Introduction to Linguistic Computation and Complexity Theory

CRISTIANO CHESI
NETS, IUSS CENTER FOR NEUROCOGNITION, EPISTEMOLOGY AND THEORETICAL SYNTAX

Ph.D. in Cognitive Neuroscience and Philosophy of Mind

Index

- **Formal Grammars**
  - Formal approaches to **Linguistic Competence**
  - Phrase Structure Grammars (**PSG**) and **Chomsky's hierarchy**

- **A theory for (linguistic) computation**
  - (Universal) Turing Machines (TM) and other formalisms
  - Introduction to **computability**, **complexity** e **automata theory**.
  - **Parsing** and complexity

- **Advances in linguistic formalisms and processing**
  - **Competence & Performance**
  - **Minimalist** Derivations
  - Complexity as **intervention**

Essential References

Introductory

- Jurafsky, D. & Martin, J. H. (2009) *Speech and Language Processing*. Prentice-Hall. *http://www.cs.colorado.edu/~martin/slp.html* (Only chapters 2, 12)
- Partee B., A. ter Meulen & R. Wall (1990) Mathematical Methods in Linguistics, Springer, 1990 (Only chapters 16 – 18)

Advanced

- Chesi C. (2015) On directionality of phrase structure building. *Journal of Psycholinguistic Research*. 44(1) http://dx.doi.org/10.1007/s10936-014-9330-6
- Chesi, C., & Canal, P. (2019). Person features and lexical restrictions in Italian clefts. Frontiers in Psychology, 10, 2105. http://dx.doi.org/10.3389/fpsyg.2019.02105
- Van Dyke, J. A., & McElree, B. (2006) Retrieval interference in sentence comprehension. *Journal of Memory and Language*, 55(2), 157-166.

Extended References

- Baddeley, A. (2013) Essentials of human memory (classic edition). Psychology Press.
- Chesi C., A. Moro (2014) Computational complexity in the brain. in Frederick J. Newmeyer and Laurel B. Preston (eds.), Measuring Linguistic Complexity. Oxford: OUP
- Chesi C. (2015) Il processamento in tempo reale delle frasi complesse. In atti del convegno “Compter Parler Soigner”, E.M. Ponti (ed). Pavia University Press.
- Hopcroft, Motwani & Ullman (2001) Introduction to the automata theory, languages and computation. Addison-Wesley. Boston
- Jurafsky, D. & Martin, J. H. (2000) Speech and Language Processing. Prentice-Hall.
- Stabler, E. 1997. Derivational minimalism. in Retoré, ed. Logical Aspects of Computational Linguistics. Springer
- Sprouse, J., Wagers, M., & Phillips, C. (2012). Working-memory capacity and island effects: A reminder of the issues and the facts. Language, 88

Formal grammars

STAGE I

Linguistic Competence

- What kind of **competence** (information structure) do we have?
  - A word can start by *wo*... (*word*) but not by *wb*...
  - The *s* in "sing*s*" is different from the one in "rose*s*"
  - "the rose is beautiful" Vs. *"the is beautiful rose"
  - "The cat chases the dog" >
    subj: cat(**agent**); verb: chase(**action**); obj: dog(**patient**)
  - ?the television chases the cat
  - "the houses" Vs. "some house"

- **Linguistic competence** is a **finite** knowledge that allows us to:
  - Recognizing as grammatical **an infinite** set of expressions
  - Assigning to them the correct meaning(s)

Linguistic Competence

- What to include:
  - Word order > meaning

    e.g. I saw a man in the park with a binocular

  - Agreement

    e.g. *la mela rosso (lit. *the$_{fem}$ red$_{fem}$ apple$_{mas}$*)

    Gianni ha visto Maria vs. Gianni l’ha vista

  - Non-local dependenccies (pronominal binding, syntactic movement)

    e.g. cosa$_i$ credi che Maria abbia chiesto a Luigi di comprare _$_i$?

    (what$_i$ do you think (that) M. asked to L. to buy _$_i$?)

    Gianni$_i$ promette a Maria$_j$ di _ $_{i/*j}$ andare a trovarla$_{j/k}$

    Gianni$_i$ chiede a Maria$_j$ di _ $_{*i/j}$ andare a trovarla$_{*j/k}$

    G$_i$ promises/asks to M. $_j$ _ $_{i/*j}$ $_{*i/j}$ to go to visit her $_{i/k}$ $_{*j/k}$

Grammar adequacy

- **Adequacy**: a grammar must provide an adequate description of the linguistic reality we want to describe.
- We will consider three levels of adequacy:
  - **Observational**: the language described by the grammar coincides with the one we want to describe
  - **Descriptive**: the grammatical analysis provides relevant structural descriptions that are coherent with the speakers' intuitions
  - **Explicative**: the grammar is learnable and it permits to draw conclusions on what's more or less difficult to be processed.

Basic formal notions

- **Finite sets definition**: A = {a, b, c}
- **Infinite (inductive) set definition**: A = {x: x has a propriety *p*}
- **Ordered sets** (*n*-tuples): A = (a, b, c)
- **Cardinality**: |A| = number of items of A

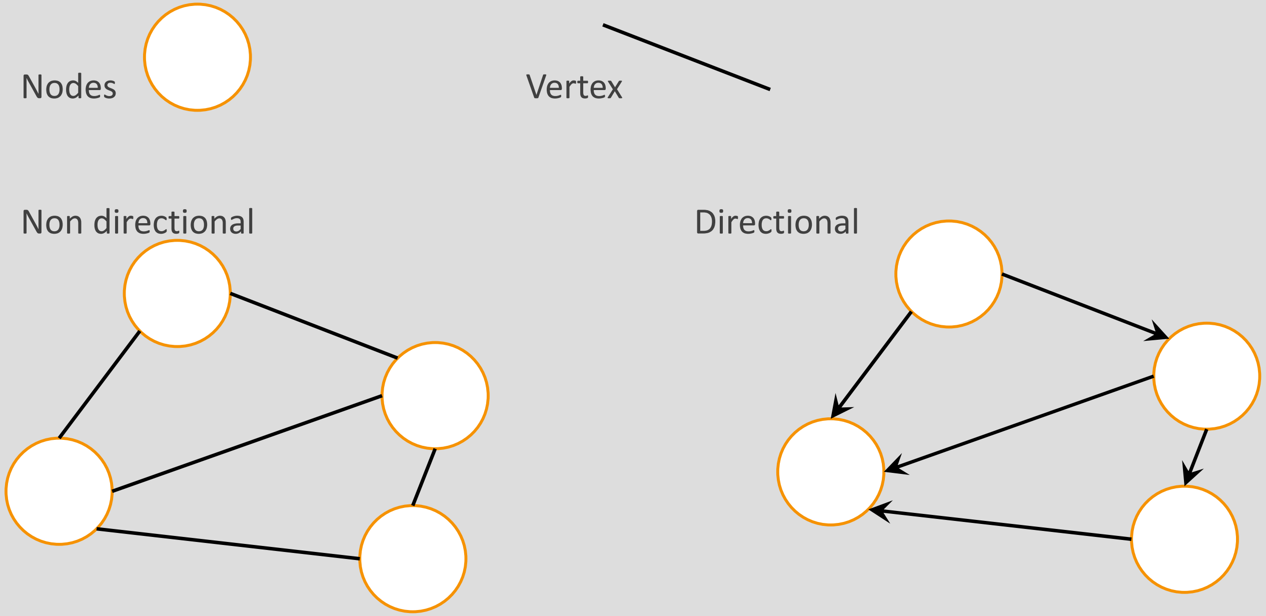
- **Cartesian product**: A = {a, b, c} B = {x, y}
  A X B = {(a, x), (b, x), (c, x), (a, y), (b, y), (c, y)}
- **Union:** $A \cup B = \{x: x \in A$ or $x \in B\}$
- **Concatenation:** $A \circ B = \{xy: x \in A$ and $y \in B\}$
- **Star (Kleene operator)**: $A^* = \{x_1x_2 \ldots x_n : n \geq 0$ for any $x_i \in A\}$

Basic formal notions

- **Indexes**: x_k = k^{th} element in a series
  x^k = a series of k elements
  X^R = mirror image of X
- **Function**: f(x) → y (x = Domain, y = Range):
- **Predicates**: f(x) → {true, false}
- **n-places predicates**: f(x, y … z) → {true, false}
- **Equivalence relation**: binary predicates R for which the following properties are valid:
  - R is **reflexive**, that is, for any x, xRx;
  - R is **symmetric**, that is, for any x and y, if xRy then yRx;
  - R is **transitive**, that is, for any x, y and z, if xRy and yRz then xRz;

Basic formal notions

- **Graphs**



Nodes

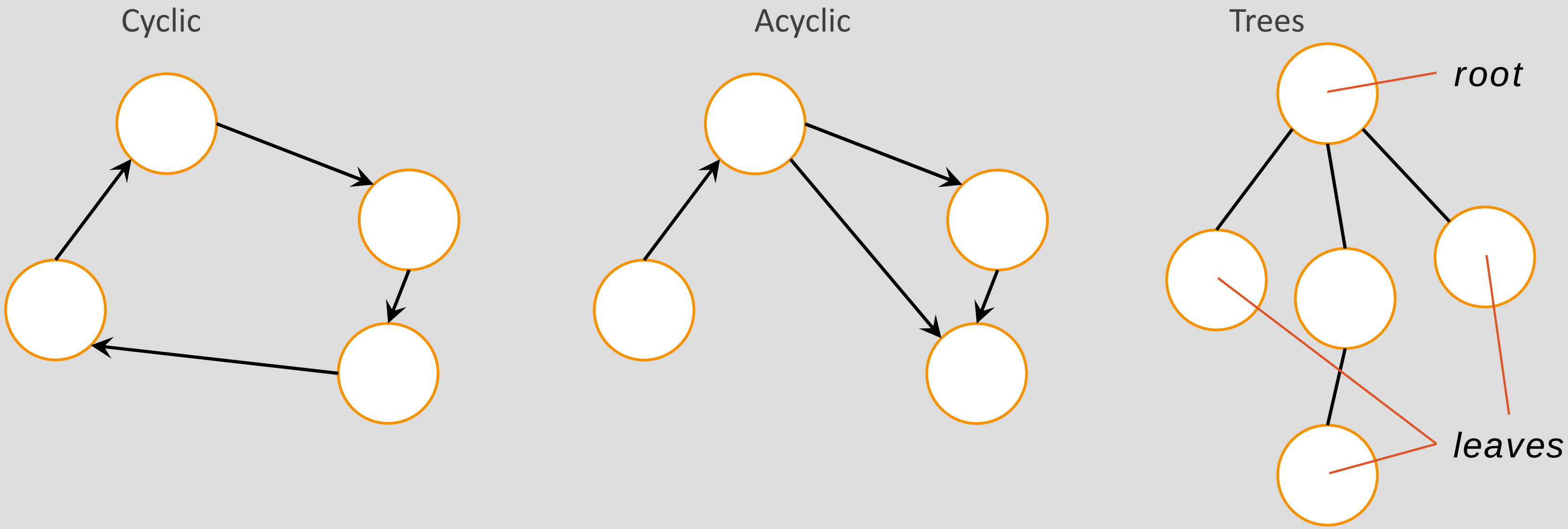
Vertex

Non directional

Directional

Basic formal notions

- **Graphs**



Cyclic

Acyclic

Trees

Degree: number of in/out vertex of a node

How to formalize a grammar

- **A** = **Alphabet**

  Finite set of chars (A^* = the set of all possible strings built concatenating elements of A; ε is the null element)

- **V** = **Vocabulary**

  (potentially in)finite set of words, built concatenating elements of A

  ($V \subseteq A^*$)

- **L** = **Language**

  (potentially in)finite set of sentences, built concatenating elements of V

  ($L \subseteq V^*$)

How to formalize a grammar

- A **formal grammar** for a language **L** is a set of rules that allows us to **recognize** and **generate** all (and only) the sentences belonging to **L** and (eventually) assign to them an adequate **structural description**.

- A Formal Grammar *G* must be:
  - **explicit** (each grammaticality judgment must be just the result of the mechanical application of the rules)
  - **consistent** (the very same sentence can’t be judged both grammatical and ungrammatical at the same time)

How to formalize a grammar

- **Phrase Structure Grammar**, **PSG** (Chomsky 1965) is an ordered 4-tuple ($\mathbf{V_T}$, $\mathbf{V_N}$, **→**, **{S}**):

V_T is the terminal vocabulary

V_N is the non-terminal vocabulary ($\mathbf{V_T} \cup \mathbf{V_N} = \mathbf{V}$)

→ is a binary, asymmetric, transitive relation defined on V*, also known as **rewriting rule**:
for any symbol $A \in \mathbf{V_N}$ $\phi A \psi \rightarrow \phi \tau \psi$ for some $\phi, \tau, \psi \in \mathbf{V^*}$

{S} is a subset of $\mathbf{V_N}$ defined as the axiom(s) of the rewriting rules.
By default, **S (Sentence)** is the only symbol present in this set.

How to formalize a grammar

- Give two strings ϕ and $\psi \in V^*$ there is a **ϕ-derivation of ψ** if $\phi \rightarrow^* \psi$.
- If there is a ϕ-derivation of ψ then we conclude that **ϕ dominates ψ**. Such a relation is reflexive and transitive.
- A ϕ-derivation of ψ is **terminated** if:
  - $\psi \in V_T^*$
  - There is no χ such that a ψ-derivation of χ exists
- Given a grammar **G**, a language generated by **G**, is said **L(G)**, that is the set ϕ of all possible strings for which a terminated S-derivation of ϕ exists

Structural description (syntactic tree)

- A **Structural Description** is a 5-tuple

  (**V**, **I**, **D**, **P**, **A**) such that:

V is a finite set of vertices (e.g. v_1, v_2, v_3…)

I is a finite set of labels (e.g. S, DP, VP, the, table…)

D is a **dominance** relation, which is a **weak relation** (namely a binary, reflexive antisymmetric and transitive relation) defined on **V**

P is a **precedence** relation, which is a **strict order** (namely a binary, anti-reflexive antisymmetric and transitive relation) defined on **V**

A is an **assignment function**;
i.e. a non surjective relation from **V** to **I**

Generative capacity and equivalence

- The generative capacity indicates the set of sentences that can be generated; two grammars can be considered **equivalent** in two senses:
  - **Weak,** if only the set of sentences is considered
  - **Strong,** if we also consider the structural description associated

Decidability

- A set Σ is considered

  - **decidable** (or recursive) if for any element *e*, belonging to the universe set, there is a mechanical procedure that in a finite set of steps terminates by saying if $e \in$ or $\notin$ to Σ (not belonging to Σ implies that *e* belongs to the complement of Σ defined as $\overline{\Sigma}$)

  - **Recursively enumerable** when a procedure exists that enumerates all and only the elements of Σ

Regular Grammars / Languages

- **Regular grammars** admit rules of this kind:

$A \rightarrow xB$

Or (systematically) of this kind:

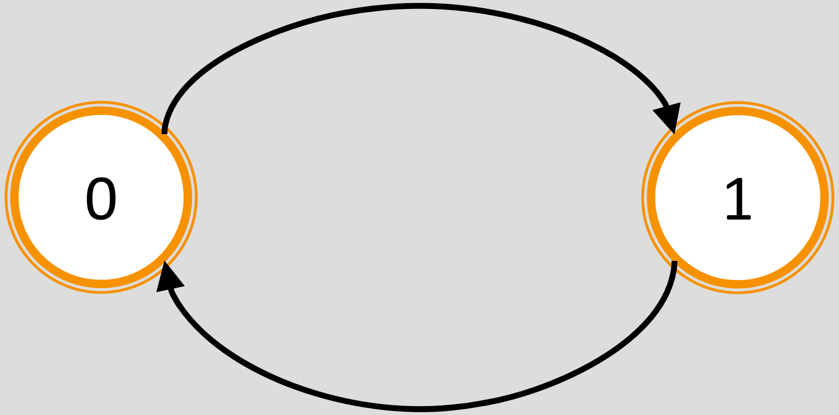
$A \rightarrow Bx$

The languages generated by such grammars are named **Regular**

Automata and computation

- **Automata** are mathematical computational models composed by states and transitions among states

- Example of automaton: electric switch!
  - 0 = on
  - 1 = off
  - -> = push



Finite State Automata (FSA)

- **Finite-State Automata** (**FSA**)

  are 5-tuples **<Q, Σ, q_0, F, δ>** where:

  Q = non-null, finite set of states

  Σ = non-null, finite set of characters (alphabet) acceptable as input

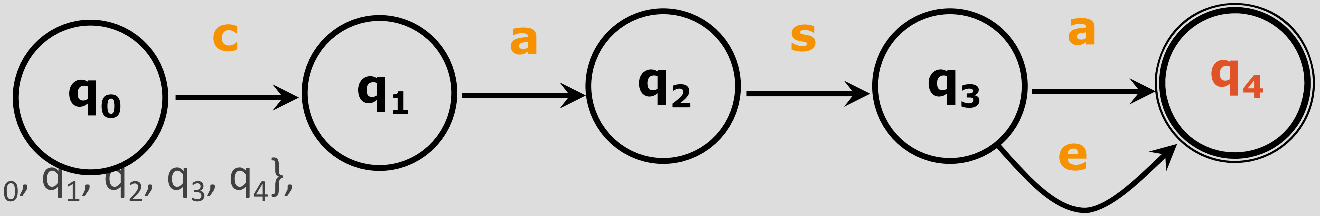
  q_0 = initial states, such that $q_0 \in Q$

