Intro to Onto

Roland Hausser

Universität Erlangen-Nürnberg (em.)

Abstract

For long-term incremental upscaling to be successful, the computational reconstruction of a complex natural mechanism must be *input-output equivalent* with the prototype, i.e., the reconstruction must take the same input and produce the same output in the same processing order as the original. Accordingly, the modeling of natural language communication in Database Semantics (AIJ'01) uses a time-linear derivation order for the speaker's output and the hearer's input. The language-dependent surfaces serving as the vehicle of content transfer from speaker to hearer are raw data, e.g., sound waves or pixels, without meaning or any grammatical properties whatsoever, but measurable by natural science.[1]

1 Ontology

The term *ontology* may be transliterated as 'account of what there is.' The ontology of a field of science comprises the basic elements and relations assumed to allow a complete analysis of its phenomena. For example, the Presocratics tried to explain nature based on an ontology of fire, water, air, and earth. Today, the ontology of physics is based on a space-time continuum, protons, electrons, neutrons, quarks, neutrinos, etc.

Similarly in theories of meaning in philosophy. There was a time in which meaning was based on naming; for example, the celestial body rising in the morning and setting in the evening served as the meaning of the word sun. Then meaning became defined in terms of set-theoretic denotations in possible worlds. Which ontology is required for building the computational cognition of a talking robot?

Just as an ontology without subatomic particles is unsuitable for modern physics, an ontology of computational cognition without an agent, without a distinction between an agent-external reality and agent-internal processing, without interfaces for recognition and action, without a distinction between the speak and the hear mode, without an on-board database (memory) with an on-board orientation system (OBOS), and without an algorithm for moment-by-moment monitoring is unsuitable for the task of building a talking robot.

2 Computational Cognition

The ontological requirements for computational cognition were essentially laid down in the year 1945 as the von Neumann machine (vNm): the interface component of DBS corresponds to the vNm input-output device, the DBS on-board

[1]Thanks to Prof. McNeilage, director of the Phonetics Lab in the Linguistics Department at UT Austin during my time as a Ph.D. student. The chance to participate as a test person in phonetic research experiments instilled a permanent appreciation of raw data in language communication.

database corresponds to the vNm memory, and the DBS left-associative operations algorithm corresponds to the vNm arithmetic-logic.

Designing and building the computational cognition of a talking autonomous robot is not only of interest for a wide range of practical applications, but constitutes the ultimate standard for evaluating the many competing theories of natural language in today's linguistics, language philosophy, language psychology, and computer science. It leads from the sign-based substitution-driven ontology of mathematics and symbolic logic to the new (or extended) ontology of agent-based data-driven robotics in general and DBS in particular. It also leads from Generative Grammar (hence GG) and its attempt to discover an innate human language ability to the effective transfer of content from the speaker to the hearer by means of raw data.

Communication is successful if the content encoded by the speaker into raw data equals the content decoded from the raw data by the hearer. DBS constructs content from the three basic content kinds of (i) concept, (ii) indexical, and (iii) name. Each has its characteristic computational mechanism: concepts use computational pattern matching based on the type-token relation, indexicals use pointing at values of the agent's on-board orientation system, and names use an explicit or implicit act of baptism which inserts a named referent as core value into a name proplet (CASM'17).

3 Agent-Based Data-Driven vs. Sign-Based Substitution-Driven

Most analyses of natural language in today's linguistics, philosophy, and computer science rely on a precomputational, sign-based, substitution-driven ontology. Sign-based means: no distinction between the speak and the hear mode. Substitution-driven means: using a start symbol as input and generating output based on possible substitutions by rewrite rules. Thereby different propositions are derived from the same S node and assigned the same denotation, i.e., True or False. However, for a functionally complete, scientific reconstruction of natural language communication, the start button is uniquely unsuitable as input to the speak mode and the truth-values are uniquely unsuitable as output of the hear mode.

In DBS, propositions do not denote but *are content*, and different propositions are different contents. A content is defined as a set of proplets, i.e., order-free (which is essential for storage in and retrieval from a content-addressable on-board database). Proplets are defined as nonrecursive feature structures with ordered attributes (which is essential for efficient pattern matching). The proplets in a content are connected by the classical semantic relations of structure, i.e., functor-argument and coordination, coded by address.

