordered triple analysis of a word form in NEWCAT to the proplet format as a nonrecursive feature structure with ordered attributes, serving as the computational data structure.

For example, the ordered triple analysis of dug in 2.1 was changed into the following proplet:

4.1 Transition from an ordered triple to a lexical proplet

ordered triple format	*proplet format of DBS*
[dug (N A up V) dig]	sur: dug verb: dig cat: N′ A′ up′ V sem: *up* ind past arg: mdr: nc: pc: prn:

The two formats differ as follows.

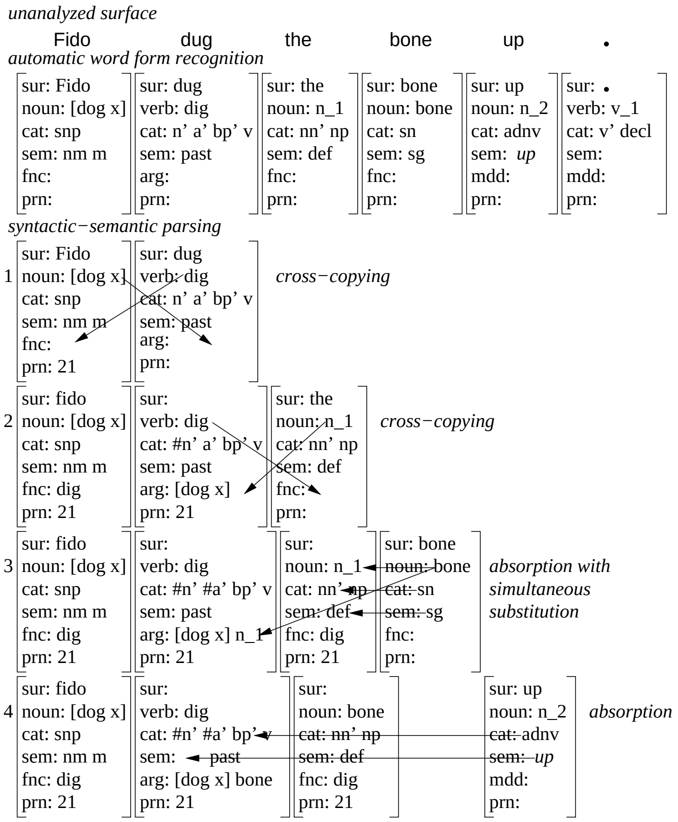
[7] Aho and Ullman 1977, p. 47.

4.2 COMPARING THE NEWCAT-COL APPROACH WITH DBS

1. The ordered triple format does not distinguish between valency slots and valency fillers, whereas DBS proplets mark valency positions with ′, e.g., N′.
2. In the ordered triple format, valency positions are canceled by deletion (as in Categorial Grammar), whereas the DBS hear mode cancels valency positions by #-marking, thus preserving the information for the speak mode.
3. Derivations in the ordered triple format prefer ending on empty category and use the complete derivation as the resulting content. A derivation in the proplet format, in contrast, results in a derivation-independent content, defined as a set of proplets. connected by address.
4. The proplet format enables string-search-based storage in and retrieval from a content-addressable database contained in a cognitive agent with an interface component (agent-based ontology).
5. The on-board database interacts with the agent's interface component for the recognition and production of language surfaces, as well as nonlanguage contents.
6. The agent's on-board orientation system provides the STAR for the interpretation of the sign kind 'indexical.'

Compare the following DBS hear mode derivation with the NEWCAT derivation 2.1:

4.3 DBS HEAR MODE DERIVATION OF Fido dug the bone up.



unanalyzed surface

Fido dug the bone up .

automatic word form recognition

sur: Fido	sur: dug	sur: the	sur: bone	sur: up	sur: .
noun: [dog x]	verb: dig	noun: n_1	noun: bone	noun: n_2	verb: v_1
cat: snp	cat: n' a' bp' v	cat: nn' np	cat: sn	cat: adnv	cat: v' decl
sem: nm m	sem: past	sem: def	sem: sg	sem: *up*	sem:
fnc:	arg:	fnc:	fnc:	mdd:	mdd:
prn:	prn:	prn:	prn:	prn:	prn:

syntactic−semantic parsing

1	sur: Fido noun: [dog x] cat: snp sem: nm m fnc: prn: 21	sur: dug verb: dig cat: n' a' bp' v sem: past arg: prn:	*cross−copying*