  F = final states, such that $F \in Q$

  δ = finite set of rules defined from QxΣ to Q

FSA as word processors

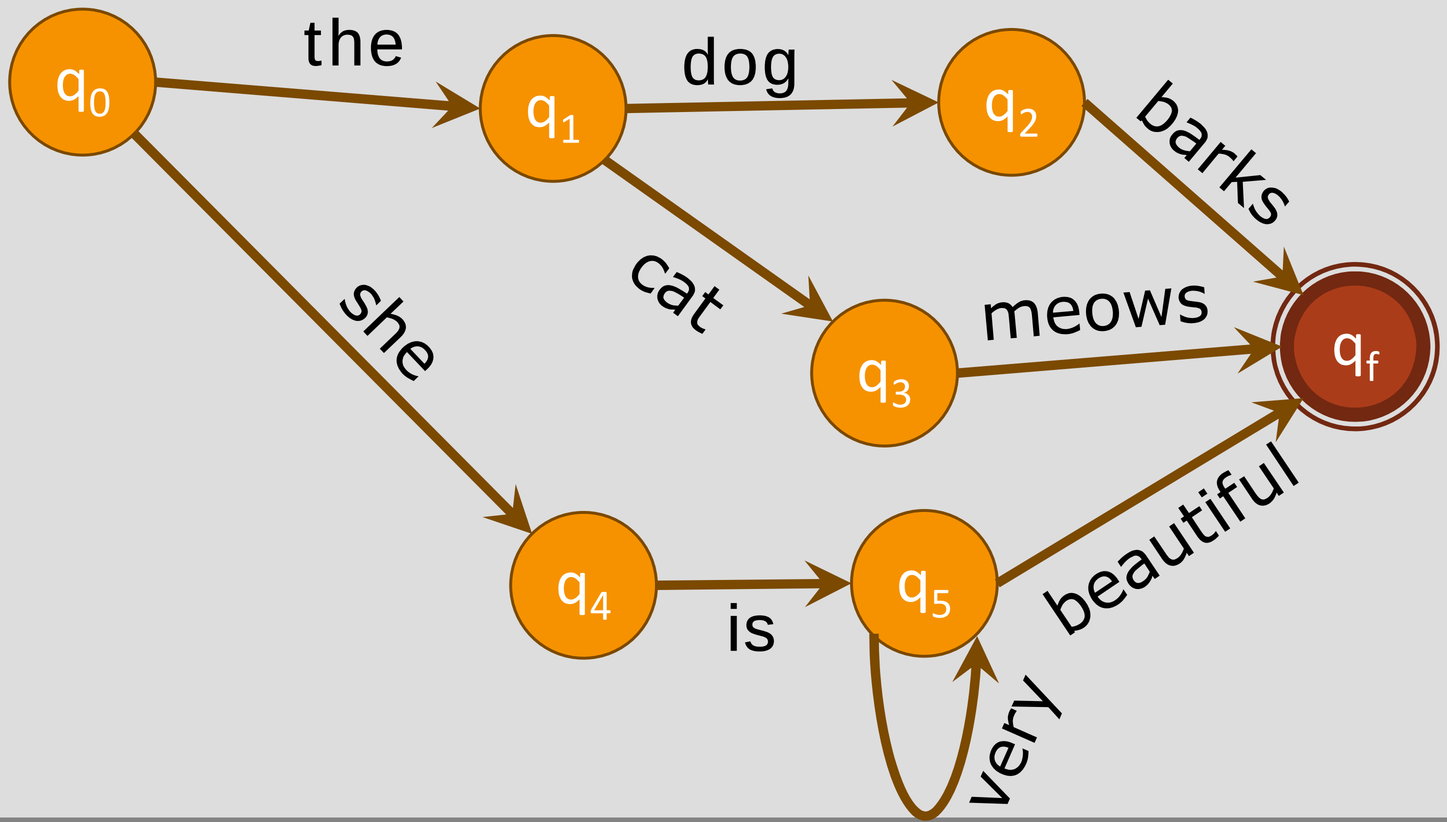
- A set of FSA is not just a mechanism that recognizes or generates a lexical item, but it might represent a sentence (or a language…).
- FSA recognizing word casa and its plural form:



- Q = {q_0, q_1, q_2, q_3, q_4},
- Σ = {c,a,s,e,#},
- Q_0={q_0},
- F ={q_4},
- δ =

	q_0	q_1	q_2	q_3	q_4
c	q_1				
a		q_2		q_4	
s			q_3		
e				q_4	

Finite State Grammars (Regular Languages)



Regular Expressions (RE)

- We can use **Regular Expressions** to query corpora for instance. **RE** are algebraic notation to express sets of strings.
- The key of a **RE** is the **identification pattern** composed by alphanumeric chars (including spacing and punctuation) and special signs indicating precise relations among chars.

RE	**Correspondence**	**E.g. Identified pattern**
[Cc]athedral	Cathedral or cathedral	The cathedral is gorgeous
[^a-z]	All but not low chars	Il duomo è …
sai?d	said or sad	You must be sad
ma.t	Any char between *a* and *t*	malt or matt
bu*	*b* followed by any (even null) number of *u*	buuuuuu! or b!
^T Vs. e$	^ = string beginning; $ = string end	The house
passe(s|d)	| expresses the logical disjuction	passes or passed
*	Escape char	A*

RG, FSA and RE equivalence

- **Regular Grammars** (RG), **Finite State Automata** (FSA) and **Regular Expressions** (RE) are equivalent, i.e. they describe the very same set of languages: **Regular Languages**.

- Proof by construction

Do you know **Eliza**?

- **Eliza** (Weizenbaum, anno 1966):

  - Human: Well, my boyfriend made me come here.
  - ***ELIZA:*** ***Your boyfriend made you come here?***
  - Human: He says I'm depressed much of the time.
  - ***ELIZA:*** ***I am sorry to hear you are depressed.***
  - Human: It's true. I'm unhappy.
  - ***ELIZA:*** ***Do you think coming here will help you not to be unhappy?***

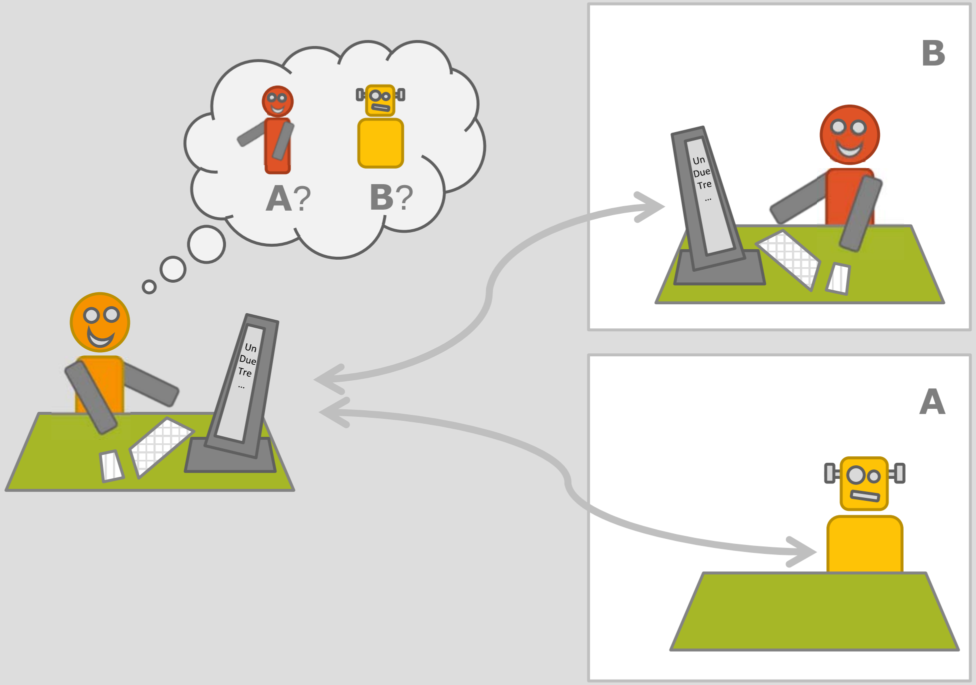
Eliza uses regular expressions!

- **RE** and **Substitution**
  - s/Regular_Expression_1/Regular_Expression_2/
  - s/ www\.[a-z]*\.com / www\.wow\.it/

- **Registers**: using block operators (round brackets indicates a block), we can reuse a matched pattern:
  - s/ the **(house|car)** has been bought by **(Mary|John)**/ **\2** bought the **\1** /

- Substitutions by **ELIZA**:
  - s/ I'm [.* ?]**(depressed|sad)**/I'm sorry to hear that you are **\1**/
  - s/ everybody is **(.*)** / in which sense they are **\1**?/
  - s/ always / can you make a specific example?

Turing's Test (the imitation game)

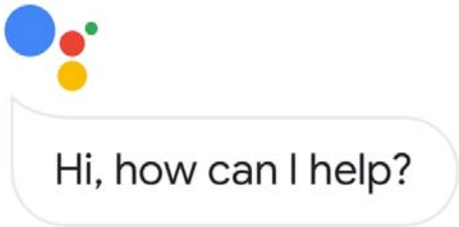


Trivia: chatbots

Alexa

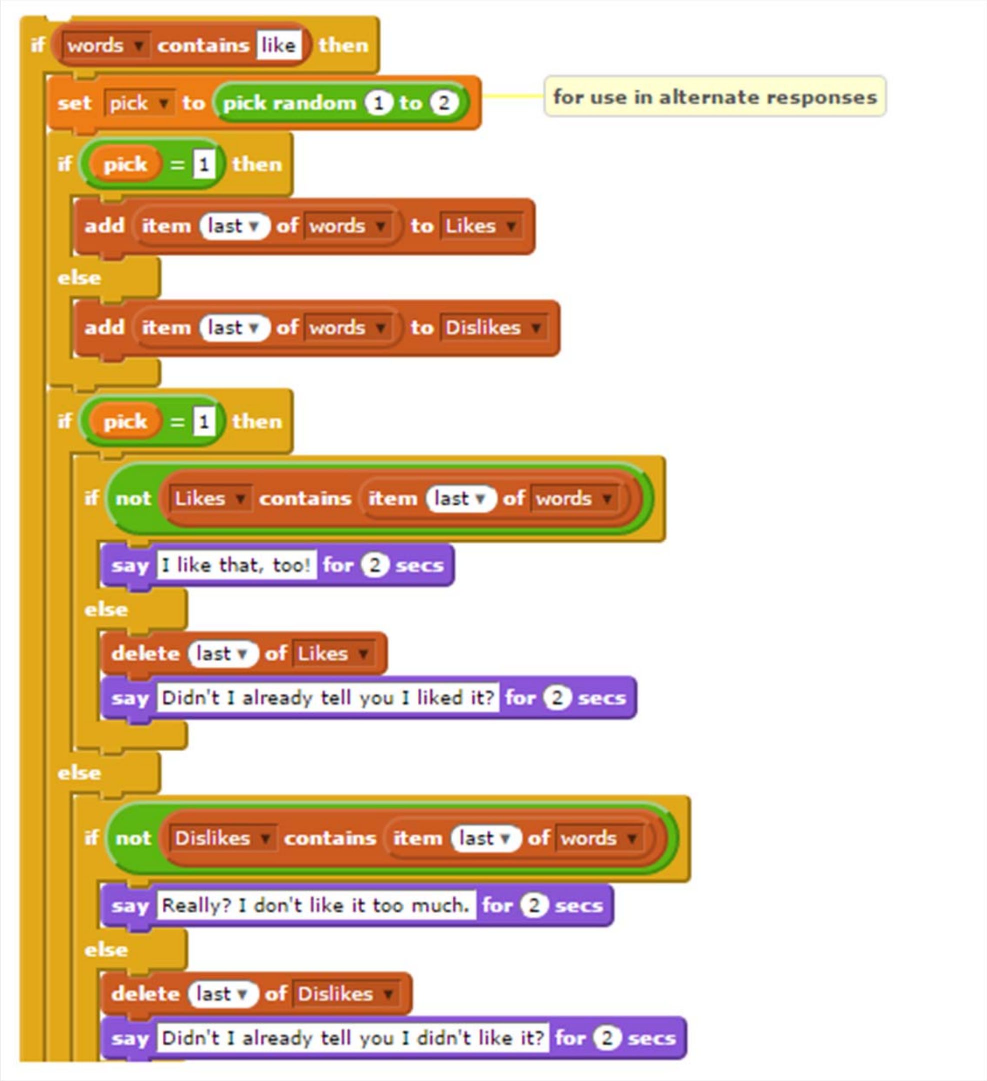
Cortana

Google Assistant



Trivia: chatbots

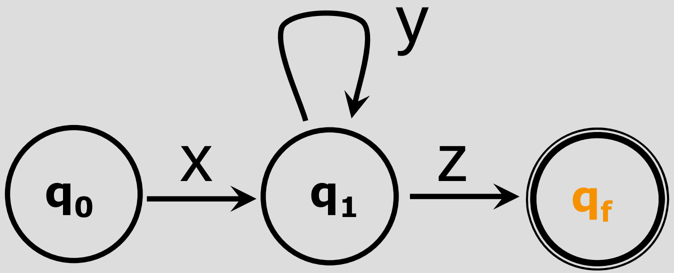
Programming a chatbot
for *like/dislike* patterns
using «Scratch»



```
if words contains like then
    set pick to pick random 1 to 2        for use in alternate responses
    if pick = 1 then
        add item last of words to Likes
    else
        add item last of words to Dislikes
    if pick = 1 then
        if not Likes contains item last of words
            say I like that, too! for 2 secs
        else
            delete last of Likes
            say Didn't I already tell you I liked it? for 2 secs
    else
        if not Dislikes contains item last of words
            say Really? I don't like it too much. for 2 secs
        else
            delete last of Dislikes
            say Didn't I already tell you I didn't like it? for 2 secs
```

How to determine if a string can be generated by a Regular Grammar?

- **Pumping lemma for Regular Grammar**
  If A is a Regular Language, then there is a number *p* (expressing «pumping» magnitude), for which, if *s* is a generic string A of length at least equal to *p*, then it can be split in 3 parts,
  s = *xyz* such that:
  I. For any $i \geq 0$, $xy^iz \in A$
  II. $|y| \rangle 0$
  III. $|xy| \leq p$



- **a**a^nb^n (**counting recursion**) cannot be generated by Regular Grammars (no way to pump a number of *as* followed by the very same number of *bs*)

Context-Free Grammars

- **Context-Free Grammars** (**CFG**) admits only this kind of rules:

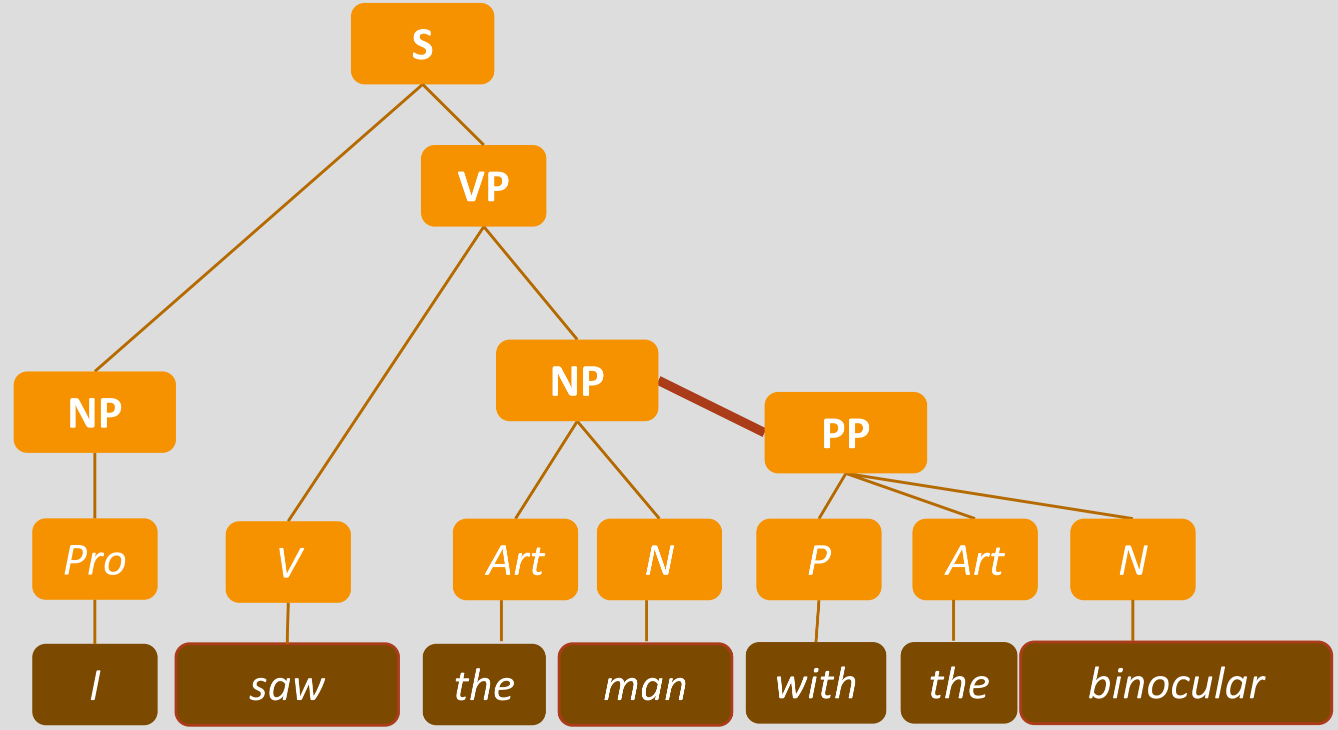
  $A \rightarrow \gamma$ (where γ is any sequence of (non)terminal symbols)

Languages generated by CFG are named **Context-Free Languages**

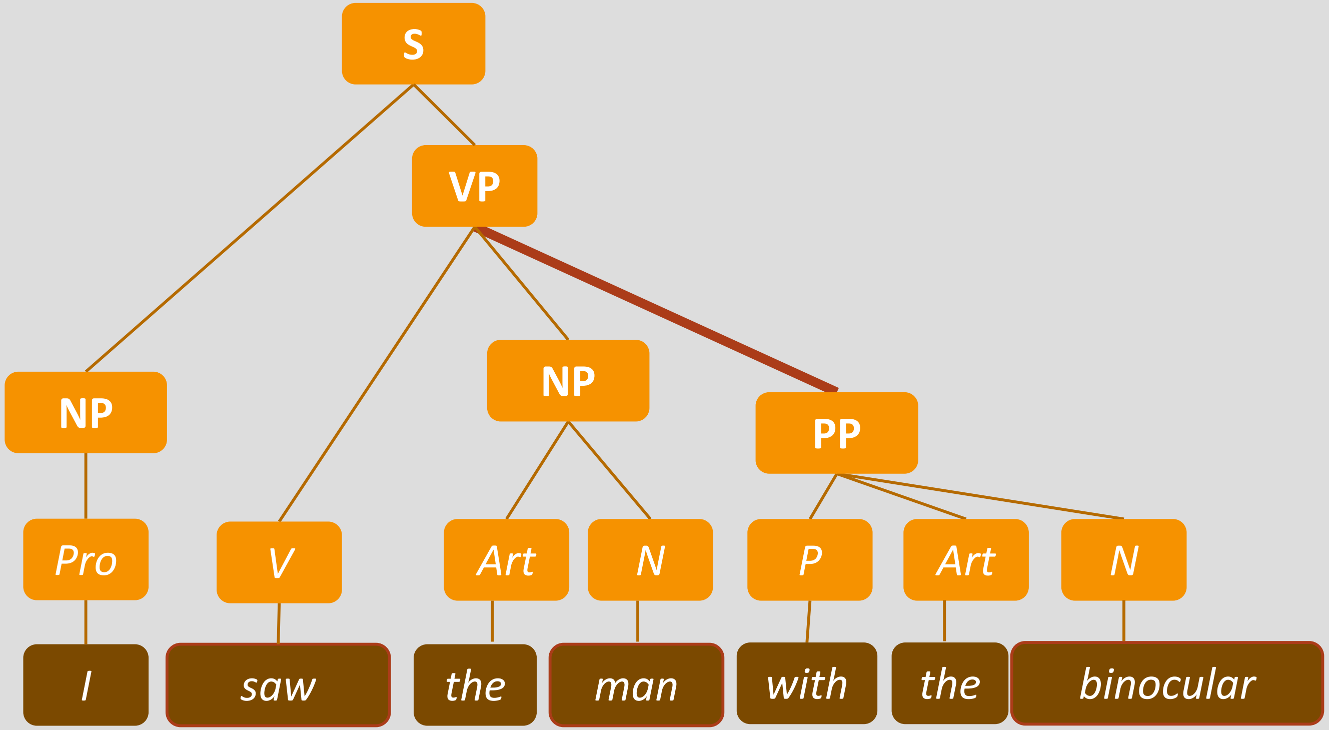
- Any **CFG** can be «converted» in a (weakly) equivalent CFG in the **Chomsky Normal Form** (**CNF**):