The ontology of DBS is agent-based and data-driven. Agent-based means: design of a cognitive agent with (i) an interface component for converting raw data into cognitive content (recognition) and converting cognitive content into raw data (action), (ii) an on-board, content-addressable database (memory) for the storage and retrieval of content, and (iii) separate treatments of the speak- and the hear-

mode. Data-driven means: (a) mapping a cognitive content as input to the speak mode into a language-dependent surface as output, and (b) mapping a surface as input to the hear mode into a cognitive content as output.

4 Reconciling the Hierarchical and the Linear Aspects of Communication

A content serving as input to the speak mode and as output of the hear is defined as a set of proplets, connected by the semantic relations of structure, coded by address:

4.1 CONTENT OF I saw you.

sur: I	sur: saw	sur: you
noun: **pro1**	verb: **see**	noun: **pro2**
cat: s1	cat: #n #a decl	cat: sp2
sem: sg	sem: past	sem: sg
fnc: **see**	arg: **pro1 pro2**	fnc: **see**
mdr:	mdr:	mdr:
nc:	nc:	nc:
pc:	pc:	pc:
prn: 3	prn: 3	prn: 3

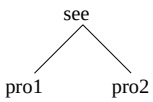
The classical semantic relations of structure are subject/predicate, object\predicate, modifier|modified, and conjunct−conjunct. In 4.1, the semantic relations are subject/predicate and object\predicate, indicated by **bold face** font. In successful communication, the input content of the speak mode and the output content of the hear mode are the same.

5 Introducing Surface Linearity in the Speak Mode

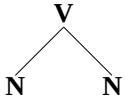
The speak mode converts the hierachy of the input content into the linear structure of the output surface by navigating along the semantic relations of structure:

5.1 GRAPH ANALYSIS UNDERLYING PRODUCTION OF 4.1

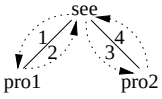
(i) SRG (semantic relations graph)



(ii) signature



(iii) NAG (numbered arcs graph)



(iv) surface realization

1	2	3	4
I	saw	you	.
V/N	N/V	V\N	N\V

The (iv) surface realization consists of three lines, showing (1) the arc numbers, (2) the surfaces realized from the goal proplet, and (3) the traversal operations.

The operations driving the navigation in 5.1 are listed as follows:

