Computational Linguistics: Formal Grammar, History and Future of CL

Raffaella Bernardi

KRDB, Free University of Bozen-Bolzano

P.zza Domenicani, Room: 2.28, e-mail: bernardi@inf.unibz.it

Contents

1 Generative Power 5
1.1 Hierarchy of Grammars and Languages 6
2 Chomsky Hierarchy of Languages 8
3 Where do Natural Languages fit? 9
3.1 NLs are not Regular Languages 9
3.2 NLs are not Regular Languages (Cont'd) 10
3.3 Dissenting Views 11
3.4 Are NL Context Free (CF)? 12
3.5 Nested and Crossing Dependencies 14
3.6 English & Copy Language 15
3.7 Cross-serial dependencies in Dutch 16
3.8 Cross-serial dependencies Swiss German 17
4 Mildly Context Free Grammars 18
5 Mildly Context-sensitive Languages (MSC) 19
6 Where do the different Formal Grammars stand? 20
6.1 Dependency Grammar 21
6.2 Categorial Grammar and CTL 22

6.3 Tree Adjoining Grammars . . . 23
7 Complexity Issue . . . 24
7.1 Input length . . . 25
7.2 Complexity of a Problem . . . 26
7.3 Complexity w.r.t. Chomsky Hierarchy . . . 27
8 History of CL . . . 28
8.1 How old is CL? . . . 29
8.2 What have Computational linguists achieved? . . . 31
8.3 A different summary from someone I don't recall the name of . . . 32
8.4 Early Roots: 1940's and 1950's . . . 33
8.4.1 Automaton . . . 33
8.4.2 Probabilistic Model . . . 34
8.5 Two Camps: 1957-1970 . . . 35
8.5.1 Symbolic paradigm . . . 35
8.5.2 Stochastic paradigm . . . 36
8.6 Additional Developments: 1960's . . . 37
8.7 1970-1983 . . . 38
8.8 Revival of Empiricism and FSM's: 1983-1993 . . . 39

8.9 Reunion of a Sort: 1994-1999 40
9 Where is LCT going 41
10 Practical Stuff 42
10.1 Next LCT Colloquium 44
10.2 You & LCT 45
10.3 You & Me 47

1. Generative Power

We have seen how to use a formal grammar to recognize natural language strings/phrases.

Every (formal) grammar generates a unique language. However, one language can be generated by several different (formal) grammars.

Formal grammars differ with respect to their **generative power**:

One grammar is of a greater generative power than another if it can recognize a language that the other cannot recognize.

Two grammars are said to be

- **weakly** equivalent if they generate the same string language.
- **strongly** equivalent if they generate both the same string language and the same tree language.

Remark Some of the slides (on Chomsky Hierarchy, etc.) are from Gerhard Jaeger course given at ESSLLI 04.

1.1. Hierarchy of Grammars and Languages

Based on this observation it's possible to construct a hierarchy of grammars, where the set of languages describable by grammars of grater power subsumes the set of language describable by grammars of less power. The most commonly used hierarchy is the **Chomsky Hierarchy of Languages** introduced in 1959.

Hence, the two questions to ask are:

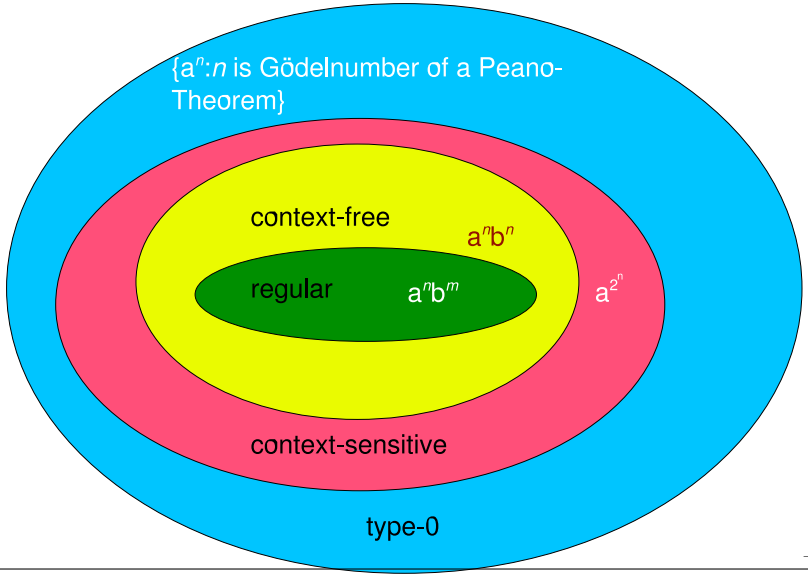
- Where does Natural Language fit in the Hierarchy?
- Which is the generative power of the different Formal Grammars?