  $A \rightarrow BC$
  $A \rightarrow a$

Describing syntactic ambiguity



Describing syntactic ambiguity



Describing syntactic ambiguity

- Rules with the same left-side symbol should be present in the grammar to permit ambiguity:

  - VP → V NP
  - VP → V NP PP

  - NP → D N
  - NP → D N PP

Push-Down Automata

- A **Push-Down Automata** (PDA) is a finite state automata endowed with a memory **stack**; PDAs are defined by 6-tuples **<Q, Σ, q_0, F, δ, Γ>** where:

Q = finite and non-null set of states

Σ = finite and non-null set of characters accepted as input (alphabet)

q_0 = initial state(s), such that $q_0 \in Q$

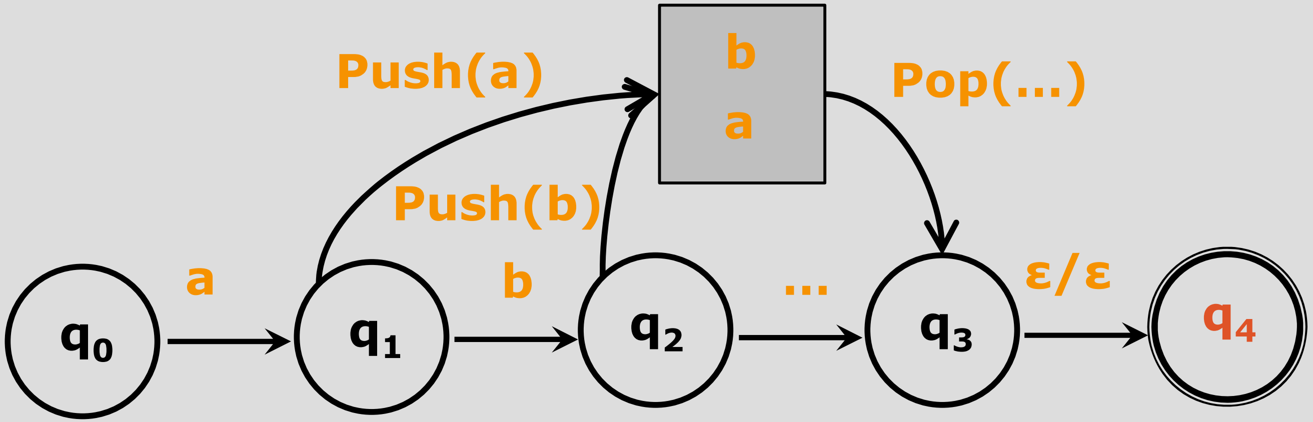
F = final states(s), such that $F \in Q$

δ = finite and non-null set of transitional rules defined from $Q \times \Sigma \times \Gamma$ to $Q \times \Gamma$

Γ = finite and non-null set of characters that can be stored in memory (Γ can have the same symbols as **Σ**)

PDA can parse mirror recursion

- XX^R



- Q = {q_0, q_1, q_2, q_3, q_4},
- Σ / Γ = {a, b, ε},
- Q_0={q_0},
- F ={q_4},
- δ =

	q_0	q_1	q_2	q_n	q_4
a	q_1 *push(a)*				
b		q_2 *push(b)*			
…			q_n *pop(…)*		
ε				q_4	

CFG and PDA equivalence

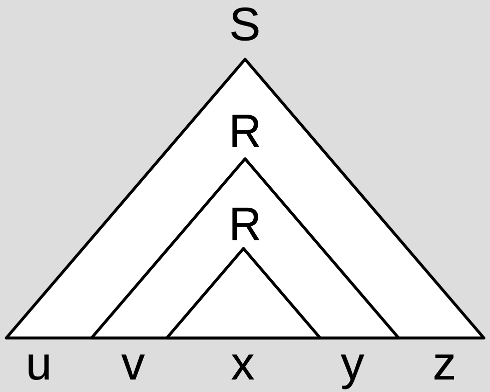
- **Context-Free Grammars** (CFG), and **Push-Down Automata** (PDA) are equivalent (i.e. they describe the very same set of languages: the Context-Free Languages).

«Demonstration» by construction:

1. For any S rule, create a PDA q_0 rule such that:
   $(q_1, \varepsilon, \varepsilon) \longrightarrow (q_1, S)$
2. For any other CFG rule such that A → x, create PDA rules such that:
   $(q_1, \varepsilon, A) \longrightarrow (q_1, x)$
3. For any symbol *a* : *a* $\in V_T$, create PDA rules such that:
   $(q_1, a, a) \longrightarrow (q_1, \varepsilon)$

Limits of CFGs?

- **Pumping lemma for Context-Free Grammars**
  If *A* is a Context-Free Language, then there is a number *p* (expressing the «pumping» length), for which, if *s* is a string of *A* of length at least equal to *p*, then it can be divided in 5 parts,
  s = uvxyz such that:
  I. For any i $\geq$ 0, $uv^ixy^iz \in A$
  II. $|vy| \rangle 0$
  III. $|vxy| \leq p$



- E.g. neither $a^nb^nc^n$ nor **XX** is not generable by CFGs.

Inclusion relations among Grammars

- **Chomsky's Hierarchy** (1956, 59):

Type 3: **Regular Grammars** (equivalent device: Finite State Automata)

$A \rightarrow xB$

Type 2: **Context Free Grammars** (equivalent device: Push-Down Automata)

$A \rightarrow \gamma$

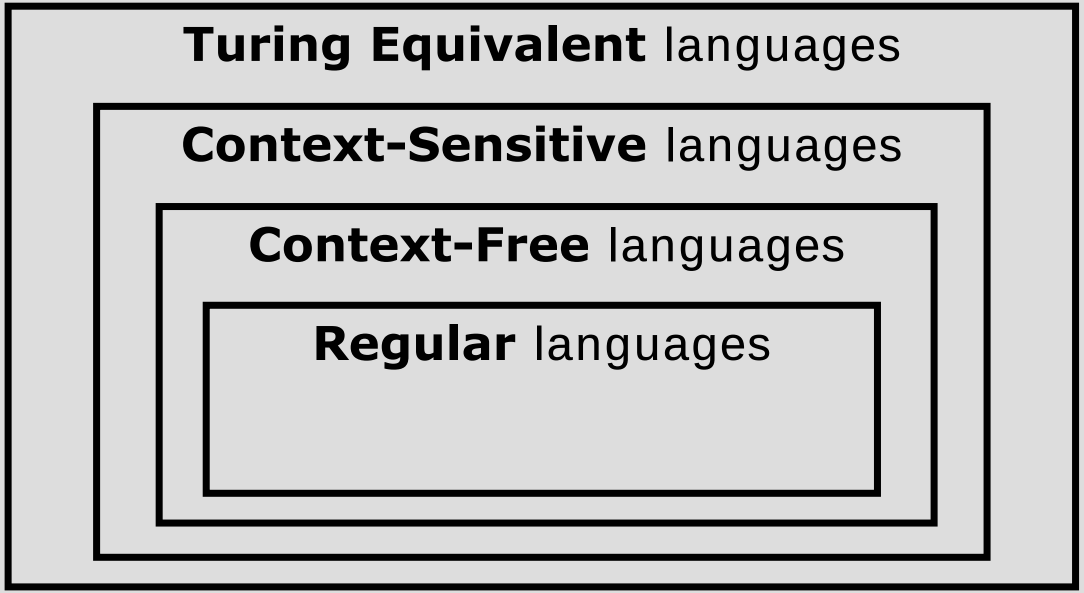
Type 1: **Context Sensitive Grammars** (e.g.: Linear-Bounded Automata)

$\alpha A \beta \rightarrow \alpha \gamma \beta$ $(\gamma \neq \varepsilon)$

Type 0: **Turing Equivalent Grammars** (e.g. Augmented Transition Networks)

$\alpha \rightarrow \beta$ $(\alpha \neq \varepsilon)$

Chomsky's Hierarchy



Turing Equivalent languages

Context-Sensitive languages

Context-Free languages

Regular languages

Where are Natural Languages?

- Natural languages are **NOT generable** by **Regular Grammars** (Chomsky 1956):

  If X **then** Y (with A and B potenzially of the form “**if** X **then** Y”, genereting then a **counting dependency** of the ***a****^{n}* ***b****^{n}* kind, that is: ***if***^{n} ***then***^{n})

- Natural languages are **NOT** even **generable** by **Context-Free Grammars** (Shieber 1985):

  Jan säit das mer em Hans es huus hälfed aastriiche
  (“famous” Swiss-German dialect)
  J. says that we to H. The house have helped painting

  Gianni, Luisa e Mario sono rispettivamente
  sposato, divorziata e scapolo
  (“***ABC...ABC***”... Are languages of the ***XX*** kind)

Where are Natural Languages?

- **Recursion** in natural languages (that is, how to make infinite use of finite means):

  - **Right recursion** (ab^n: iteration or «tail recursion»):

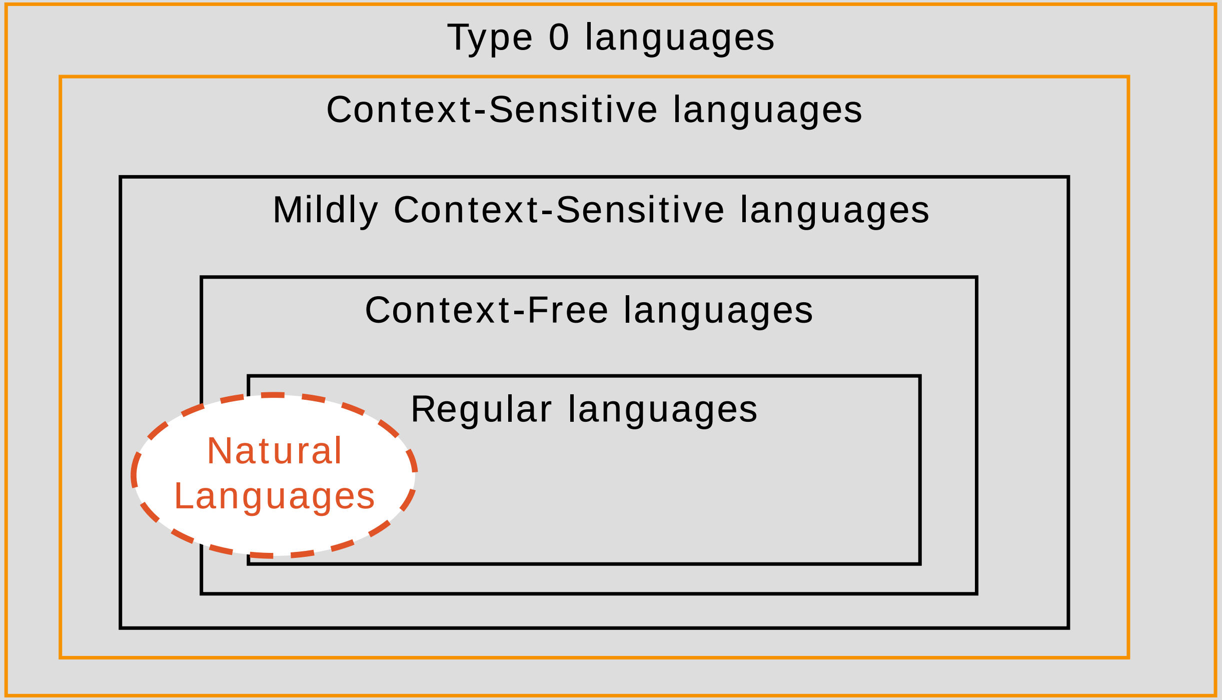
    [the dog bit [the cat [that chased [the mouse [that ran]]]]]

  - **Center embedding** (a^nb^n: counting recursion or «true recursion»):
    [the mouse [(that) the cat [(that) the dog bit] chased] ran]

  - **Cross-serial dependencies** (xx, identity recursion)

    Aldo, Bea e Carlo sono rispettivamente sposato, nubile e divorziato
    A$_{.male,}$ *B*$_{.female}$ *,* *C*$_{.male}$ *are respectively married*$_{male}$*, unmarried*$_{female}$ *& divorced*$_{male}$

Where are Natural Languages?



Today's key concepts

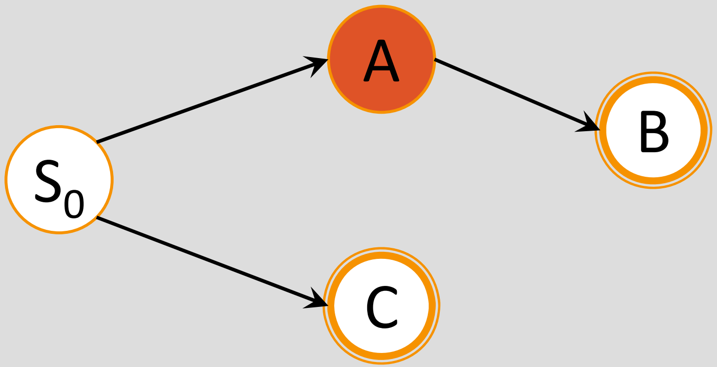
- **What's a formal grammar**
  - **Rewriting Rules** and **Recursion**
  - Rewriting Rules **restrictions** create grammar classes organized in an inclusion hierarchy (**Chomsky's Hierarchy**)
  - **Regular Grammars** (RG), **Regular Expressions** (RE) and **Finite State Automata** (FSA) **equivalence**
  - **Context-Free Grammars** (CFG) and **Push-Down Automata** (PDA) **equivalence**
  - Using **pumping lemmas** to decide if a certain string property can be captured of not by a certain class of grammars
  - **Natural languages** are **neither** Regular, **nor** Context-Free (though RGs and CFGs are often used to process Natural Languages!)

Theory of (linguistic) Computation

STAGE II

Why having a computational model

- Predict possible dysfunctions



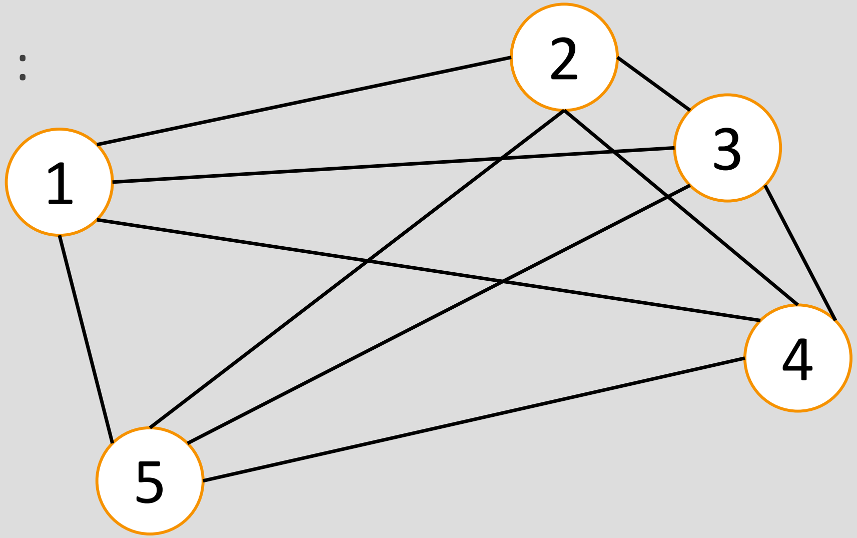
- Calculate the complexity of certain processes…

What’s complexity

- Sorting problem: order the following 5 numbers

1 3 2 7 5

- travelling salesman problem :
  find the shortest path
  connecting 5 cities

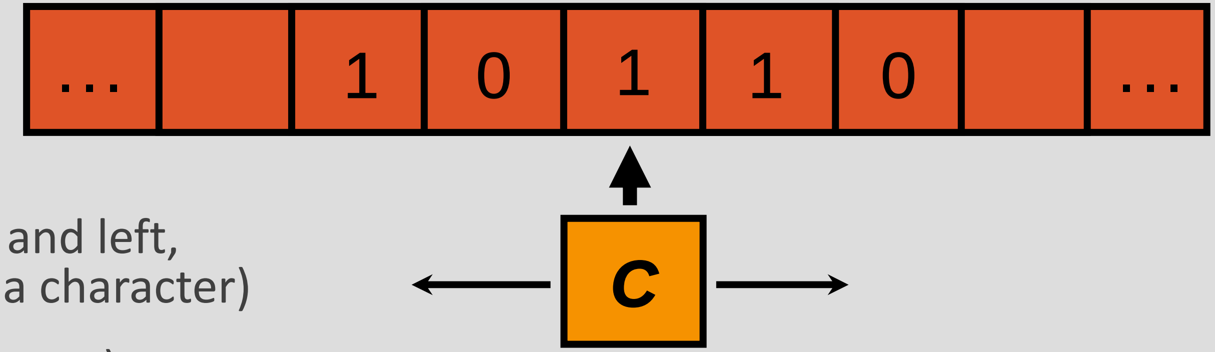


What's computable

- (informally speaking) a **computation** is **a relation between an input and an output**. This relation can be defined by various algorithms: a series of computational states and transitions among them until the final state is reached. A computation attempts at reaching the final state through legal steps admitted by the computational model (**problem space** = **set of all possible states the computation can reach**).
- **Turing-Church thesis** (simplified)
  every computation realized by a physical device can be realized by means of an algorithm; if the physical device completes the computation in n steps, the algorithm will take m steps, with m differing from n by, at worst, a polynomial.
- Some algorithm might take too much time to find a solution (e.g. years or even centuries); other algorithms can not even terminate!