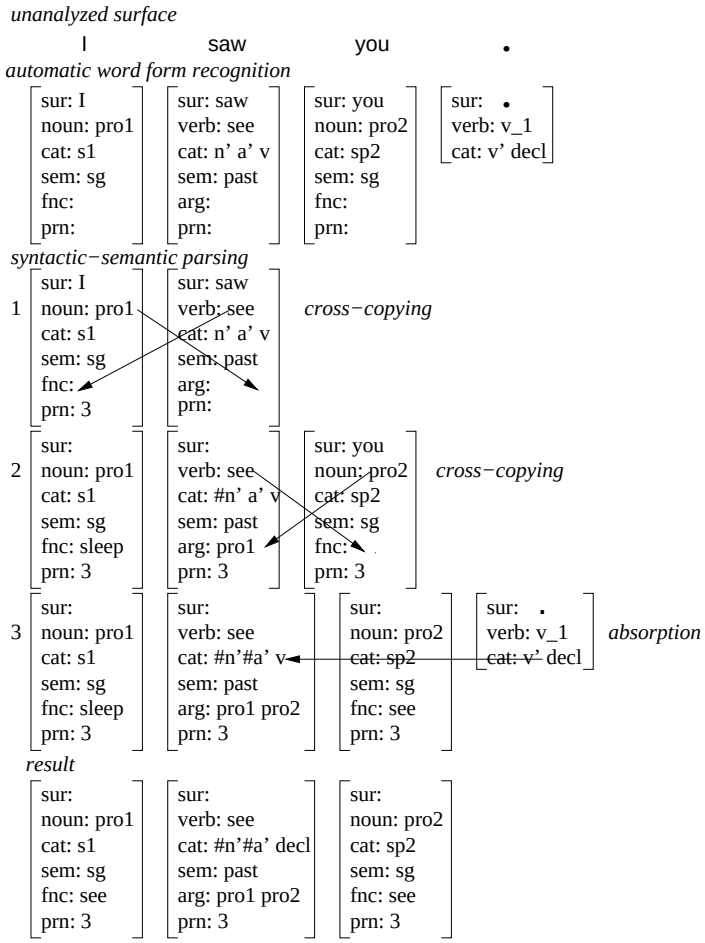
5.2 SEQUENCE OF OPERATION NAMES AND SURFACE REALIZATIONS

arc 1:	V↙N	from *see* to *pro1*	I	(TExer 2.3.8)
arc 2:	N↗V	from *pro1* to *see*	saw	(TExer 2.3.9)
arc 3:	V↘N	from *see* to *pro2*	you	(TExer 2.3.10)
arc 4:	N↖V	from *pro2* to *see*	.	(TExer 2.3.11)

6 Re-conversion of Linear Input Surface into Hierarchical Output Content

The hear mode re-converts the stream of raw input data into the hierarchical structure of 4.1 by incremental lexical lookup and *syntactic-semantic composition*.

6.1 GRAPHICAL HEAR MODE DERIVATION OF THE CONTENT 4.1



The composition is time-linear in that the current next word (lexical proplet) is related semantically to a proplet in the current sentence start (set of connected proplets).

The hear mode operations are of three kinds: (i) cross-copying (connective $\times$), (ii) absorption (connective $\cup$), and (iii) suspension (connective $\sim$). Operations with

the same connective may re-introduce different semantic relations of structure, for example, SBJ×PRD and OBJ×PRD, defined as follows:

6.2 CROSS-COPYING *pro1* AND saw WITH SBJ×PRD (LINE 1)

SBJ×PRD (h1)

pattern level

noun: α	verb: β	noun: α	verb: β
cat: NP	cat: NP′ X v	cat: NP	cat: #NP′ X v
fnc:	arg:	fnc: β	arg: α
prn:K	prn:	prn: K	prn: K

NP ε {snp, pnp, s1, s3, p1, sp2, p3}
NP′ ε {n′, ns3′, n-s3′, ns1′, ns2p′, ns3′, ns13′}
If NP = s1, then NP′ ε {ns1′, ns13′, n′}
If NP = sp2, then NP′ ε {ns2p′, n-s3′, n′}
If NP ε {s3, snp}, then NP′ ε {ns3′, ns13′, ns3′, n′}
If NP = p1, then NP′ ε {ns2p′, n-s3′, n′}
If NP ε {p1,p3, pnp}, then NP′ ε {ns2p′, n-s3′, n′}

content level

sur: I	sur: saw	sur:	sur:
noun: **pro1**	verb: **see**	noun: **pro1**	verb: **see**
cat: s1	cat: n′ a′ v	cat: s1	cat: #n′ a′ v
sem: sg	sem: past	sem: sg	sem: past
fnc:	arg:	fnc: **see**	arg: **pro**1
mdr:	mdr:	mdr:	mdr:
nc:	nc:	nc:	nc:
pc:	pc:	pc:	pc:
prn: 3	prn:	prn: 3	prn: 3

6.3 CROSS-COPYING saw AND *pro2* WITH PRD×OBJ^2 (LINE 2)

PRD×OBJ (h22)

pattern level

verb: β	noun: α	verb: β	noun: α
cat: #X′ N′ Y γ	cat: CN′ N	cat: #X′ #N′ Y γ	cat: CN′ N
arg: Z	fnc:	arg: Z α	fnc: β
prn: K	prn:	prn: K	prn: K