If we know where NL fit, we would know

- which formal language can represent NL;
- which rules to use in writing formal grammars for NL.

2. Chomsky Hierarchy of Languages

The Chomsky Hierarchy



3. Where do Natural Languages fit?

3.1. NLs are not Regular Languages

It is universally agreed that NLs are not regular. E.g. an evidence was provided by Chomsky (1956, 1957). Let $S_1, S_2, \ldots S_n$ be declarative sentences, the following syntactic structures are grammatical English sentences:

- If S_1, then S_2
- Either S_3, or S_4
- The man who said S_5 is arriving today

In each case there is a lexical dependency between one part of each structure and another. "If" must be followed by "then" "either" must be followed by "or". Moreover, these sentences can be embedded in English one in another.

3.2. NLs are not Regular Languages (Cont'd)

E.g. If either the man who said S5 is arriving today or the man who said S5 is arriving tomorrow, then the man who said S6 is arriving the day after.

if	$\rightarrow a$
then	$\rightarrow b$
either	$\rightarrow c$
or	$\rightarrow d$
"other words"	$\rightarrow \epsilon$

If we consider just the "if . . . then" construction, we have $a^n b^n$ which is not a regular language.

3.3. Dissenting Views

- all arguments to this effect use center-embedding
- humans are extremely bad at processing center-embedding
- notion of competence that ignores this is dubious
- natural languages are regular after all.

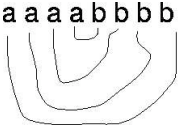
3.4. Are NL Context Free (CF)?

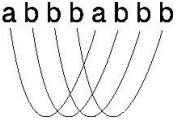
History of the problem:

1. Chomsky 1957: conjecture that natural languages are not CF
2. sixties, seventies: many attempts to prove this conjecture
3. Pullum and Gazdar 1982:
   - all these attempts have failed
   - for all we know, natural languages (conceived as string sets) might be context-free
4. Huybregts 1984, Shieber 1985: proof that Swiss German is not context-free
5. Culy 1985: proof that Bambara is not context-free

Nested and crossing dependencies

- CFLs—unlike regular languages—can have unbounded dependencies
- however, these dependencies can only be **nested**, not **crossing**
- example:
  - $a^n b^n$ has unlimited nested dependencies → context-free
  - the copy language has unlimited crossing dependencies → not context-free





3.6. English & Copy Language

Bar-Hillel and Shamir (1960): "English contains copy-language. Hence it cannot be context-free". E.g. Consider the sentence

> John, Mary, David, ... are a widower, a widow, a widower, ..., respectively.

Claim the sentence is only grammatical under the condition that if the nth name is male (female) then the nth phrase after the copula is a widower (a widow).

Counterargument :

- crossing dependencies triggered by respectively are semantic rather than syntactic.

compare above example to

> (Here are John, Mary and David.) They are a widower, a widow and a widower, respectively.

3.7. Cross-serial dependencies in Dutch

Huybregt (1976):

- Dutch has copy-language like structures
- thus Dutch is not context-free

  (1) dat Jan Marie Pieter Arabisch laat zien schrijven
  THAT JAN MARIE PIETER ARABIC LET SEE WRITE
  tr. "that Jan let Marie see Pieter write Arabic"

Counterargument

- crossing dependencies only concern argument linking, i.e. semantics
- Dutch has no case distinctions
- as far as plain string are concerned, the relevant fragment of Dutch has the structure

$$NP^nV^n$$

  which is context-free

3.8. Cross-serial dependencies Swiss German

Today many theorists believe natural languages are not context-free. (Huybregts 1984, Shieber 1985).