Turing Machine

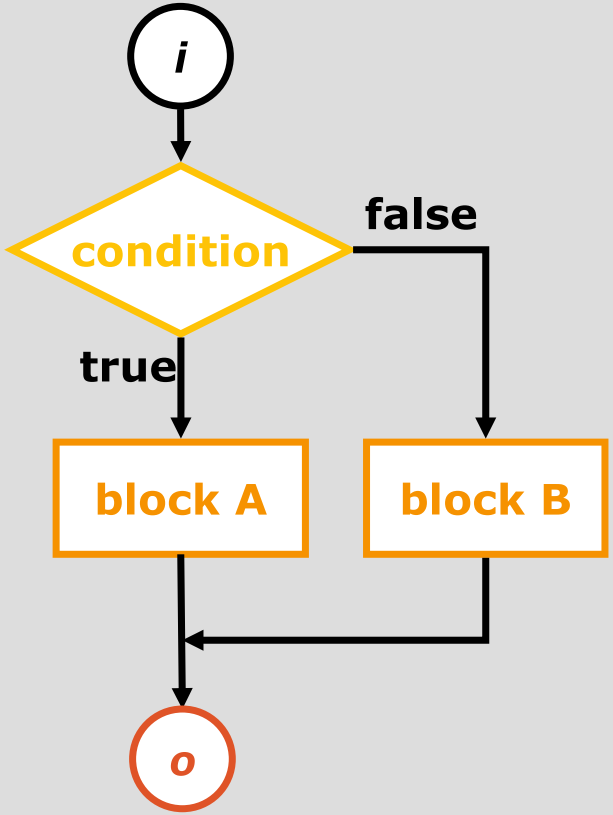
- **Infinite tape** subdivided in cells
- **alphabet A** (e.g. A ={0, 1})



- **cursor C** (that can move right and left, and can read, delete or write a character)
- Finite set of **states Q** = (q_0, q_1 … q_n)
- Finite **input I** constituted by a sequence of characters of A
- Finite set of **states S** described as 5-tuples $<q_i a b v q_j>$ such that $q_i, q_j \in$ Q; $a, b \in$ A; v = {right, left}

Flow charts

- Oriented graph:
  - An **input** (i)
  - One or more **exit** (o)
  - Finite set of **instructions** blocks such that any instruction is in the form X = Y, X = X+1, X = X-1
  - Finite set of special blocks, named **conditions**, of the (Boolean) form X = Y?
  - A finite set of **connectors** that links the blocks, such that from every block just one arrow goes outbound and, in case of conditional blocks, 2 arrows go outbound



Modularity

- **Turing Machines** and **flow charts** are **equivalent**:
  they express the very same class of function (**computable functions**)

- Both formalisms guarantee **compositionality** (M1 • M2).

- Hence: “divide et impera” is a programming paradigm that suggests decoupling a problem in smaller sub-problems for which a solution would be easier to be found.

Complexity

- Directly proportional to the **resource usage**:
  - **Time** (time complexity): number of elementary steps needed
  - **Memory** (space complexity): quantity of information to be stored at each step

- Complexity is directly proportional to the problem dimension (e.g. ordering 1000 words will be more complex that ordering 10 words);

- Grammar complexity should be related to its generative power.

Complexity

- The problem dimension is expressed in terms of input length to be processed

- The order of complexity should be expressed in terms of input length, e.g.:
  - $c \cdot n^2$ (example of polynomial time problem complexity)
  - n = input length
  - c = constant data (depending on the kind of computation)

- In this case we will say that the complexity order of the problem is n^2 since the c constant will be irrelevant with respect to n growing to the infinite.
  Such complexity order is defined as: $O(n^2)$.

Complexity

- We are interested in the growing rate of the complexity function expressing the mapping between input and output in terms of **input dimension**

- For space and time **limited problems** (resource usage surely finite) the complexity calculus is **irrelevant**

- For ***n*** growing to the infinite, as in the case of the grammars we want to study, the growing rate is crucial for determining the **tractability of the problem**

- A problem is considered **computable/tractable** if a procedure exists and **terminates** with an answer (positive or negative) in a **finite amount of time**

Complexity

- A problem with **exponential time complexity** (e.g. ***O(2^N)***) will be hardly computable in a reasonable amount of time. To have an idea, assume a device able to deal with **1 million steps per second**, there the calculation for specific input given specific complexity function:

input length →	10	20	50	100
↓ function				
N^2	0,0001 second	0,0004 sec.	0,0025 sec.	0,01 sec.
N^5	0,1 sec.	3,2 sec.	5 min. e 2 sec.	2 hours and 8 min.
2^N	0,001 sec.	1 sec.	35 year e 7 months	400 trillions of centuries
N!	3,6 sec.	about 771 centuries	A number of centuries with 48 digits	A number of centuries with 148 digits
N^N	2 hours and 8 minutes	More than 3 trillions of years	A number of centuries with 75 digits	A number of centuries with 185 digits

Complexity of classic problems

- **3SAT** problem (satisfability problem or **SAT**)
find a value assignment for all propositional letters satisfying the formula below:

  $(a \vee \neg b \vee c) \wedge (\neg a \vee b \vee \neg c) \wedge (a \vee b \vee c) \wedge \ldots$

- In **the worst case**, all possible assignments must be evaluated, that is $\boldsymbol{2^N}$ (where ***2*** are the possible **assignment values**, True and False, and ***N*** is the number of propositionals *a*, *b*, *c*...).

- The problem has an **exponential time growth complexity** function, but, once solved, can be readily proved: hard to solve, easy to verify!

Complexity of classic problems

- **Quantified Boolean Formula** (**QBF**) problem
  find a value assignment for all propositional letters satisfying the formula below :
  - $Qx_1, Qx_2 \dots Qx_n$ F(x_1, $x_2 \dots x_n$)
  - (with Q = ∃ or ∀)

- The problem is **hard to be solved**, as **3SAT**, but also **hard to be verified**:
  the 3SAT problem is a special case of QBF where all Q are existential

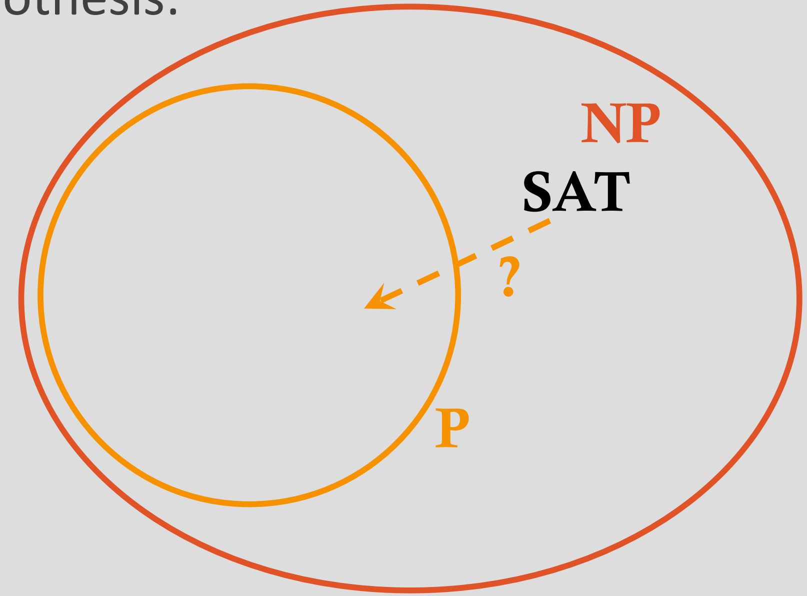
- The universal quantification requires any assignment of values to be verified.

Complexity of classic problems and reducibility

- If a computer effectively solve a problem like 3SAT, it will use an algorithm that is, at worst, **polynomial**.
- Because of the problem structure/space, such algorithm should be necessarily **non-deterministic**.
- We call the complexity of this king of problems **NP** (**Non-deterministic Polynomial time**)
- Problem with complexity **P** are deterministic and polynomial. Problems with an order **P** of complexity are (probably) included in problems with a **NP** complexity order (**no proof of reducibility from NP to P exists**… yet).

Complexity of classic problems and reducibility

- Hypothesis:



P ⊆ NP

- Problems like SAT are dubbed NP-hard (same difficulties, i.e. problem structure/space with respect to NP class problems).

What’s Parsing

- Given a Grammar ***G*** and an input ***i***, parsing ***i*** means applying a function ***p(G, i)*** able to:
  - Accept/Reject ***i***
  - Assign to ***i*** an adequate descriptive structure (e.g. syntactic tree)

Universal Recognition Problem (URP) and reduction

- **Universal Recognition Problem (URP)**
  Given a Grammar ***G*** (in any grammatical framework) and a string ***x***, ***x*** belongs to the language generable by ***G***?

- **Reduction**
  is there any **efficient mapping** from this problem to a another well know problem for which we can easily evaluate the complexity?

- YES… **SAT** problem!

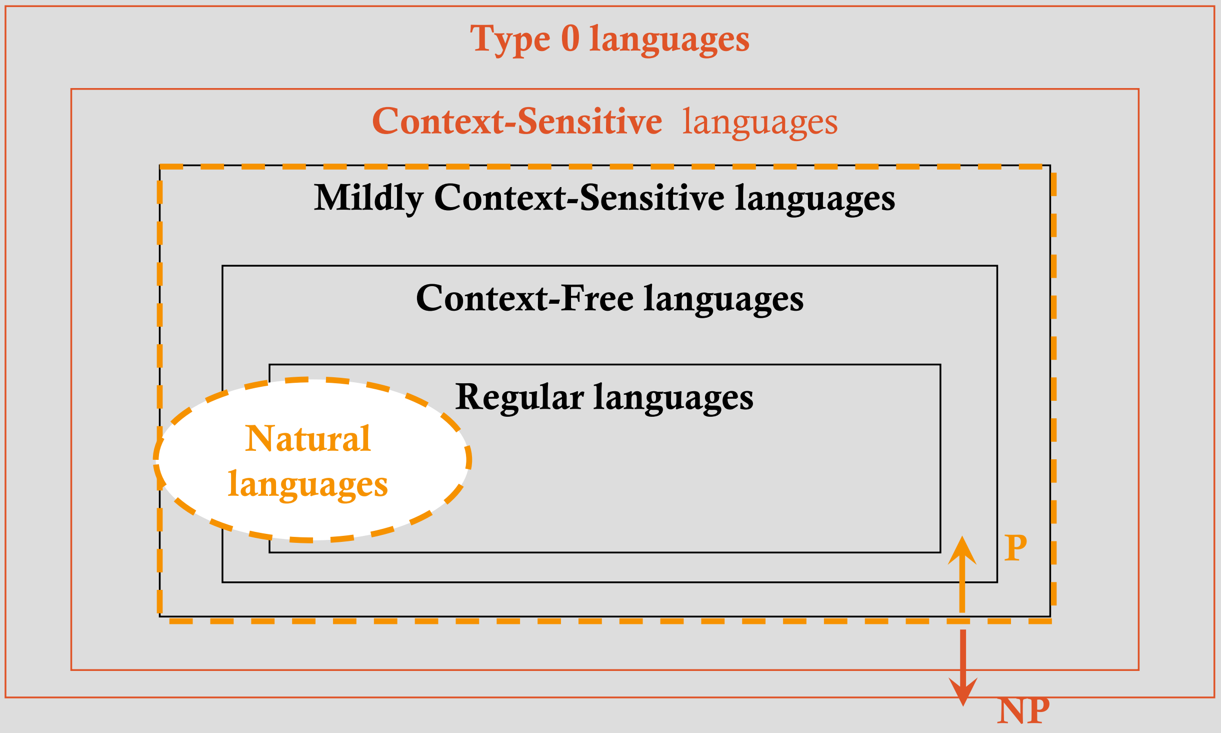
Universal Recognition Problem (URP) and reduction

- **URP** is a generalized parsing problem that can be reduced to **SAT** in its core critical structure

- In a nutshell: a string *x*, as a propositional *a* in a **SAT** formula, can receive an ambiguous value assignment (for instance “vecchia” in Italian can both be a noun and an adjectival, while a can be true or false).
  We then need to keep the assignment coherent in *x* (to evaluate the correctness of the final outcome) as in a **SAT** formula.

- We conclude that **URP** is at **least as complex as SAT**, that is, **NP-hard**!

Chomsky's hierarchy and complexity



Psycholinguistic complexity

- **Complexity** = difficulty in processing a sentence
- **Hypothesis 1**: formal complexity = psycholinguistic complexity
- **Hypothesis 2**: limited processing memory
  - On the one hand, **memory buffer capacity** could be sufficient to store only *N* structures;
  - On the other, using the memory for storing **similar** incomplete structures might create **confusion**.

Psycholinguistic complexity

- **Hypothesis 1**
  processing non context-free structures causes major difficulties
  (Pullum & Gazdar 1982)

- **Hypothesis 2**
  Limited-size Stack (Yngve 1960)
  linguistic processing uses a stack to store partial analyses.
  The more partial phrases are stored in the stack,
  the harder the processing will be.

Psycholinguistic complexity

- **Syntactic Prediction Locality Theory** (SPLT, Gibson 1998) total **memory load** is proportional to the sum of required an integration + referentiality needs:

  - **DPs** required **VPs** (in SVO languages):
    DP DP DP VP VP VP... is harder than DP VP

  - A **pronoun** referring to an already introduced referential entity is **less complex** than a **new referent** (***pro < full DPs***).

Grammar and Parsing

- **Grammars** (generally) are **declarative devices** that does not specify algorithmically how an input must be analyzed.

- **non-determinism** (multiple options all equally suitable in a given context) and recursion are critical in parsing: not all rules lead to a grammatical tree-structure in the end... And sometimes some algorithm could not even terminate!

Problem Space and searching strategies

- Given a **sentence** and a **grammar** the **parser** should tell us if the sentence is **generable** by the grammar (URP, Universal Recognition Problem) and, in the affirmative case, provide an **adequate tree structure**

- The problem space is the **complete forest of trees** and **subtrees** that can be legally generated by applying the grammatical rules in a given context

Problem Space and searching strategies

- English ambiguity:
  - Buffalo Buffalo buffalo Buffalo Baffalo
  - «a buffalo from Buffalo intimidates another buffalo from Buffalo» https://www.youtube.com/watch?v=TWbzjGIec20

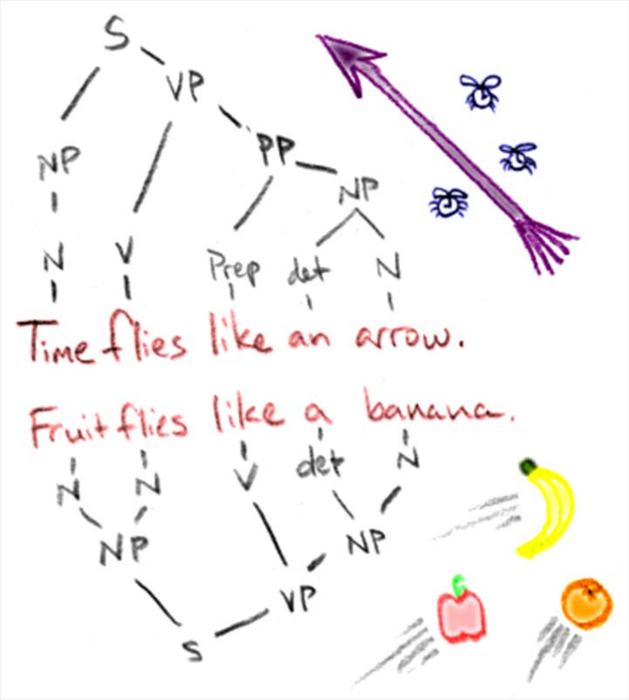
- Grammar:

→ (non terminals)	→ (terminals)
S → DP VP	N → buffalo
VP → V DP	Np → Buffalo
DP → N Np	V → buffalo

Problem Space and searching strategies

- English ambiguity:
  - Time flies like an arrow
  - Fruit flies like a banana
- Grammar:

→ (non terminals)		→ (terminals)	
S → DP VP	VP → V PP	N → time	N → fruit
VP → V PP	PP → P DP	N → flies	V → flies
DP → N	DP → D N	N → bananas	D → a(n)
DP → N N		P → like	V → like



Problem Space and searching strategies

- Italian sentence:
  - la vecchia legge la regola
    («the old rule regulates it» vs. «the old woman reads the rule»)
- Grammar:

→ (non terminals)	→ (terminals)
S → DP VP	pro → la
VP → V DP	D → la
VP → pro V	AGG → vecchia
DP → D NP	N → vecchia
NP → (AGG) N;	N → legge
	N→ regola
	V → legge
	V→ regola

Problem Space and searching strategies

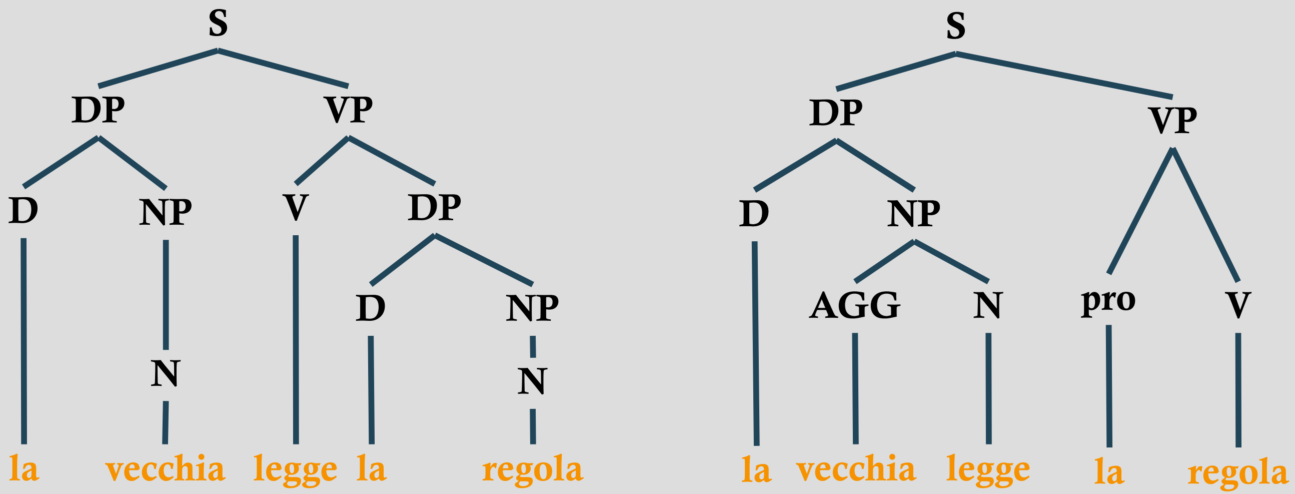
- Two main constraints:
  - Grammatical rules predicts that from a root node S certain expansion will lead to terminals;
  - The words in the sentence, indicates how the S expansions must terminate

- We can start from the root node ***S*** for generating the structure:
  Top-Down or **goal-driven** algorithm

- We can start from single words, trying to combine then in phrases up to the root node ***S***:
  Bottom-Up, or **data-driven** algorithm

Top-Down Parsing Algorithm

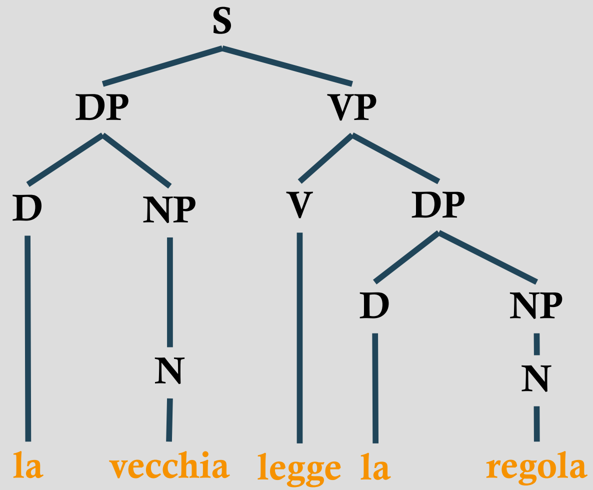
- A simple (blind) top-down algorithm explores all possible expansion of S offered by the grammar (assuming parallel expansions affects memory usage).