N ε {obq, snp, pnp, s1, sp2, p1, p3}
N′ ε {d′, a′, be′, hv′}, CN′ ε {NIL, nn′, sn′, pn′}, γ ε {v, vi, vimp, inf}
If N ε {s1, sp2, p1, p3}, then N′ = be′; otherwise N′ ε {obq, snp, pnp}

content level

sur:	sur: you	sur:	sur:
verb: **see**	noun: **pro2**	verb: **see**	noun: **pro2**
cat: #n′ a′ v	cat: sp2	cat: #n′ #a′ v	cat: sp2
sem: past	sem:	sem: past	sem:
arg: pro1	fnc:	arg: pro1 **pro2**	fnc: **see**
mdr:	mdr:	mdr:	mdr:
nc:	nc:	nc:	nc:
pc:	pc:	pc:	pc:
prn: 3	prn:	prn: 3	prn: 3

Comparison of the **SBJ×PRD** and the **OBJ×PRD** application illustrates the highly precise coding of grammatical detail, provided by the computational pattern matching and the variable restrictions of DBS. For the complete declarative analysis of

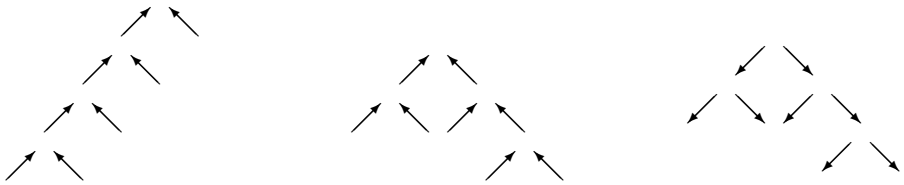
I saw you. in the speak and hear mode see TExer 2.3.

7 Derivation Order

The regular, total-order derivation of time-linear left-associative LAG (as the precursor of DBS) is in contrast to the irregular, partial-order derivations of today's CG (bottom up) and PSG (top down):

7.1 THREE CONCEPTUAL DERIVATION ORDERS (FOCL 10.1.1)

LA Grammar C Grammar PS Grammar



bottom-up left-associative *bottom-up amalgamating* *top-down expanding*

The initial empirical test of using the left-associative derivation order for the syntactic-semantic analysis of a nontrivial set of natural language expressions was programming the time-linear derivations of 221 constructions of German and 114 constructions of English during a research stay at CSLI Stanford in 1984-1986.[3]

8 Type Transparency

In computational linguistics, the purpose of formal grammars for fragments of natural language is (i) a linguistically well-motivated analysis of examples which is suitable (ii) for efficient automatic derivation by a computer program and (iii) for systematic upscaling. This requires *input-output equivalence* between the declarative derivation order of the formal grammar and the procedural derivation order of the parser.

Called *type transparency* by Berwick and Weinberg, input-output equivalence between a formal grammar and its parser was originally intended also in PSG:

> Miller and Chomsky's original (1963) suggestion is really that grammars be realized more or less directly as parsing algorithms. We might take this as a methodological principle. In this case we impose the condition that the logical organization of rules and structures incorporated in the grammar be mirrored rather exactly in the organization of the parsing mechanism. We will call this *type transparency*.
>
> Berwick and Weinberg (1984), p. 39

On page 81, Berwick and Weinberg define *absolute type transparency* as follows:

8.1 DEFINITION OF ABSOLUTE TYPE TRANSPARENCY

- For any given language, parser and generator use the same *formal grammar*,
- apply the rules of the grammar *directly*,
- in the same *order* as the grammatical derivation,
- take the same *input* expressions as the grammar, and
- produce the same *output* expressions as the grammar.

It turned out, however, that a direct application of the grammar rules by a parser is inherently impossible in PSG (FoCL pp. 175 et seq.). The historical background for this is that Post (1936) developed his production or rewrite system to mathematically characterize the notion of *effective computability* in recursion theory.[4] In this original application, a derivation order based on the substitution of signs by other signs is perfectly natural. When Chomsky (1957) borrowed the Post production system under the name Phrase Structure Grammar (PSG) for analyzing natural language, he inherited its substitution-driven derivation order.

Because a parser takes terminal strings as input but a PSG a start symbol, PSGs and their parsers are not input-output equivalent – which means that a type transparent PSG parser can not exist. Instead, huge intermediate structures are required to reconcile the time-linear input order of the parser and the top-down substitution order of the grammar's rewrite rules (Earley 1970).