Evidences are given by **cross-serial dependencies** found in Swiss German where verbs and their argument can be orderded cross-serially.

A sentence can have a string of

dative nouns followed by a string of accusative nouns, followed by a string of dative-taking verbs, followed by a string of accusative-taking verbs. E.g.

mer em Hans es huus halfed aastriiche
we Hans/Dat the house/Acc helped paint.

tr. we helped Hans paint the house.

The number of verbs requaring dative objects must equal the number of dative NPs and similarly for accusatives, and this number can be arbitrary. Hence, the language representing this phenomenon is $wa^nb^mxc^nd^my$ which is not Context Free (CF).

However, notice that those construction types used to prove that NLs is not CF appear to be hard to understand for humans too.

4. Mildly Context Free Grammars

However, how large NL are continues to be a less simple matter. There are two main non-compatible views:

1. Natural Language forms a class of languages that **includes the CF** family, but is larger than it.
2. Natural Language occupies a position eccentric with respect to that hierarchy, in such a way that it does not contain any whole family in the hierarchy but is **spread along all of them**

The first view gave rise to a new family of languages which is of clear linguistic interest, Mildly Context-sensitive Languages.

5. Mildly Context-sensitive Languages (MSC)

A concept motivated by the intention of characterizing a narrow class of formal grammars which are **only slightly more powerful than CFGs**, and which nevertheless allow for descriptions of natural languages in a linguistically significant way (Joshi 1985).

According to Joshi (1985, p. 225) a mildly context-sensitive language, L, has to fulfill three criteria, to be understood as a rough characterization. Somewhat paraphrased, these are:

1. the parsing problem for L is solvable in **polynomial time** (see later),
2. L has the constant **growth property**, and
3. there is a finite **upper bound** for L limiting the number of different instantiations of factorized **cross-serial dependencies** occurring in a sentence of L.

6. Where do the different Formal Grammars stand?

The interest in the frameworks is tied to their generative power, ... as well as their **destiny**.

Chomsky's formal language theory made it possible to ask for the generative strength of a grammar.

After the discovery of languages which require cross-serial dependencies, grammars that were proved to be Context Free **lost their appeal**. Since CFGs were shown to be inadequate to model those natural languages.

6.1. Dependency Grammar

- Various authors (Cf. Gross (1964), Hays (1964), Gaifman (1965), Robinson (1970)) established that **a class of DGs are weakly equivalent to context-free PSGs**.
- But, Gross (1964)(p.49) claimed that the dependency languages are **exactly** the context-free languages.
- Gross claim turn out to be a mistake, and now there is new interested in DG.

6.2. Categorial Grammar and CTL

- Bar-Hillel, Gaifman and Shamir showed in 1964 that, like DG, Bar- Hillel's categorial grammar (**CG**) is **context-free**; cf. (Bar-Hillel, 1964).
- Chomsky (1963) conjectured that **Lambek calculi** were also **context-free**.
- This conjectured was proved by Pentus and Buszkowski in 1997
- CTL has been proved to be Mildly Sensitive (Moot), or Context Sensitive (Moot) or Turing Complete (Carpenter), accordingly to the structural rules allowed.

6.3. Tree Adjoining Grammars

TAG has been proved to be Mildly Context Free.

It has been proved that other frameworks belong to this class, too.

NL is believed (but of course not proved!) to be Mildly CF.

7. Complexity Issue

For any computational problem we face (hence for parsing a NL sentence too), we are interested in **algorithms** (step-by-step procedures) that can be used to solve the problem (hence we are interested in the parsers).

For these algorithms, we are interested in how **efficient** the algorithm is in terms of its run time or its use of memory (or other system resources such as database accesses).

In its broadest sense, the notion of efficiency involves all the various computing resources needed for executing an algorithm. However, by the most efficient algorithm one normally means the **fastest**. Time requirements are often a dominant factor determining whether or not a particular algorithm is efficient enough to be useful in practice.

Time complexity is determined from the corresponding execution time and input length.