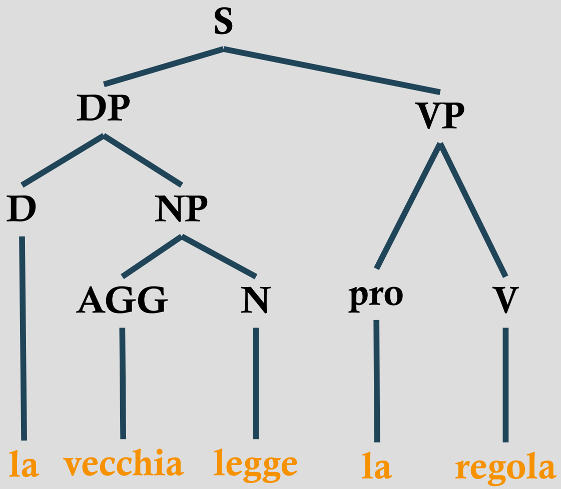


- Notice that "la regola regola la regola", "la legge legge la vecchia legge"… will be plausible analysis proposed by the Top-Down algorithm.

Bottom-Up Parsing Algorithm

- Historically, the first parsing algorithm (Yngve 55) and possibly the most common (e.g. in programming languages parsers). It starts from lexical elements, that are terminal symbols, and, phrase by phrase, up to S:





What’s better?

- **Top-Down** strategy does**n’t** loose time generating **ungrammatical trees**, but it generates sentences without considering the input till the end.

- **Bottom-Up** strategy, will be **locally consistent** with the input, but it will generate **ungrammatical phrases** unable to be rejoined under the root node **S**.

- Both blind strategies are complete, then roughly equivalent, but:
  - Consider starting from the side with the most precise (unambiguous) information
  - Explore the tree trying to be guided by the smallest possible ramification factor.

LEFT CORNER Algorithm

- Basic idea
  combination of a Top-Down strategy, filtered by Bottom-Up considerations.

- **Left-corner** rule
  - Every non-terminal category will be rewritten at some point by a word in the input
  - Then B if the «left-corner» of the A category
    IFF A →* B → α.

- Off-line table of left corner given a standard grammar:

category	**S**	**DP**	**VP**
left-corner	**D, N_{proper}, V**	**D, N_{proper}**	**aux, V**

Unresolved problems

- Left-recursion
  - A →* Aα (es. DP → DP PP)
    how do we stop?

- Ambiguity
  - PP attachment (I saw a man with the binocular)
  - coordination («papaveri e paperi rossi», red poppies and ducks)
  - exponential growth of alternatives (Church e Patil 82) with respect to the number of PPs (3 PPs up to 5 possible analyses, 6 PPs up to 469 possible analyses… 8 PPs … 4867 possible analyses!).

Unresolved problems

- Inefficiency in subtrees analysis (backtracking is not needed in certain analysis):
  - a flight from Rome to Milano at 7:00PM with a Boeing 747
  - DP → D N (ok, but incomplete…)
  - DP → D N PP (ok, but incomplete…)
  - DP → D N PP PP (ok, but incomplete…)
  - …

Dynamic Programming

- **Dynamic programming** reuses useful analyses by storing them in tables (or **charts**).

- Once **sub-problems** are resolved (sub-trees in parsing), a global solution is attempted by **merging partial solutions** together.

Dynamic Programming: Earley Algorithm

- **Earley Algorithm** (Earley 1970) is a classic example of **Top-Down**, **Parallel**, **Complete** dynamic programming approach.

- The problem complexity (remember that generalized parsing is NP-hard) is reduced to Polynomial complexity. In the worst case: $\boldsymbol{O(n^3)}$.

- **One** input pass, from **left-to-right**, partial analyses are stored in chart with *n+1* entries, with *n* equals to the input length .

Dynamic Programming: Earley Algorithm

- Each **chart** entry will include three levels of information:
  - A **subtree** corresponding to one single grammatical rule
  - the **progress** in the **completion** of the rule (we use a dot • indicating the processing step, the rule is then dubbed "**dotted rule**")
  - the position of the subtree with respect to the **input position** (two numbers indicating where the rule began and where the rule is applied now:
    e.g. DP → D • NP [0,1] the rule started at the beginning of the input (position 0) and it is waiting between the first and the second word (position 1))

Dynamic Programming: Earley Algorithm

- Three fundamental operations are combined in Earley Algorithm:
  - **Predictor**
    add new rules in the chart, representing top-down expectations in the grammar; every rule in the grammar that is an expansion of a non-terminal or pre-terminal node to the right of the dot will be added here. e.g. S → • DP VP [0,0] DP → • D NP [0,0]
  - **Scanner**
    check the input, in the expected position, and trigger an advancement when the word is recognized as belonging to the expected POS. A correct scan introduce a new rule in the next position of the chart. e.g. DP → • D NP [0,0] iff D → article, then DP → D • NP [0,1]
  - **Completer**
    when the dot reached the end of the rule, the algorithm informs the chart that at the rule starting position, the category has been recognized, hence advancing the rules with the dot to the left of the relevalt category:
    e.g. NP → AGG N • [1,3] will advance the rule in the [1] position,
    DP → D • NP [0,1] adding DP → D NP • [0,3];

Some consideration on efficiency and plausibility

- A grammar can avoid considering **space/time** limits while focusing only on descriptive adequacy;
- the **parser** should take into consideration such limits. It happens that one grammar can be **used by different parsing algorithms**.
- The adequacy of the parser can be a matter of **computational performance** or **psycholinguistic plausibility**:
  - **token transparency** (Miller e Chomsky 63) or **strict isomorphism** (is the **null hypothesis**) the parser implements exactly the derivation suggested by the grammar.
  - **type transparency** (Bresnan 78) suggests that, overall, the parsers implements **different derivations** with respect to the grammar, but overall, the same phenomena (e.g. passive constructions) are processed, globally, in a coherent way.

Some consideration on efficiency and plausibility

- **covering grammars** (Berwick e Weinberg 83, 84) parser and grammar must cover the same phenomena. But the **parser** should **be psycholinguistically plausible** or **computationally efficient** then implementing derivations that are not included in the grammar.

Minimal(ist) derivation, memory & intervention

STAGE III

Minimalist Grammars

- Stabler’s (1997) formalization of a Minimalist Grammar, **MG** (Chomsky 1995) as a 4-tuple (V, Cat, Lex, F) such that:

  V is a finite set of non-syntactic features, (P ∪ I) where

  P are phonetic features and I are semantic ones;

  Cat is a finite set of syntactic features,

  Cat = (base ∪ select ∪ licensors ∪ licensees) where

  base are standard categories {comp, tense, verb, noun ...},

  select specify a selection requirement {=x | x base}

  licensees force phrasal movement {–wh, –case ...},

  licensors satisfy licensee requirements {+wh, +case ...}

  Lex is a finite set of expressions built from V and Cat (the lexicon);

  F is a set of two partial functions from tuples of expressions to expressions : {merge, move};

Minimalist Grammars

V = *P* = {/what/, /did/, /you/, /see/},
I = {[what], [did], [you], [see]}

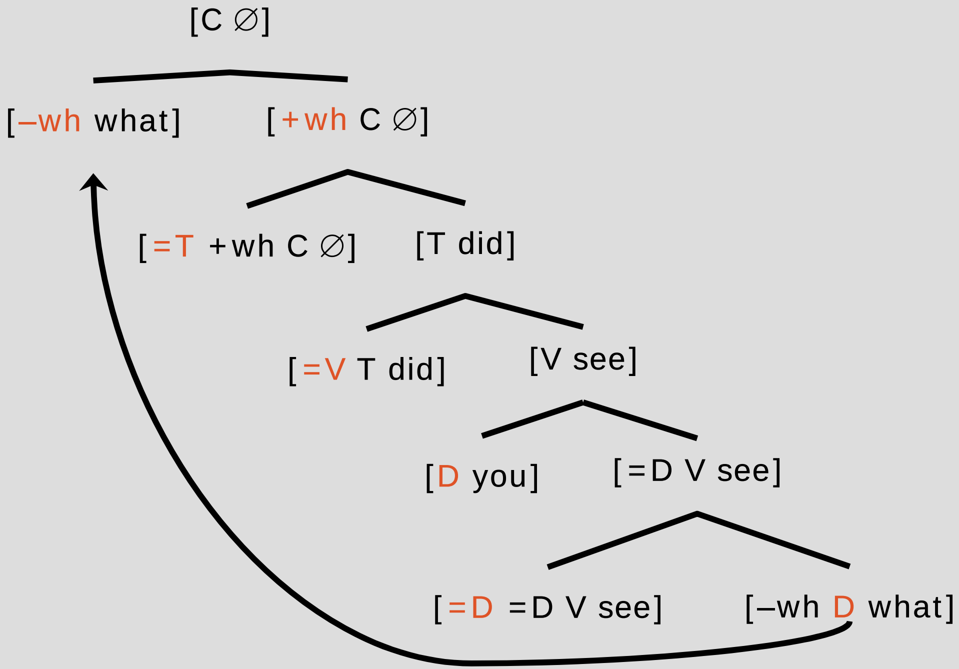
Cat = *base* = {D, N, V, T, C}
select = {=D, =N, =V, =T, =C}
licensors = {+wh}
licensees = {–wh}

Lex = { [–wh D what], [=V T did], [D you], [=D =D V see],
[=T +wh C ∅] }

F = {*merge*, *move*} such that:
merge ([=*F* X] , [*F* Y]) = $[_X$ X Y]
(“simple merge” on the right, “complex merge” on the left)
move ([+*g* X] , [W [–*g* Y]]) = $[[_X$ Y X] W, t_Y]

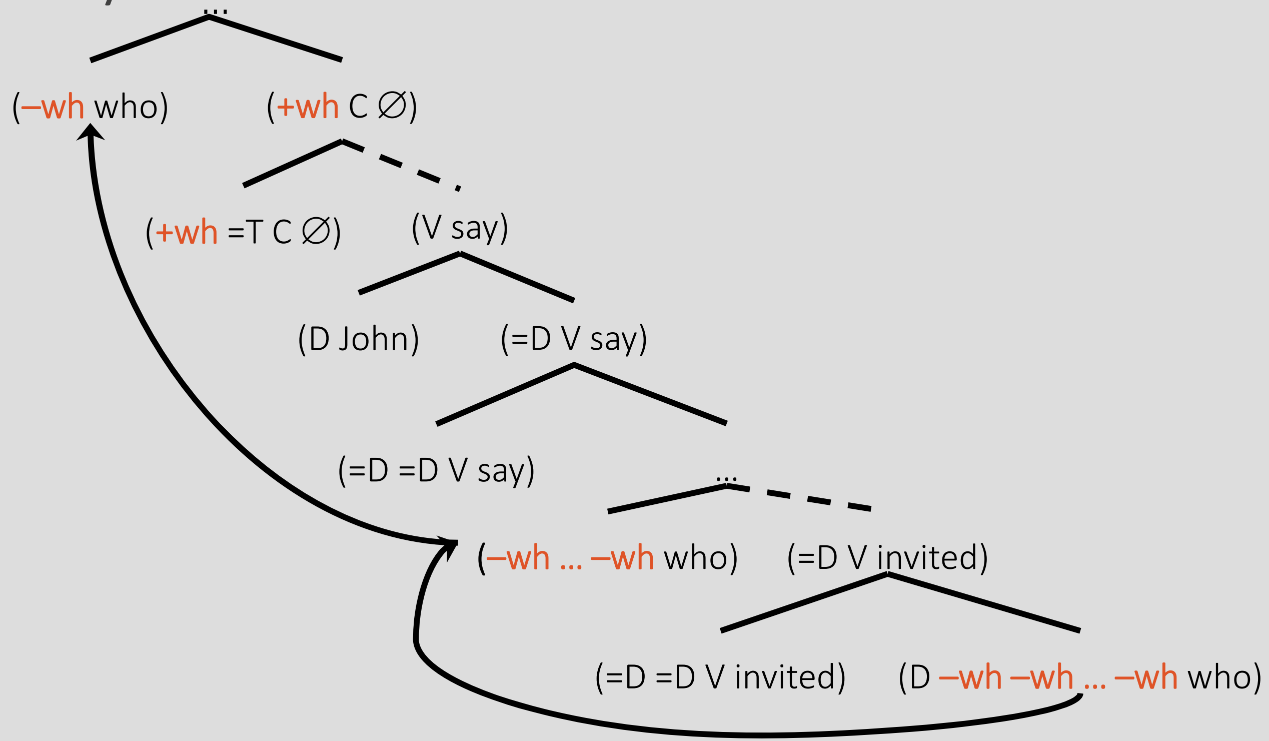
Minimalist Grammars

1. merge ([=D =D V see], [-wh D what]) → [$_{see}$ =D V see, –wh what]
2. merge ([D you], [=D V see, -wh what]) → [$_{see}$ you, [$_{see}$ V see, –wh what]]
3. merge ([=V T did], [$_{see}$ you, [$_{see}$ V see, -wh what]]) →
   ([$_{did}$ T did, [$_{see}$ you, [$_{see}$ see, –wh what]]]
4. merge ([=T +wh C ∅], [$_{did}$ T did, [$_{see}$ you, [$_{see}$ see, –wh what]]]) →
   ([$_{C}$ +wh C ∅, [$_{did}$ did, [$_{see}$ you, [$_{see}$ see, –wh what]]]])
5. move ([$_{C}$ +wh C ∅, [$_{did}$ did, [$_{see}$ you, [$_{see}$ see, –wh what]]]]) →
   [$_{C}$ What C ∅, [$_{did}$ did, [$_{see}$ you, [$_{see}$ see, t$_{what}$]]]]



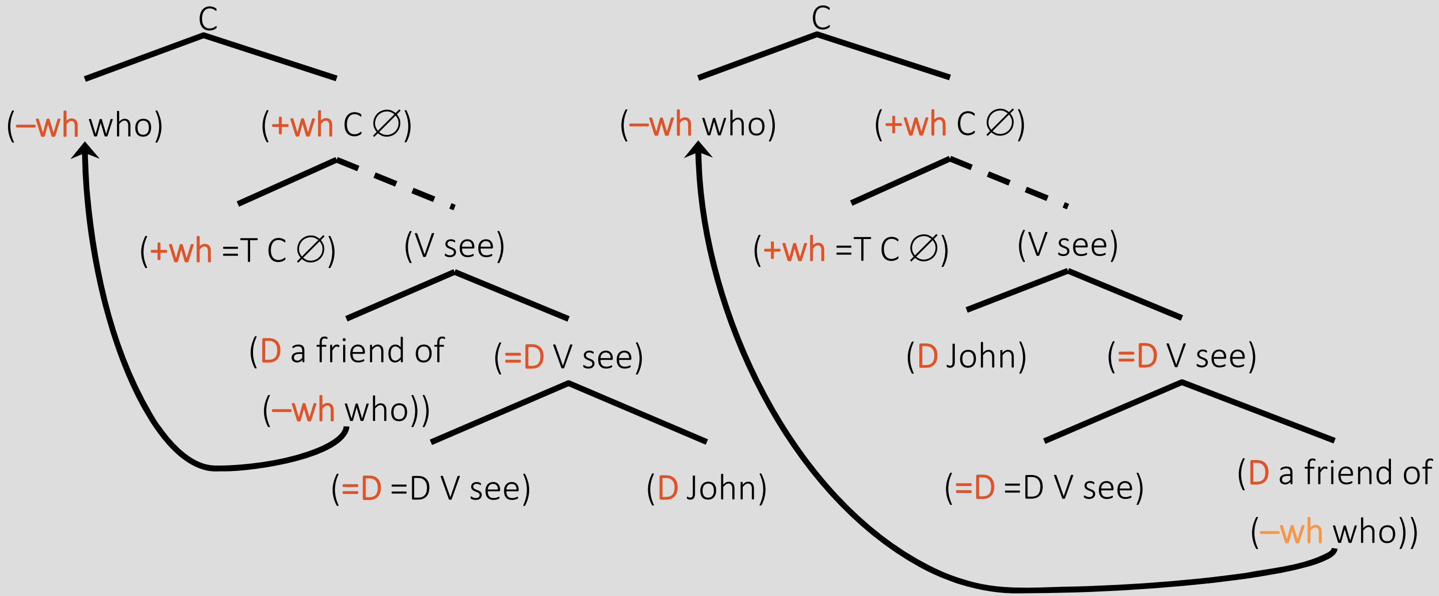
MG: problems with successive cyclicity

- *Wh-* successive cyclic movement



MG: how explaining islandhood?