Consequently, (i) the computational complexity of PSG is polynomial,[5] and (ii) debugging and upscaling in PSG-based parsing is greatly impeded: if a well-formed input is rejected or an ill-formed input accepted, the error must be found in the complex intermediate structures of the context-free PSG parser, which are not easy to read. In type-transparent LA-grammar, in contrast, an error is located in the output close to where the time-linear derivation broke off or the ill-formed continuation began. Moreover, the error is explicitly documented in the automatic analysis serving simultaneously as the trace of the parse and the linguistic analysis.[6]

9 Four Kinds of Type-Token Relations

The interaction between the DBS agent's computational cognition and its cognition-external surroundings is based on the pattern-matching of concepts. Recognition is

[3]Thanks to CSLI Stanford for their generous hospitality, especially by providing the at the time most advanced workstations by HP with a team of helpful operators, and to the DFG for a five year Heisenberg grant. The research stay was initially intended to program the Montague Grammar defined in *Surface Compositional Grammar* (SCG). Even though the syntactic-semantic λ-derivations of surfaces into formulas of intensional logic were explicitly defined to high standard, a reasonable programming of the 'fragment' presented unsurmountable difficulties. In response, a time-linear approach was developed, programmed, and published as NEWCAT, including the source code written in Lisp.

[4]See for example Church (1956), p. 52, footnote 119.

[5]In contrast to the linear time complexity of type-transparent LAG/DBS (TCS'92).

[6]FoCL Sects. 9.4, 10.4, 10.5, specifically 10.5.5.

a concept type matching raw data, resulting in a token stored in short term memory. Action is adapting a type to a purpose, resulting in a token realized as raw data.

DBS uses the type-token relation directly for elementary proplets of the semantic kinds concept, indirectly for indexicals and names, and for complex contents of declarative, interrogative, and imperative sentences.

9.1 TYPE AND TOKEN OF A CONCEPT

type	*token*
sur: Hund	sur: Hund
noun: dog	noun: dog
cat: def sg	cat: def sg
sem:	sem:
fnc:	fnc: snore
mdr:	mdr:
nc:	nc:
pc:	pc:
prn:	prn: 24

The attributes fnc and prn of the type have no value, while those of the token have the values snore and 24. The sur value is from German.

9.2 TYPE AND TOKEN OF AN INDEXICAL

type	*token*	STAR
sur: you	sur: you	S: veranda
noun: pro2	noun: pro2	T: Monday
cat: sp2	cat: sp2	A: John
sem:	sem:	R: Mary
fnc:	fnc: see	prn: 24
mdr:	mdr:	
nc:	nc:	
pc:	pc:	
prn:	prn: 24	

The type has no prn value and no STAR to point at, while the token has the prn value 24 and may point at the STAR value John or Mary, depending on the syntax.

9.3 TYPE AND TOKEN OF A NAME

type	*token*
sur: Fido	sur: Fido
noun:	noun: [dog x]
cat: snp	cat: snp
sem: m sg	sem: m sg
fnc:	fnc:
mdr:	mdr:
nc:	nc:
pc:	pc:
prn:	prn:24

The type has no prn value and the core attribute noun has no ‘named referent’,

while the token has the prn value 24 and the core attribute has the named referent [dog x].

Finally consider the type and token of a DBS proposition,[7] defined as a content. The syntactic mood is specified by the verb's cat value decl as a declarative.

9.4 TYPE OF A CONTENT

type

sur:	sur:	sur:
noun: dog	verb: find	noun: bone
cat: snp	cat: #n′ #a′ decl	cat: snp
sem: def sg	sem: past ind	sem: indef sg
fnc: find	arg: dog bone	fnc: find
mdr:	mdr:	mdr:
nc:	nc:	nc:
pc:	pc:	pc:
prn: K	prn: K	prn: K

This content is a type because there is no STAR and the prn value is a variable, here K. It is a nonlanguage content because the sur slots are empty.