2	sur: fido noun: [dog x] cat: snp sem: nm m fnc: dig prn: 21	sur: verb: dig cat: #n' a' bp' v sem: past arg: [dog x] prn: 21	sur: the noun: n_1 cat: nn' np sem: def fnc: prn:	*cross−copying*

3	sur: fido noun: [dog x] cat: snp sem: nm m fnc: dig prn: 21	sur: verb: dig cat: #n' #a' bp' v sem: past arg: [dog x] n_1 prn: 21	sur: noun: n_1 cat: nn' ~~np~~ sem: def fnc: dig prn: 21	sur: bone ~~noun:~~ bone ~~cat:~~ sn ~~sem:~~ sg fnc: prn:	*absorption with simultaneous substitution*

4	sur: fido noun: [dog x] cat: snp sem: nm m fnc: dig prn: 21	sur: verb: dig cat: #n' #a' bp' ~~v~~ sem: ~~past~~ arg: [dog x] bone prn: 21	sur: noun: bone ~~cat: nn' np~~ ~~sem: def~~ fnc: dig prn: 21	sur: up noun: n_2 ~~cat:~~ adnv ~~sem:~~ *up* mdd: prn:	*absorption*

	sur: fido	sur:	sur:	sur: •	
5	noun: [dog x]	verb: dig	noun: bone	verb: v_1	*absorption*
	cat: snp	cat: #n' #a' #bp' v	~~cat: nn' np~~	~~cat:~~ v' decl	
	sem: nm m	sem: *up* past	sem: def	sem:	
	fnc: dig	arg: [dog x] bone	fnc: dig	fnc:	
	prn: 21	prn: 21	prn: 21	prn:	

result

sur: fido	sur:	sur:
noun: [dog x]	verb: dig	noun: bone
cat: snp	cat: #n' #a' #bp' decl	cat: nn' np
sem: nm m	sem: *up* past	sem: def
fnc: dig	arg: [dog x] bone	fnc: dig
prn: 21	prn: 21	prn: 21

The bare preposition up is absorbed in line 4. It #-cancels the valency position bp′ in the third cat slot of *dig* and writes its sem value up into the initial sem slot of the verb, making it available for the speak mode. Like the NEWCAT-style derivation 2.1, the derivation exactly mirrors the sequence of operation applications[8] (tracing).

The transition from NEWCAT and CoL to FoCL, NLC, CLaTR, TExer, and CC may be summarized as follows. Instead of taking the whole surface of a sentence or text as input (holistic loading), there is incremental next word lookup.[9] Instead of deleting valency positions in the verb's cat slot, they are preserved for the speak mode by #-canceling. Instead of using the derivation, e.g., 2.1, as the grammatical analysis, there results a *content* (e.g., result in 4.3) which is not dependent on the hear mode derivation (essential for using language content in nonlanguage cognition). Instead of longer and longer intermediate states, there are larger and larger sets of proplets connected by address (order-free), which is essential for the storage and retrieval in the on-board content-addressable database.

5 Formal Fragments of Natural Language

There are formal language analyses in the tradition of symbolic logic (Montague 1974) and computational complexity (FoCL) theory which use explicit rule systems to analyze-generate limited 'fragments' of natural or formal languages like a^kb^k, $a^kb^kc^k$, $a^kb^kc^kd^k$, etc. (2.3). A fragment is precisely defined as a set of examples for the analysis of specific, natural or artificial, grammatical structures. The language data in a fragment are limited, but their analysis is required to be explicit.

The use of software in the computational analysis of fragments opens a new dichotomy as compared to precomputational linguistics, namely between (i) the *declarative specification* and (ii) the *operational implementation*. The declarative specification represents the necessary properties of the software and must be simultaneously suitable (a) for reading by humans and (b) for a straightforward translation into a general purpose programming language of choice. The operational implementation, in contrast, has additional accidental properties, namely those which distinguish equivalent implementations in different programming languages.

[8] For the sequence of explicit hear and speak mode operation applications see TExer 4.3.

[9] For the time-linear transition from one sentence to the next see TExer 2.1.