- No difference in picking up an element from a **subject** or an **object** (idem for **RCs** and **Adjuncts**)



Representations vs. Derivations

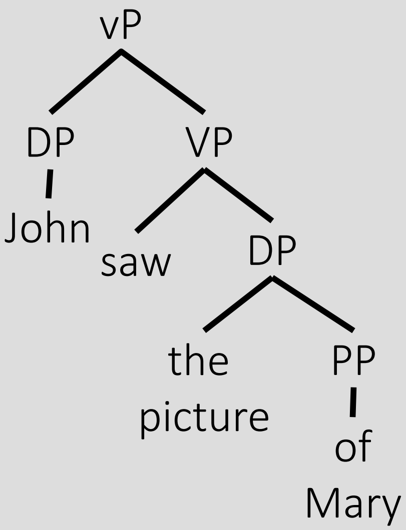
- “the computational system takes **representations** of a given format and **modifies them**” (Chomsky 1993:6)

- The **order** of **Structure Building Operation** is **abstract** with “no temporal interpretation implied” (Chomsky 1995:380)

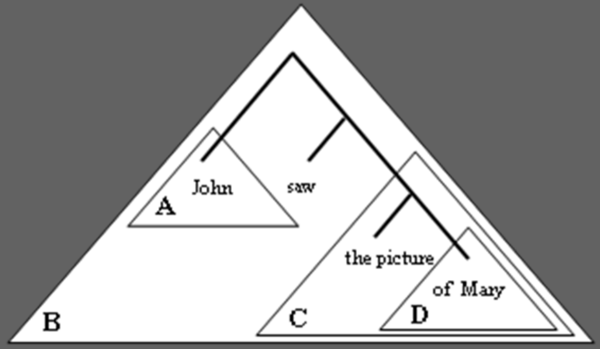
- **Derivation by Phase** (Chomsky 2005-08): a phase is a Syntactic Object built assuming Structure Building Operations (**Merge** and **Move**) over a finite set of Lexical Item (Lexical Array, aka **Numeration**) **CP** and **vP** are phases (maybe **DP**)

Derivations: some logical possibilities

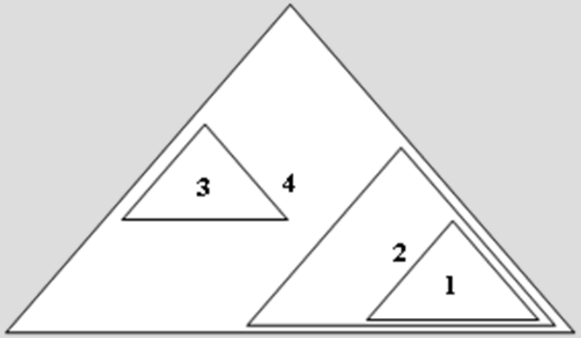
- ((John) saw ((the picture) (of Mary)))



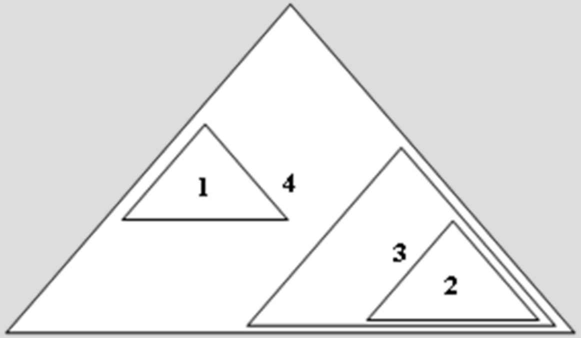
Derivations: Local Relations



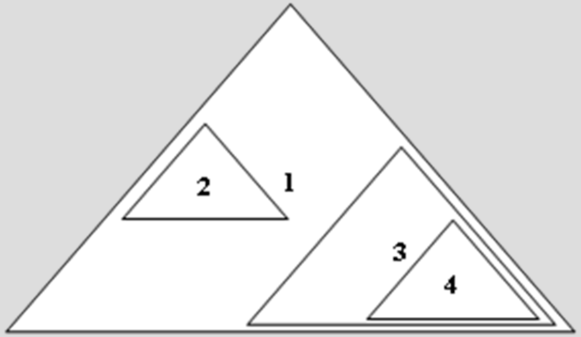
($_B$ ($_A$ John) saw ($_C$ the picture ($_D$ of Mary)))



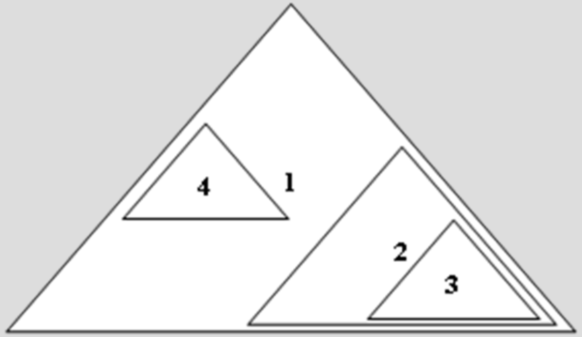
bottom-up, right left



bottom-up, left-right



top-down, left-right



top-down, right-left

Processing Object Relatives (ORs)

- Bever (1970)
  double embedding is not always nearly impossible to process (Miller & Chomsky 1963):

  - The reporter the politician the commentator met trusts said the president won't resign.
  - The reporter everyone I met trusts said the president won't resign.

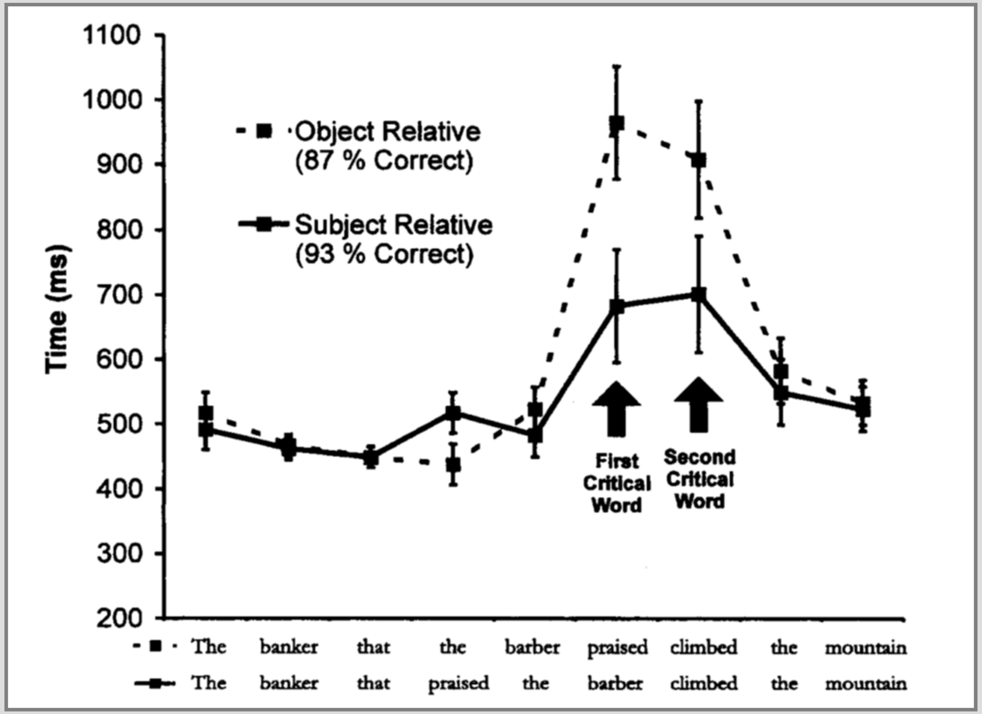
Processing Object Relatives (ORs)

- Gordon, Hendrick & Johnson (2001)
  working memory request is evaluated by studying **reading time** (**RT**) and **comprehension accuracy** in **self-paced reading** experiments comparing critical regions of various kinds of **Relative Clauses**:

- **Experiment 1** (materials): **SRs** (a) and **ORs** (b)
  - The banker [that _ praised the barber] climbed the mountain
  - The banker [that the barber praised _] climbed the mountain

Processing Object Relatives (ORs)

- Gordon et al. (2001) - **Experiment 1** (results)



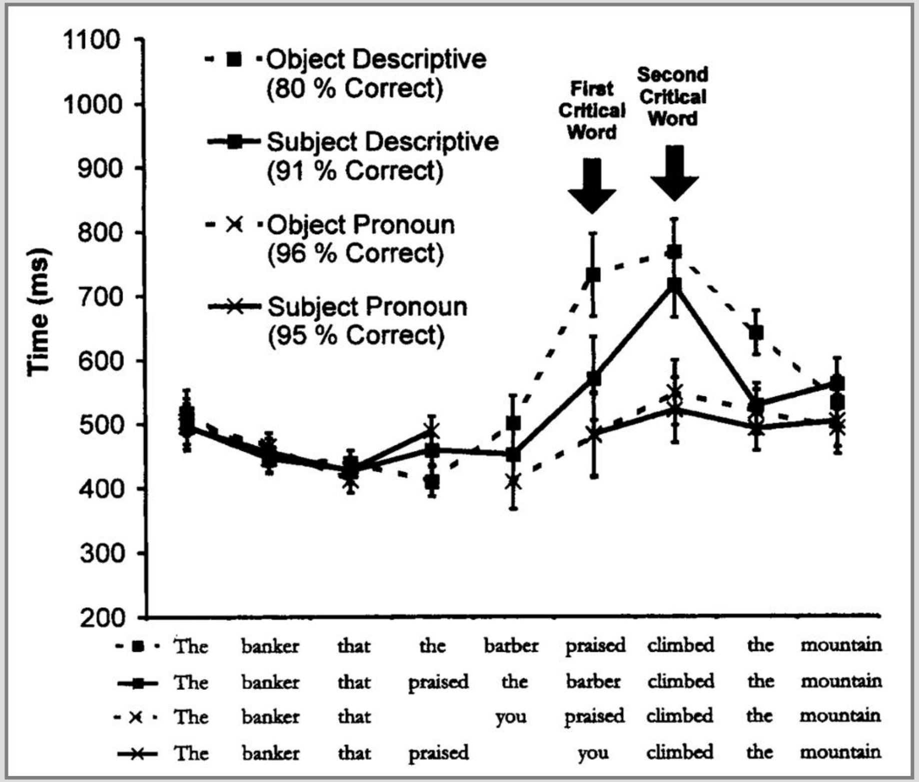
Processing Object Relatives (ORs)

- Gordon et al. (2001) - **Experiment 2**
  complexity can be mitigated by varying the RC Subject typology (reading time (**RT**) and comprehension accuracy in self-paced reading experiments are tested, as before):

- **Experiment 2** (materials): DP (a) vs. Pro (b)
  - The banker [that the barber praised _] climbed the mountain
  - The banker [that you praised _] climbed the mountain

Processing Object Relatives (ORs)

- Gordon et al. (2001)
  Experiment 2 (results)

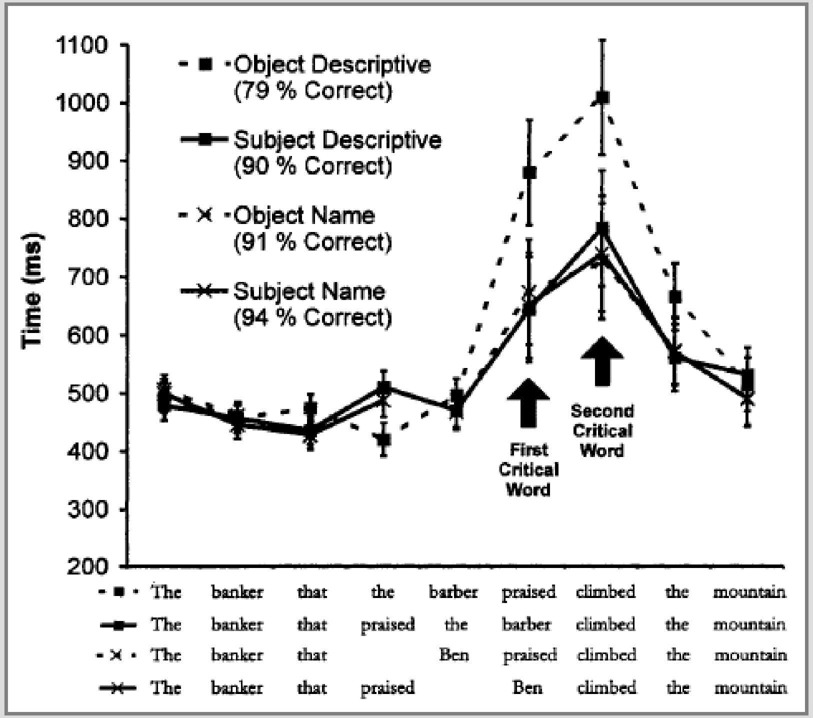


Processing Object Relatives (ORs)

- Gordon et al. (2001) - **Experiment 3** (materials): DP (a) vs. proper names (b)
  - The banker [that the barber praised _] climbed the mountain
  - The banker [that Ben praised _] climbed the mountain

Processing Object Relatives (ORs)

- Gordon et al. (2001)
  Experiment 3 (results)

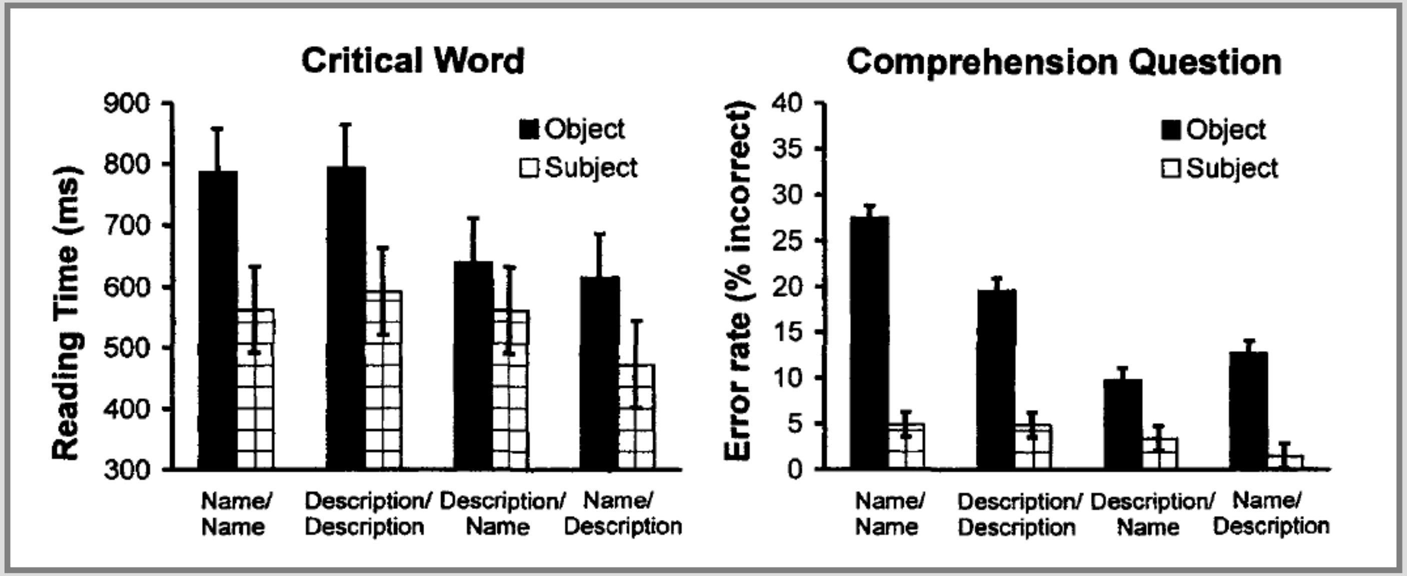


Processing Object Clefts

- Gordon et al. (2001) **- Experiment 4** (materials):
  Subject vs. Object Clefts X DP vs. proper names
  - It was the banker that the lawyer saw _ in the parking lot
  - It was the banker that Bill saw _ in the parking lot
  - It was John that the lawyer saw _ in the parking lot
  - It was John that Bill saw _ in the parking lot

Processing Object Clefts

- Gordon et al. (2001) - **Experiment 4** (results):



Explaining complexity

- **Role-determinant** accounts (MacWhinney & Pleh 1988)
  - Double role for the RC head: **subject** in the matrix sentence, **object** in the RC:
    The banker [that the barber praised _] climbed the mountain (OR)
- **Memory-load** accounts (Ford 1983, MacWhinney 1987, Wanner & Maratsos 1978)
  - The RC head must be **kept in memory longer** in OR before being integrated:

    The banker [**that praised** the barber] climbed … (SR)
    The banker [**that the barber praised** _] climbed … (OR)

Explaining complexity

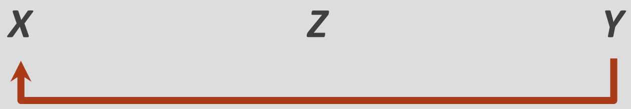
- **Linguistic Integration Cost** (Gibson 1998:12-13)
  - Processing difficulty is proportional to the **distance** expressed in terms of number of **intervening discourse referents**, following a "**referentiality hierarchy**":
    descriptions > (short) names > referential pronouns > indexical pronouns

- **Similarity based accounts** (Gordon et al. 2001)
  - Having **two DPs of the same kind** stored in **memory** makes the OR more complex than SR. This models memory interference during encoding, storage and retrieval (Crowder 1976)

Explaining complexity

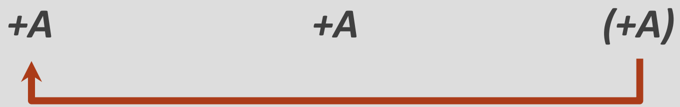
- More on **Similarity based accounts** (Gordon et al. 2001)
  - It might be able to explain why SR vs. OR asymmetry disappears with RC subject pro/proper names (those DPs are legal heads only for clefts)

- **Intervention effects**
  (Grillo 2008, Friedmann et al. 2009, Rizzi 1990)
  - Processing difficulty is proportional to the number and kind of relevant features shared between the moved item and any possible intervener:



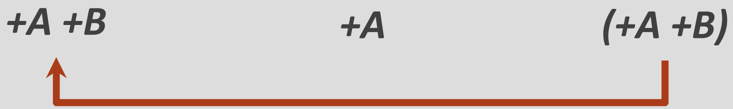
Explaining complexity

- More on **Intervention effects** (Friedmann et al. 2009)
  - **Identity** (bad for adults, bad for children)



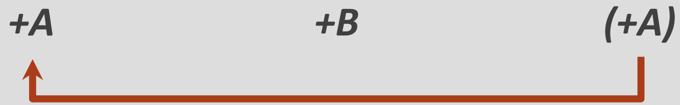
+A *+A* *(+A)*

  - **Inclusion** (ok for adults, bad for children)