9.5 CORRESPONDIG TOKEN

token

sur:	sur:	sur:	S: yard
noun: dog	verb: find	noun: bone	T: friday
cat: snp	cat: #n′ #a′ decl	cat: snp	A: sylvester
sem: def sg	sem: past ind	sem: indef sg	R:
fnc: find	arg: dog bone	fnc: find	3rd:
mdr:	mdr:	mdr:	prn: 12
nc:	nc:	nc:	
pc:	pc:	pc:	
prn: 12	prn: 12	prn: 12	

This content is a token because the three content proplets and the STAR proplet are connected by a common prn constant, here 12. According to the STAR, the content resulted as an observation by the agent Sylvester on Friday in the yard.

In summary, the types of individual proplets are lexical word form analyses which are provided by the on-board memory for automatic word form recognition/production. The type of a complex content results from concatenating proplet types with the semantic relations of structure. The content type of a proposition is turned into a content token by adding a STAR and replacing the prn variables with constants (simultaneous substitution).

[7] An elementary proposition is a content which uses exactly one prn value.

10 Conclusion

In computer science, the input-output distinction holds (i) between a system and its external environment and (ii) between interacting components within a system. The sign-based substitution-driven ontology of phrase structure grammar (PSG) avoids input-output interaction with the system-external reality by using the same S node like a start button as input for the random generation of all the different grammatical structures in the fragment. The agent-based data-driven ontology of DBS, in contrast, provides external nonlanguage input-output in (i) action and (ii) recognition between agents and their environment, and external language input and output between agents in the (iii) speak and (iv) hear modes.

The input to the (iii) speak mode is a hierarchical content and the output a linear surface. The input to the (iv) hear mode is a linear surface and the output a hierarchical content. The challenge for a functionally complete, scientific computational reconstruction of natural language communication is a bidirectional conversion between a linear and a hierarchical coding of the semantic relations of structure.

In DBS, the speak mode turns hierarchical input contents into linear output surfaces by *navigating* along the semantic relations of structure in the input. The hear mode turns linear input surfaces into hierarchical output content by incremental time-linear *syntactic-semantic composition* between the sentence start, defined as a set of proplets already connected (at least partially), and the next word, reintroducing the classical semantic relations of subject/predicate, object\predicate, modifier|modified, and conjunct−conjunct, coded by address.

Bibliography

AIJ'01 = Hausser, R. (2001) "Database Semantics for Natural Language," *Artificial Intelligence*, 130.1:27–74, Elsevier

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

CASM'17 = Hausser, R. (2017) "A computational treatment of generalized reference," *Complex Adaptive Systems Modeling*, Vol. 5.1:1–26, Springer

Church, A. (1932). "A set of postulates for the foundation of logic," *Annals of Mathematics* Series 2. 33 (2): 346–366

Earley, J. (1970) "An Efficient Context-Free Parsing Algorithm," *Commun. ACM* 13.2:94–102

FoCL = Hausser, R., ([1999, 2001] 2014) *Foundations of Computational Linguistics; Human-Computer Communication in Natural Language,* pp. 518. Springer

MacNeilage, P. (2008) *The Origin of Speech*, Oxford: OUP

Miller, G., and N. Chomsky (1963) "Finitary models of language users," in D. Luce (ed.), *Handbook of Mathematical Psychology*. John Wiley & Sons. pp. 2–419.

Neumann, J.v. (1945) *First Draft of a Report on the EDVAC*, in IEEE Annals of the History of Computing. Vol. 15, Issue 4, 1993, doi:10.1109/85.238389, pp. 27–75

NEWCAT = Hausser, R. (1986) *NEWCAT: Natural Language Parsing Using Left-Associative Grammar*, (Lecture Notes in Computer Science 231), 540 pp., Springer

Post, E. (1936) "Finite Combinatory Processes — Formulation I," *JSL*, Vol. I:103–105

SCG = Hausser, R. (1984) *Surface Compositional Grammar*, Munich: Wilhelm Fink Verlag

TCS'92 = Hausser, R. (1992) "Complexity in Left-Associative Grammar," *Theoretical Computer Science*, Vol. 106.2:283-308, Elsevier

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`