After working on implementing a fragment of natural language in a general purpose programming language of choice, there naturally arises a scientific interest in leaving the accidental properties behind and work out the necessary ones in the systematic format of a declarative specification.[10] Conversely, after working on a declarative specification for a fragment of a natural language, there naturally arises a scientific interest in verifying the fragment in the form of an operational implementation.[11]

6 Incremental Upscaling Cycles

Once a current fragment has been supplied with a declarative specification for the speak and the hear mode, and been verified by an operational implementation, the next upscaling cycle is started by extending the current fragment with a limited number of additional examples which have new and interesting syntactic and semantic properties. For this kind of work, a standard computer of today is sufficient. It provides the keyboard for input and the screen for output, which allows to implement the hear mode, the content-addressable database with its now front mechanism, and the think-speak mode navigation with and without surface realization, using placeholders for concepts.

Conclusion

The computational reconstruction of natural language communication in DBS combines insights and methods from the humanities, the engineering sciences, and the natural sciences. The approach is incremental by starting with a fragment of limited data coverage, but functionally as complete as possible. Data coverage is represented by sets of concrete language examples which represent grammatical constructions in the natural language of choice. After completing fragment n, it is extended into fragment n+1 by adding new linguistic constructions.

Functional completeness includes nonlanguage recognition and action, the language hear and speak modes, an on-board memory for the storage and retrieval of content, and reasoning by inference. Contents are built from concepts connected by the semantic relations of structure, i.e. functor-argument and coordination.

Currently, computational verification is limited to the key board and the screen of today's standard computers for input and output. As a temporary substitute for recognition and action by sensors and acuators, concepts are represented by place holder values. The *laboratory setup* uses the output content of the hear mode as input content to the speak mode for testing the cycle of natural language communication in a fragment.

[10] The first operational implementations were published as NEWCAT and CoL. Work on the declarative specification began with FoCL, continued with NLC and CLaTR, and culminated for now in TExer and CC.

[11] This kind of work is not only of practical use, but may also serve as an inexhaustible source of theoretically demanding thesis topics at all degree levels.

Bibliography

Aho, A.V., B.W. Kernighan, and P.J. Weinberger (1977, 1988) *The AWK Programming Language*, Addison-Wesley

Aho, A.V. and J.D. Ullman (1977) *Principles of Compiler Design*, Reading, Mass.: Addison-Wesley

Berwick, R.C., and A.S. Weinberg (1984) *The Grammatical Basis of Linguistic Performance: Language Use and Acquisition*, Cambridge, Mass.: MIT Press

Carbonell, J.G. and R. Joseph (1986) "FrameKit$^+$: a knowledge representation system," Carnegie Mellon University, Department of Computer Science

CC = Hausser, R. (2019) *Computational Cognition: Integrated DBS Software Design for Data-Driven Cognitive Processing*, pp. i–xii, 1–237, `lagrammar.net`

Chomsky, N. (1965) *Aspects of the Theory of Syntax*, Cambridge, Mass.: MIT Press

Church, K.W., & R.L. Mercer, (1993) "Introduction to the Special Issue on Computational Linguistics Using Large Corpora," ACL, Vol. 19.1:1-24

FoCL = Hausser, R. ([1999, 2001] 2014) *Foundations of Computational Linguistics*, Springer

CoL = Hausser, R. (1989) *Computation of Language*, Springer

Montague, R. (1974) *Formal Philosophy*, Collected Papers, R. Thomason (ed.), New Haven: Yale University Press

Peters, S., and R. Ritchie (1973) "On the Generative Power of Transformational Grammar," *Information and Control*, Vol. 18:483–501

NEWCAT = Hausser, R. (1986) *NEWCAT: Parsing Natural Language Using Left-Associative Grammar*, LNCS 231, Springer

Saussure, F. de [1916](1972) *Cours de linguistique générale*, Édition critique préparée par Tullio de Mauro, Paris: Éditions Payot

TCS = Hausser R. (1992) "Complexity in Left-Associative Grammar," *Theoretical Computer Science*, Vol. 106.2:283-308 Elsevier

Tesnière, L. (1959) *Éléments de syntaxe structurale*, Paris: Editions Klincksieck

TExer = Hausser, R. (2020) *Twentyfour Exercises in Linguistic Analysis, DBS software design for the Hear and the Speak mode of a Talking Robot*, `lagrammar.net`