7.1. Input length

The time requirements of an algorithm are conveniently expressed in terms of a single variable, the "size" of a problem instance, which is intended to reflect the **amount of input** data needed to describe the instance.

The time complexity function for an algorithm expresses its time requirements by giving, for each possible input length, the **largest amount of time needed by the algorithm to solve a problem instance of that size**.

We are also interested in how the algorithms fare as their **input load gets higher**; if a grammar intended for a fragment of English is extended to full texts, what impact will this have on the run time of the parser?

7.2. Complexity of a Problem

When we need to solve a problem, we are interested in the most efficient algorithm that solves it.

The complexity of a problem is the complexity of such an algorithm.

Classes We distinguish between

- polynomial time problems (PTIME)
- problems believed to be exponential time (e.g. NP, PSPACE)
- problems known to be exponential time or more difficult (e.g. EXPTIME)

See, Diego Calvanese course: Theory of Computing.

7.3. Complexity w.r.t. Chomsky Hierarchy

We are interested in the problem of determining whether a string is in the language generated/recognized by a grammar of a certain type.

- For Context Free Language the problem is polynomial.
- the same holds for Mildly CFL.
- whereas, for Context Sensitive Languages the problem is PSPACE-complete

8. History of CL

Yesterday, we have seen how the different Formal Grammar came about.

Now, we will zoom out and look at the whole field of Computational Linguistics: to understand current research and directions, it is also important to know the past (what has been tried, what succeeded, what failed and why.)

8.1. How old is CL?

"Computational linguistics, or natural language processing (NLP), is nearly as old as serious computing. Work began more than forty years ago, and one can see it going through successive phases, roughly ten year periods from the late fifties onwards."

1. The first phase, beginning in the late fifties, was linguistically oriented, focusing on machine translation, with people learning, painfully, how to do things computationally.
2. The second phase, from the late sixties to the late seventies, recognised the role of real world knowledge, was strongly motivated by AI, and drove NLP from this.
3. The third phase, dominating the eighties, acknowledged the specific modulating or controlling function for language relative to the world, and tried to capture this, in its necessarily systematic aspect, in grammatico-logical models for NLP.
4. The fourth phase, that we are in now, while taking the grammatico-logical skeleton for granted, recognises the significance of actual language usage, both

idiosyncrastic and habitual, as a constraint on performance, and is therefore heavily into data mining from corpora.

8.2. What have Computational linguists achieved?

But what can we actually do now, given NLP's necessary concerns both with generic capabilities like syntactic parsing and with particular tasks like translation, i.e. with both subsystem and whole system functions?

Read:

Karen Sparck Jones. "Natural language processing: she needs something old and something new (maybe something borrowed and something blue, too)"

8.3. A different summary from someone I don't recall the name of

Two foundational paradigms

- Automaton
- Probabilistic Models

8.4. Early Roots: 1940's and 1950's

8.4.1. Automaton

- Turing's (1936) model of algorithmic computation
- Kleene's (1951, 1956) finite automata and regular expressions
- Shannon (1948) applied probabilistic models of discrete Markov processes to automata for language
- Chomsky (1956): The first who considered finite-state machines as a way to characterize a grammar
- Led to the field of **Formal Language Theory** which used algebra and set theory to define formal languages as sequences of symbols.

8.4.2. Probabilistic Model

- algorithms for speech and language processing
- Shannon: the “noisy channel” model
- Shannon: borrowing of “entropy” from thermodynamics to measure the information content of a language

8.5. Two Camps: 1957-1970

8.5.1. Symbolic paradigm

- Chomsky:
  - Formal language theory, generative syntax, parsing
  - Linguists and computer scientists
  - Earliest complete parsing systems: Zelig Harris, UPenn
- Artificial intelligence
  - Created in the summer of 1956
  - Two-month workshop at Dartmouth
  - Focus of the field initially was the work on reasoning and logic (Newell and Simon)
  - Early natural language systems were built: 1. Worked in a single domain 2. Used pattern matching and keyword search

8.5.2. Stochastic paradigm

- Took hold in statistics and EE
- Late 50's: applied Bayesian methods to OCR
- Mosteller and Wallace (1964): applied Bayesian methods to the problem of authorship attribution for The Federalist papers.