+A +B *+A* *(+A +B)*

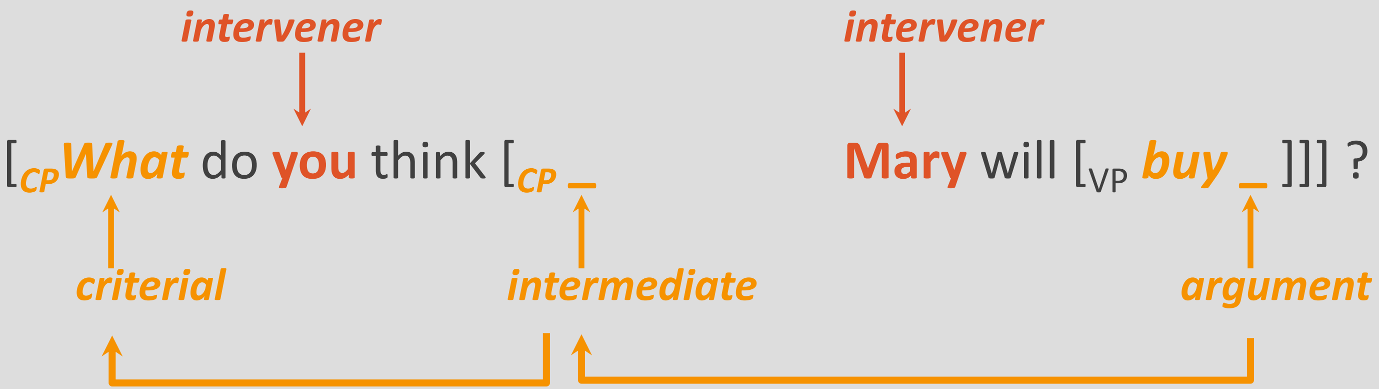
  - **Disjunction** (ok for adults, ok for children)



+A *+B* *(+A)*

Kinds of non-local dependencies

Long distance *Wh*- dependencies



intervener

intervener

[$_{CP}$***What*** do **you** think [$_{CP}$ _ **Mary** will [$_{VP}$ ***buy*** _]]] ?

criterial

intermediate

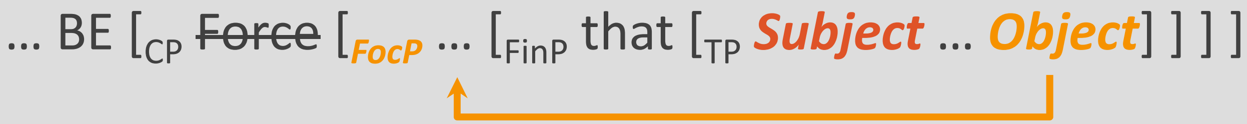
argument

Kinds of non-local dependencies

Object Clefts

- In **Object Clefts** (**OCs**), the **copula** selects a truncated CP (Belletti 2008):

It is [$_{FocP}$ ***an ice cream*** that [$_{TP}$ ***Mary*** will ***buy _***]]



… BE [$_{CP}$ ~~Force~~ [$_{FocP}$ … [$_{FinP}$ that [$_{TP}$ ***Subject*** … ***Object***]]]]

Comparing Object Clefts

- Warren & Gibson (2005) - **Experiment** (materials):
  definite descriptions vs. **proper names** vs. **pronouns**

a. It was **the banker** that **the lawyer** **avoided _** at the party

b. It was **the banker** that **Dan** **avoided _** at the party

c. It was **the banker** that **we** **avoided _** at the party

d. It was **Patricia** that **the lawyer** **avoided _** at the party

e. It was **Patricia** that **Dan** **avoided _** at the party

f. It was **Patricia** that **we** **avoided _** at the party

g. It was **you** that **the lawyer** **avoided _** at the party

h. It was **you** that **Dan** **avoided _** at the party

i. It was **you** that **we** **avoided _** at the party

Comparing Object Clefts

- Warren & Gibson (2005) - results (Tessa Warren P.C.)
  D = definite description (e.g. **the banker**)
  N = proper names (e.g. **Dan**)
  P = pronouns (e.g. **you**)

condition	**D-D**	**D-N**	**D-P**	**N-D**	**N-N**	**N-P**	**P-D**	**P-N**	**P-P**
Read. time (SE) ms	**365** (19)	**319** (12)	**306** (14)	**348** (18)	**347** (21)	**291** (14)	**348** (18)	**311** (15)	**291** (13)

Predicting reading times (rt) with intervention-based accounts

- Assuming that **Definite Description** = {+NP, N}, **Proper Names** = {+NP, NProper}, **pro** = {} (Belletti & Rizzi 2013),
Intervention effects are predicted to be stronger in matching **D-D** and **N-N** condition (against memory-load accounts), while **P-P** is expected not to be critical (because of the +NP absence):

condition	D-D	D-N	D-P	N-D	N-N	N-P	P-D	P-N	P-P
Read. time (SE) ms	**365** (19)	**319** (12)	**306** (14)	**348** (18)	**347** (21)	**291** (14)	**348** (18)	**311** (15)	**291** (13)
prediction	hard	?	easy	?	hard	easy	easy	easy	easy

Some problems with the intervention-based account

- Features triggering movement are those relevant for intervention (Friedmann et al. 2009:82), but:
  - "**+R**" feature causing Object movement in ORs (or "**+Foc**" in OCs) is not present on Subject;
  - Neither the "**lexical restriction**" nor **phi-features** trigger any movement in **ORs** or **OCs**
  - The "**lexical restriction**" should be not accessible at the **edge of the DP**, where features triggering movement should be located (but see Belletti & Rizzi 2013, next slide)
  - Why slow-down is observed at **verb segment**?

Some problems with the intervention-based account

- Belletti & Rizzi 2013:
  - Evidence that lexically restricted wh-items occupy different positions in the left periphery (Munaro 1999):

a. Con **che tosat** à-tu parlà?
with which boy did you speak?

b. Avé-o parlà de **chi**?
Have you spoken of whom?

Feature Retrieval Cost (FRC)
Why do we need it? (a summary)

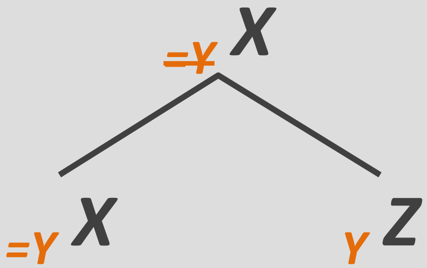
- An “integration cost” (cf. Gibson 1998) is **not enough**
  - È **il bambino** che ***il signore*** ha salutato …
  - È **Luigi** che ***Gianni*** ha salutato …

- **Intervention-based** accounts are **not “gradable”** (no quantitative, precise, measurements)

- **Bottom-Up** standard theories **do not make clear predictions on processing**: they predict **what** creates complexity, but not **when**, **why** and **how** exactly in **parsing** and **generation**?

Processing-friendly Phase-based Minimalist Grammar

- Common restriction on **Merge**:
  - Given two lexical items [$_{=Y}$ X] and [$_{Y}$ Z] such that X selects Z, then:

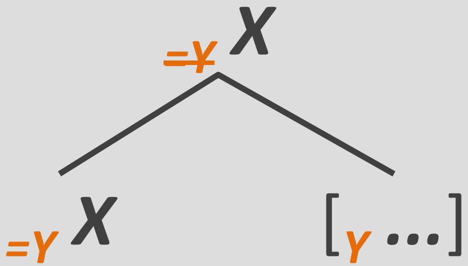


  - [$_{=Y}$ X] is processed before Y
  - When [$_{=Y}$ X] is processed, an expectation for [$_{Y}$ ...] is created

Processing-friendly
Phase-based Minimalist Grammar

- A **Phase** is the minimal computational domains within which a selection requirement must be satisfied:

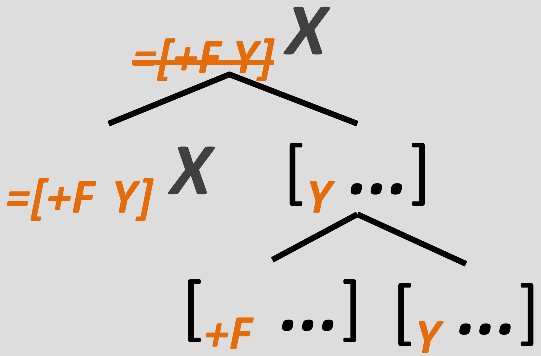
  - Given a lexical item [$_{=Y}$ X], [Y ...] is the selected phase:



  - Merge reduces to lexical selection (or unification)
    (e.g. [$_{Y}$ Z] insertion)

Processing-friendly Phase-based Minimalist Grammar

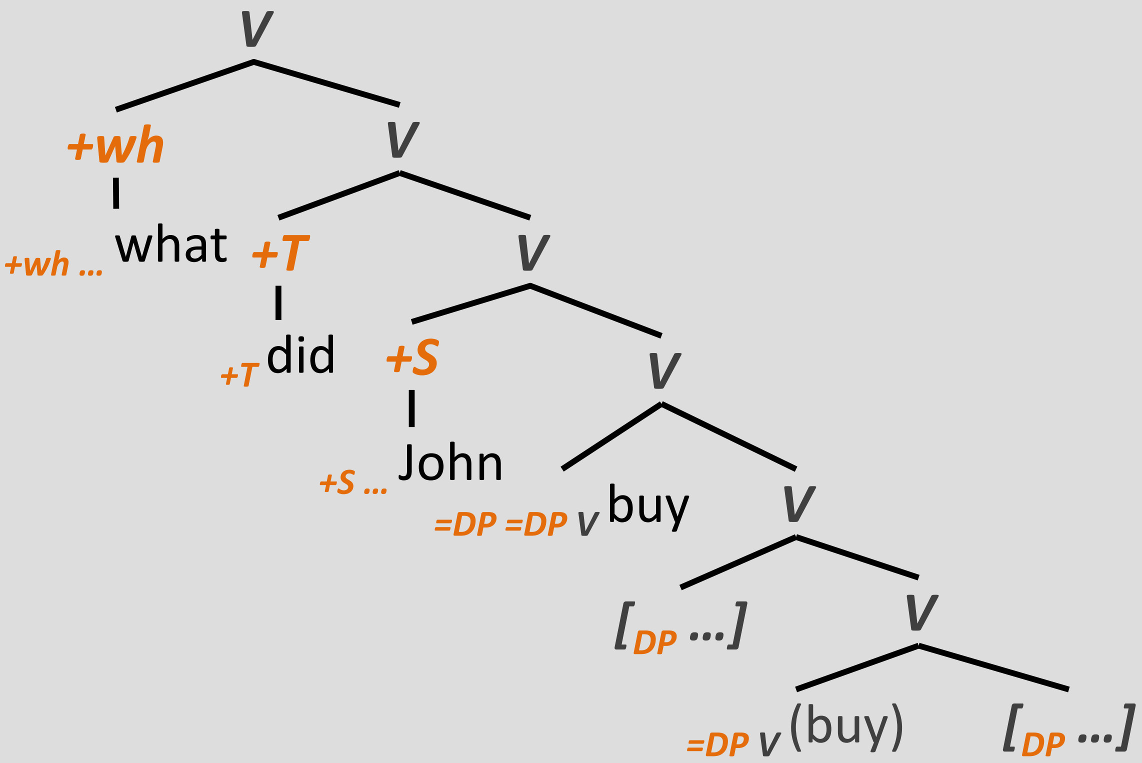
- If we assume that selection can include both functional features (+F) and lexical features (Y) at the same time, a Phase becomes a subtree to be expanded:
  - Given a lexical item [$_{=[+F\ Y]}$ X], [$_{+F\ Y}$ ...] is the selected phase:



  - [$_{+F\ Y}$ ...] is an extended projection of a lexical category Y (e.g. a DP is an extended projection of N, i.e. [+D N])

Processing-friendly Phase-based Minimalist Grammar

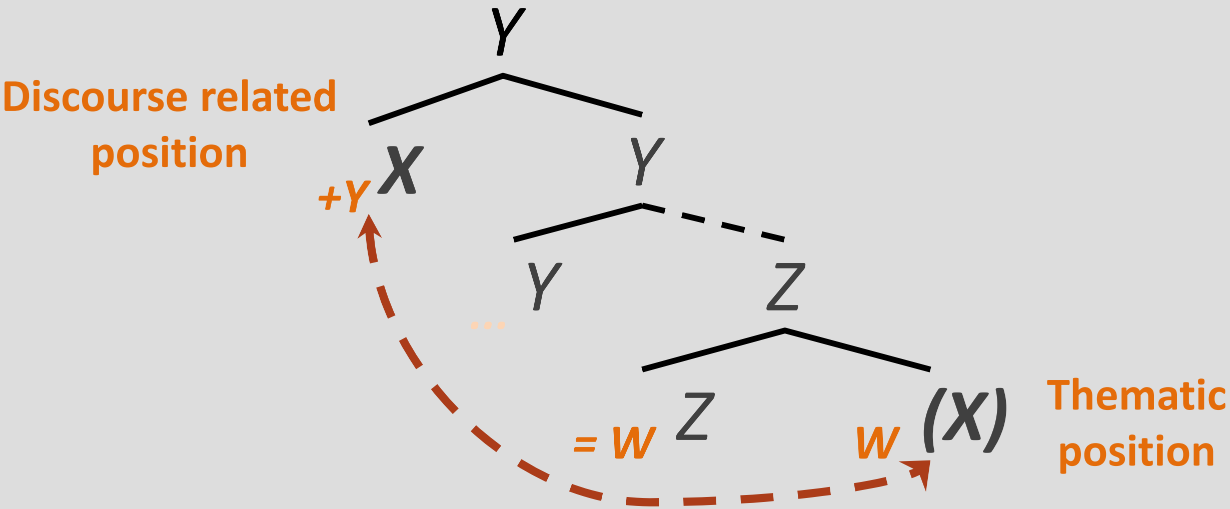
- Both a declarative sentence [***+S*** ***+T*** ***V***] and a ***wh-*** question [***+wh*** ***+T*** ***+S*** ***V***] are phases (i.e. extended projections of a V head)



- [$_{+wh\ \ldots}$ what], [$_{+T}$ did], [$_{+S\ \ldots}$ John], [$_{=DP\ =DP\ V}$ buy]

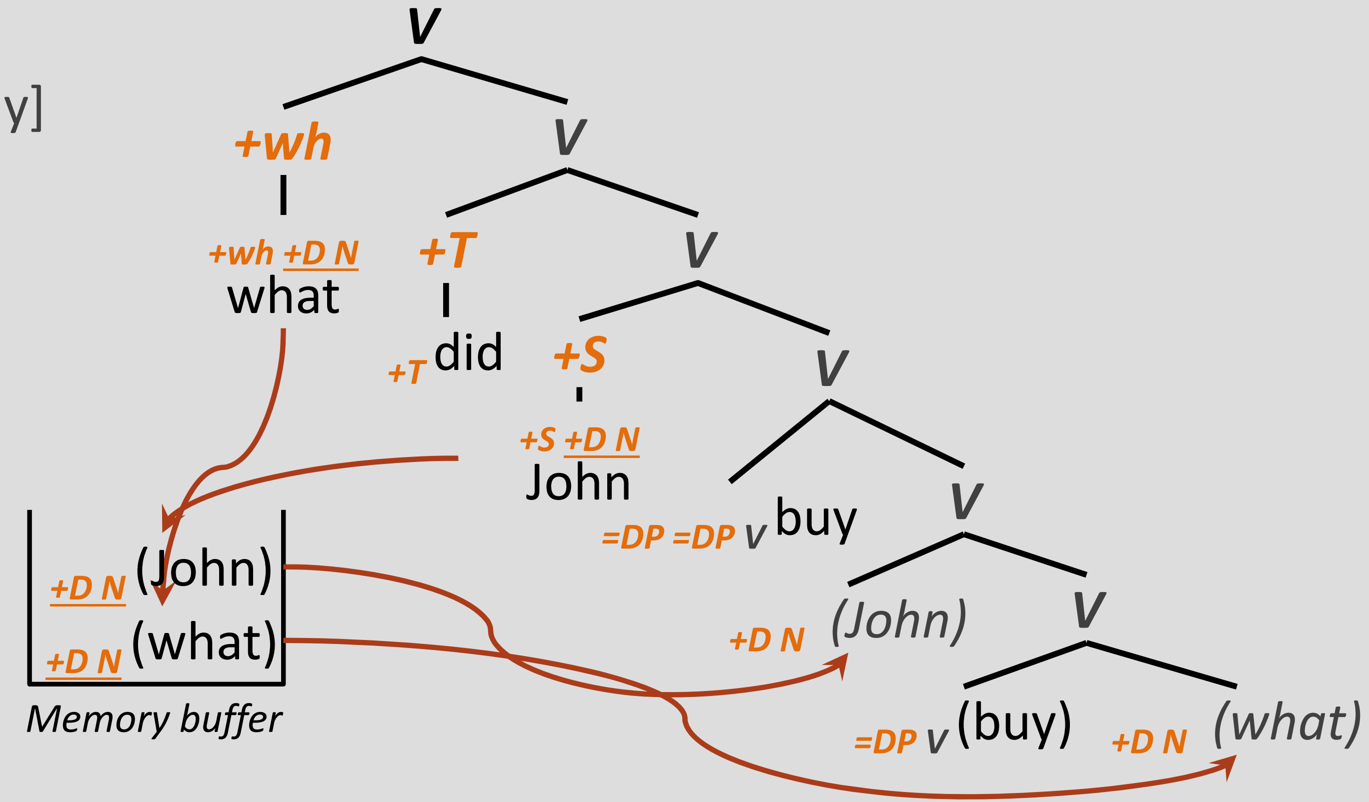
Processing-friendly Phase-based Minimalist Grammar

- Common trigger for Move:
  - An item $[_{+Y \ldots W}\ X]$, in a given structure, must be moved if it can not be fully interpreted in its insertion position:



Processing-friendly Phase-based Minimalist Grammar

- [$_{\text{+wh +D N}}$ what], [$_{\text{+T}}$ did], [$_{\text{+S +D N}}$ John], [$_{\text{=DP =DP V}}$ buy]

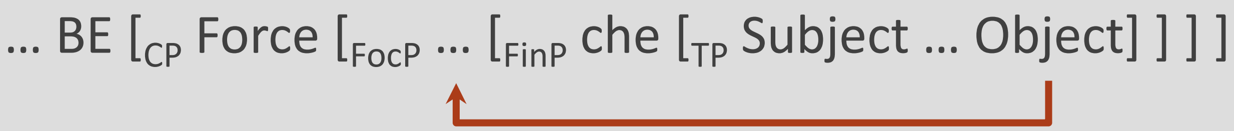


Processing-friendly Phase-based Minimalist Grammar

- The derivation unfolds **Top-Down** and (as a consequence) **Left-Right**
- **Unexpected features** trigger **movement**
- **Phases** restrict the domain in which a **non-local dependency** must be satisfied
- **Last-In-First-Out memory** buffer, as a first approximation, is used to store and retrieve items for **non-local dependencies** (memory buffer must be empty at the end of the derivation)
- The order in which phases are expanded makes a difference: the last selected phase has a special status (**sequential phase**) while phases that are not the last selected ones (e.g. phases that results from expansion of functional features) qualifies as **nested phases** (Bianchi & Chesi 2006)

Deriving OCs Top-Down

- In Object Clefts (OCs), the copula selects a truncated CP (Belletti 2008):