8.6. Additional Developments: 1960's

1. First serious testable psychological models of human language processing. Based on transformational grammar
2. First on-line corpora
   - The Brown corpus of American English
   - 1 million word collection
   - Samples from 500 written texts
   - Different genres (news, novels, non-fiction, academic,.)
   - Assembled at Brown University (1963-64, Kucera and Francis)
   - William Wang's (1967) DOC (Dictionary on Computer) – On-line Chinese dialect dictionary

8.7. 1970-1983

Explosion of research

1. Stochastic paradigm: Developed speech recognition algorithms
   - HMM's
   - Developed independently by Jelinek et al. at IBM and Baker at CMU
2. Logic-based paradigm
   - Prolog, definite-clause grammars (Pereira and Warren, 1980)
   - Functional grammar (Kay, 1979) and LFG 1970-1983
3. Natural language understanding
   - SHRDLU (Winograd, 1972)
   - The Yale School: Focused on human conceptual knowledge and memory organization
   - Logic-based LUNAR question-answering system (Woods, 1973)
4. Discourse modeling paradigm

8.8. Revival of Empiricism and FSM's: 1983-1993

1. Finite-state models
   - Phonology and morphology (Kaplan and Kay, 1981)
   - Syntax (Church, 1980)
2. Return of empiricism
   - Rise of probabilistic models in speech and language processing
   - Largely influenced by work in speech recognition at IBM
3. Considerable work on natural language generation

8.9. Reunion of a Sort: 1994-1999

1. Probabilistic and data-driven models had become quite standard
2. Increases in speed and memory of computers allowed commercial exploitation of speech and language processing: Spelling and grammar checking
3. Rise of the Web emphasized the need for language based information retrieval and information extraction

9. Where is LCT going

1. future of the field

2. americana.

3. Spark Jones.

10. Practical Stuff

- **Topics of Exam** (exercises 40 %): CFG, Lambda Caclulus, CG. General questions on not technical issues.
- **Critiques**: by the 20th of Feb. (But give me the title of the chosen paper NOW!)
- **Project** written description with oral discussion (with me –by appointment before the 23rd); or oral presentation (with the other students on the 24th.)

Number of workload for the project :

Since LCT is a 4 credits course,

- you are expected to work on your own 64 hrs.
- the project gives you 40 % of the mark of the final exam, hence you are expected to work on it 25 hrs.

Tips Make use of the On-line NL(P) related **dictionaries**. Read the suggested **textbooks** (for each topic you find in the course web site the suggested book and chapter). Slides should be just a *support*!!!

10.1. Next LCT Colloquium

Speaker: Marcello Federico (IRST-ITC)
Title: The Statistical Approach to Machine translation
Time: 16:00-17:00
Place: Seminar Room, P.zza Domenicani.

My Calendar I plan to be away:

- 26/01-6/02
- 15/02-20/02
- 6/03-16/04

10.2. You & LCT

If you got interested into LCT, the possibilities for studying it at FUB (for now!) are:

- **LCT Colloquia**: We still have many Colloquia: Speech, NL generation, Corpus/Web, Multimedia, Information Extraction, Lexical Resources, Question Answering, Logic & Language.
- **Reading Group**: We can organize a reading group on LCT topics, using the LCT Colloquia program. Proposal: every two week I distribute a paper with background information for the next LCT Colloquium. After the Colloquium we discuss the paper and the Colloquium itself. Sign in, if you are interested!
- **Next year courses**:
  - Multimodal Intelligent Interfaces
  - Automatic speech recognition,
  - Cross-Language Information Technologies (Language translation)
  - Text Processing

- **Intership/stage**: you can do your internship with KRDB members on LCT related topics (eg. FSA/morphology, NLDB, DL & NL, etc.), or stage at EURAC or ITC-IRST (to be discussed!).

10.3. You & Me

I don't dare to give one more questionnaire divided by topic, but if you found the presentation of some topic unclear, not deep enough, or not necessary, please send me an e-mail.

And the last thing

> Thank you for the nice new experience!