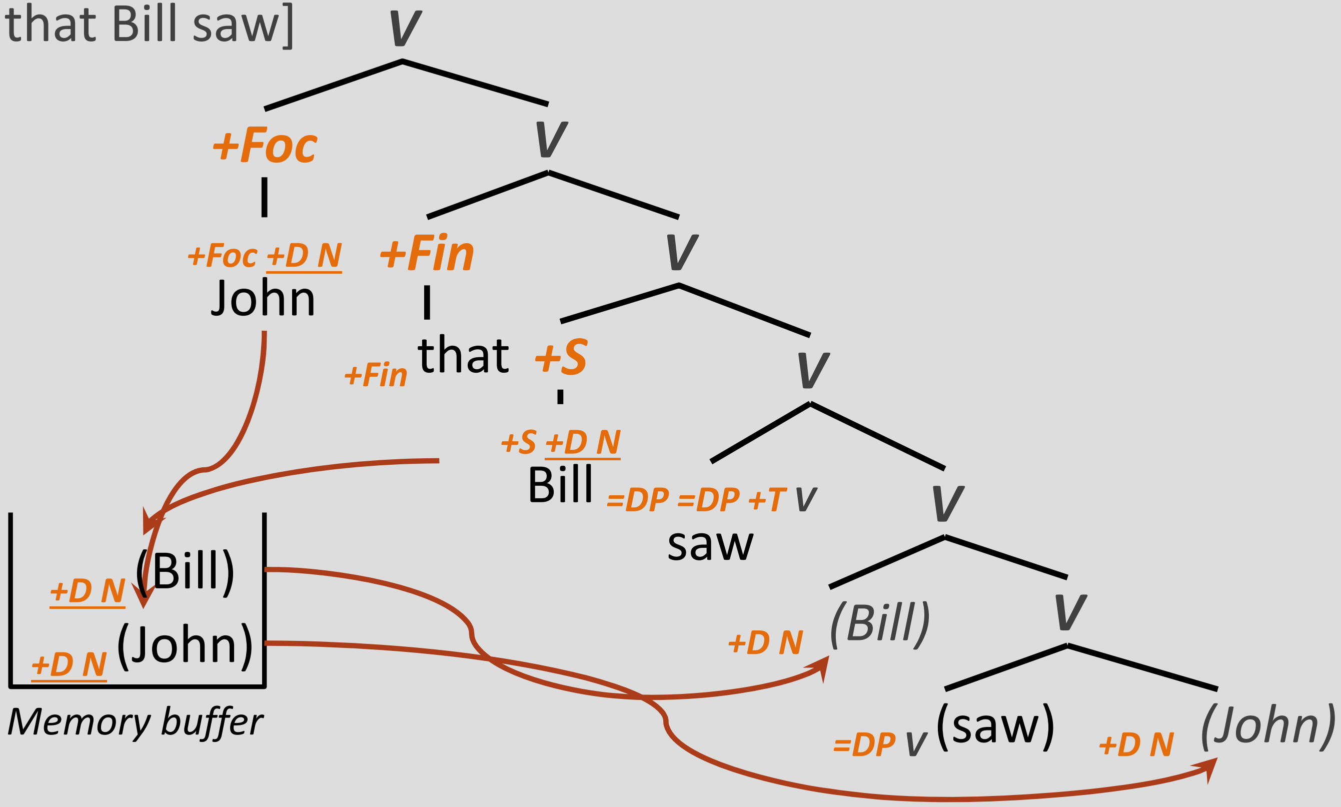


... BE $[_{\text{CP}}$ Force $[_{\text{FocP}}$ ... $[_{\text{FinP}}$ che $[_{\text{TP}}$ Subject ... Object]]]]

- Reduced CP (CPr) = [**+Foc +Fin +S +T** **V**]

Deriving OCs Top-Down

- It [… $_{=CPr}$ … was] [$_{CPr}$ John that Bill saw]



Cue-based retrieval and intervention

- **interference** is the major constraint on accessing information in memory (Anderson & Neely 1996; Crowder 1976; see Nairne 2002 for a review).
- the locus of the interference effect is at **retrieval**, with little or no effect on memory encoding or storage (Dillon & Bittner 1975; Gardiner et al. 1972; Tehan & Humphreys 1996)
- **Content-adressable memory** (e.g. memory load paradigm, Van Dyke & McElree 2006), no exhaustive search, no delay
- Search of **Associative Memory** (**SAM**) model (Gillund & Shiffrin 1984)

  $$P(I_i|Q_1, \ldots , Q_n) = \frac{\prod_{j=1}^{m} S(Q_j, I_i)^{w_j}}{\sum_{k=1}^{N} \prod_{j=1}^{m} S(Q_j, I_k)^{w_j}}$$

On DP features (and structure)

- Elbourne (2005)
  [[THE *i*] NP]

- Zamparelli (1995-2000)
  $[_{SDP}$ Strong QP $[_{PDP}$ Week QP $[_{KIP}$ (Restrictive Adj) $[_{NP}$ Noun]]]]

- Longobardi (1994-2005), a rough summary:

Definite Descriptions	$[_{D}$ the $[_{N}$ man]]
Proper Names	$[_{D}$ John$_{i}$ $[_{N}$ t$_{i}$]]
Pronouns	$[_{D}$ you $[_{N}$ $\varnothing$]]

Relevant DP features
Definite Descriptions & Proper Names

- Both proper names and common nouns have category N

N in situ **(common nouns)**	***N-to-D raising***
Il mio Gianni (Il mio amico)	*mio Gianni
La sola Maria (la sola amica)	Maria sola (*l'amica sola)

- Two different kinds of N: $\mathbf{N_{proper}}$, $\mathbf{N_{(common)}}$

Relevant DP features
On D and Pronouns

- Both **determiners** and **personal pronouns** introduce a “**referential pointer**” to an individual constant or variable in the domain of discourse

- **Pro** are **NP-ellipsis licensors** (they can be used as determiners «we italians»):
  [$_D$ noi [$_N$ ~~italiani~~]]
  (**D** introduces an *index*, that bounds a variable predicated in N)

- (More) features on **pro**:
  - **1st** and **2nd** person (highly accessible referents) vs. **3rd** person (**default person**, context-determined referent)
  - **case**

Relevant DP features

- **Definite descriptions**: **{D, N}**

- **Proper names**: **{D, N_{prop}}**

- **Pronouns**: **{D, case, pers}**

Feature Retrieval Cost (FRC) metrics at work

- Cost function (at ***X*** given ***m_x*** items to be retrieved from memory)

- CFRC(x) = $\prod_{i=1}^{m_x} \frac{(1+nF_i)^{m_i}}{(1+dF_i)}$

  - ***m*** = number of items stored in memory at retrieval
  - ***nF*** = new features to be retrieved from memory
  - ***dF*** = number of distinct cued features (e.g. agreement and case features probed by the verb)

Feature Retrieval Cost (FRC) metrics at work

$$C_{FRC}(x) = \prod_{i=1}^{m_x} \frac{(1+nF_i)^{m_i}}{(1+dF_i)}$$

- ***D-D*** matching

  it was **the lawyer**$_{\{D, N\}}$ who **the businessman**$_{\{D, N\}}$ *avoided…*

  C$_{FRC}$ (*avoided*) = **27**

  that is **9 · 3**:
  9 for retrieving **the businessman**,
  since ***nF***=**2** (***D*** and ***N*** count as one), ***m*=2** because two DPs are in memory at this time, and ***dF*=0** because no feature is cued by the verb distinguishing one DP from the other; 3
  1 for retrieving **the lawyer**,
  since ***nF*=2** (D and N are new now), ***m*=1** and ***dF*=0**

Feature Retrieval Cost (FRC) metrics at work

$$C_{FRC}(x) = \prod_{i=1}^{m_x} \frac{(1+nF_i)^{m_i}}{(1+dF_i)}$$

- ***N*-*N*** matching

  it was **Dan**$_{\{D,\ N\_prop\}}$ who **Patricia**$_{\{D,\ N\_prop\}}$ *avoided…*

  C$_{FRC}$ (*avoided*) = **18**

  that is **9 · 2**:
  9 for retrieving **Dan**,
  nF*=2** (even though ***D should be contextually salient, being two proper names presents, the same ***D***, i.e. a co-referential index, cannot be sufficient to distinguish them, then an extra cost must be paid here as in the ***D-D*** condition), ***m*=2**, ***dF*=0**;
  2 for retrieving **Patricia**,
  since ***nF*=0** (just N is new since the determiner is now contextually salient and unique, *m*=1 and *dF*=0)***m*=1** and ***dF*=0**

Feature Retrieval Cost (FRC) metrics at work

$$C_{FRC}(x) = \prod_{i=1}^{m_x} \frac{(1+nF_i)^{m_i}}{(1+dF_i)}$$

- ***P*-*P*** matching

  it was **you**$_{\{D,\ pers\_II,\ case\}}$ who **we**$_{\{D,\ pers\_I,\ case\_nom\}}$ *avoided…*

  C$_{FRC}$ (*avoided*) = **4**

  that is **2 · 2**:

  2 for the **we**, ***nF*=1**, ***m*=2** and ***dF*=1** (**number**, **person** and **case** mismatches are always present; **case** is cued by the verb),

  2 for retrieving **you**, ***nF*=1**, ***m*=1** and ***dF*=0** for the object pronoun

Feature Retrieval Cost (FRC) metrics at work

$$C_{FRC}(x) = \prod_{i=1}^{m_x} \frac{(1+nF_i)^{m_i}}{(1+dF_i)}$$

- ***D*-*N*** matching

  it was **the lawyer**$_{\{D,\ N\}}$ who **Patricia**$_{\{D,\ N\_prop\}}$ *avoided…*

  C$_{FRC}$ (*avoided*) = **12**

  that is **4 · 3**:
  4 for **Patricia**, *nF*=1, that is N, since D is contextually salient, m=2, dF=0,
  3 for retrieving **the lawyer** (nF=2, m=1, nF=0)

Feature Retrieval Cost (FRC) metrics at work

$$C_{FRC}(x) = \prod_{i=1}^{m_x} \frac{(1+nF_i)^{m_i}}{(1+dF_i)}$$

- ***D***-***P*** condition

  it was **the lawyer**$_{\{D, N\}}$ who **we**$_{\{D, pers\_I, case\_nom\}}$ *avoided…*

  C$_{\textbf{FRC}}$ **(*avoided*)** = **6**

  that is **2 · 3**:

  2 for retrieving **we** (*nF*=1 even if deictic pronouns are contextually salient, the correct person must be retrieved, *m*=2, *dF*=1 since a distinct case on pronouns is cued by the verb),

  3 for retrieving **the lawyer** (*nF*=2, *m*=1, *nF*=0)

Feature Retrieval Cost (FRC) metrics at work

$$C_{FRC}(x) = \prod_{i=1}^{m_x} \frac{(1+nF_i)^{m_i}}{(1+dF_i)}$$

- ***P*-*D*** condition

  it was **you**$_{\{D,\ pers\_II,\ (case)\}}$ who **the businessman**$_{\{D,\ N\}}$ *avoided*…

  C$_{\mathbf{FRC}}$ **(*avoided*)** = **18**

  that is **9 · 2**:
  9 for the **the businessman** (*nF*=2, *m*=2, *dF*=0);
  2 for retrieving **you** (*nF*=1, *m*=1, *dF*=0);

Feature Retrieval Cost (FRC) metrics at work

- The complete prediction set:

condition	D-D	D-N	D-P	N-D	N-N	N-P	P-D	P-N	P-P
Read. time (SE) ms	**365** (19)	**319** (12)	**306** (14)	**348** (18)	**347** (21)	**291** (14)	**348** (18)	**311** (15)	**291** (13)
prediction log(FRC)	1,43	1,08	0,78	1,26	1,26	0,60	1,26	0,90	0,69

Feature Encoding Cost (FEC)

- **Feature Encoding Cost** (***FEC***) is a numerical value associated to each new item merged that is proportional to the number of new relevant features integrated in the structure:

$$FEC(x) = \sum_{i=1}^{n} eF_i$$

- eF is the cost of each new relevant feature to be encoded at *x*.
- For simplicity $\boldsymbol{eF}$ = **1** for a **new categorial feature** introduced (e.g. 1 for D and 1 for N), 2 for a **duplication** of the same lexical category requiring structural integration (i.e. 2 for the second N both in D_1-D_2 and N_1-N_2), 0 otherwise.

Feature Encoding Cost (FEC)

	object$_{focalized}$	***subject***	***verb***	***spill-over***	***condition***
a.	It was *(1)* **the banker** *(2)*	that *(1)* **the lawyer** *(3)*	**avoided** _	*(2)* at the party *(3)*	*[D$_1$-D$_2$]*
b.	It was *(1)* **the banker** *(2)*	that *(1)* **Dan** *(1)*	**avoided** _	*(2)* at the party *(3)*	*[D$_1$-N$_2$]*
c.	It was *(1)* **the banker** *(2)*	that *(1)* **we** *(0)*	**avoided** _	*(2)* at the party *(3)*	*[D$_1$-P$_2$]*
d.	It was *(1)* **Patricia** *(1)*	that *(1)* **the lawyer** *(2)*	**avoided** _	*(2)* at the party *(3)*	*[N$_1$-D$_2$]*
e.	It was *(1)* **Patricia** *(1)*	that *(1)* **Dan** *(2)*	**avoided** _	*(2)* at the party *(3)*	*[N$_1$-N$_2$]*
f.	It was *(1)* **Patricia** *(1)*	that *(1)* **we** *(0)*	**avoided** _	*(2)* at the party *(3)*	*[N$_1$-P$_2$]*
g.	It was *(1)* **you** *(0)*	that *(1)* **the lawyer** *(2)*	**avoided** _	*(2)* at the party *(3)*	*[P$_1$-D$_2$]*
h.	It was *(1)* **you** *(0)*	that *(1)* **Dan** *(1)*	**avoided** _	*(2)* at the party *(3)*	*[P$_1$-N$_2$]*
i.	It was *(1)* **you** *(0)*	that *(1)* **we** *(0)*	**avoided** _	*(2)* at the party *(3)*	*[P$_1$-P$_2$]*

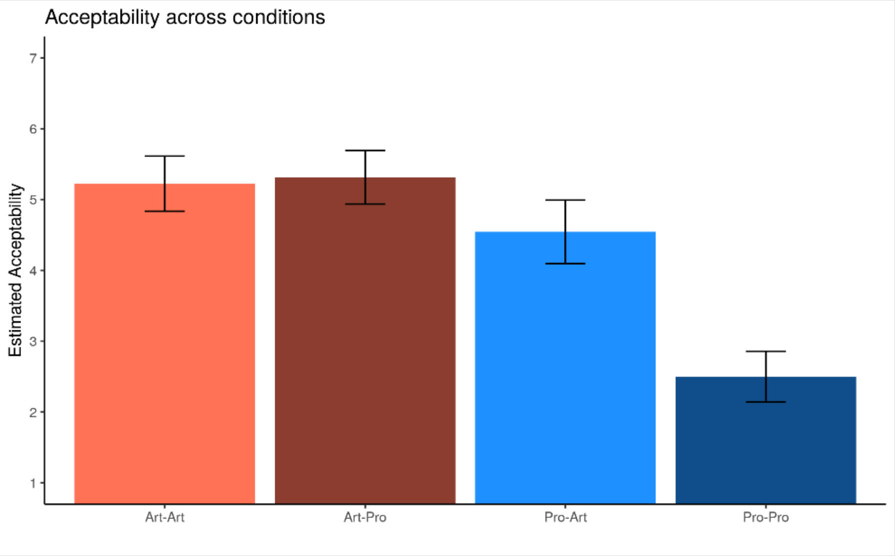
Chesi & Canal (2019)

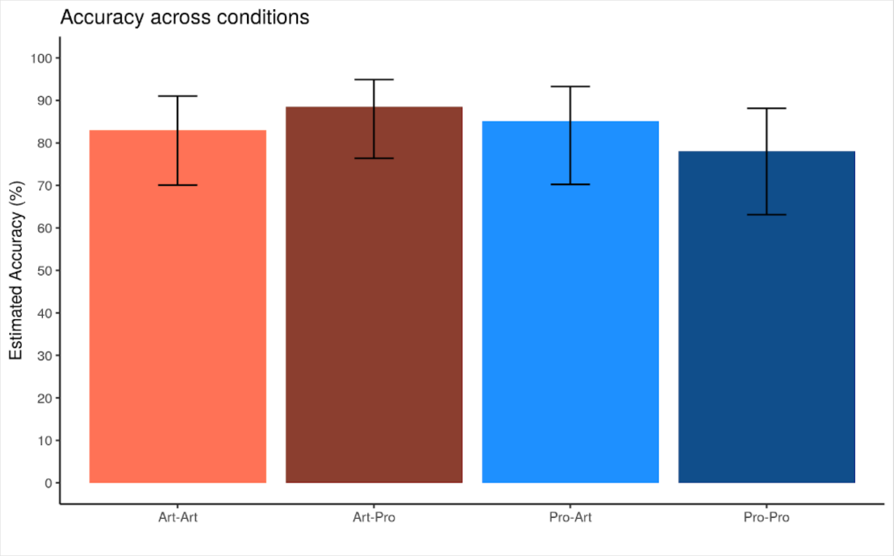
	***object**$_{focalized}$*	***subject***	***verb***	***spill-over***	***condition***
a.	Sono [**gli** architetti]$_i$ che	[**gli** ingegneri]	**hanno**	consultato _$_i$ prima di iniziare i lavori.	**D$_{art}$-D$_{art}$**
	*are$_{3P\_PL}$ **the** architects that*	***the** engineers*	***have**$_{3P\_PL}$*	*consulted before beginning the works*	
b.	Sono [**gli** architetti]$_i$ che	[**voi** ingegneri]	**avete**	consultato _$_i$ prima di iniziare i lavori.	**D$_{art}$-D$_{pro}$**
	*are$_{3P\_PL}$ **the** architects that*	***you** engineers*	***have**$_{2P\_PL}$*	*consulted before beginning the works*	
c.	Siete [**voi** architetti]$_i$ che	[**gli** ingegneri]	**hanno**	consultato _$_i$ prima di iniziare i lavori.	**D$_{pro}$-D$_{art}$**
	*are$_{2P\_PL}$ **you** architects that*	***the** engineers*	***have**$_{3P\_PL}$*	*consulted before beginning the works*	
d.	Siete [**voi** architetti]$_i$ che	[**voi** ingegneri]	**avete**	consultato _$_i$ prima di iniziare i lavori.	**D$_{pro}$-D$_{pro}$**
	*are$_{2P\_PL}$ **you** architects that*	***you** engineers*	***have**$_{2P\_PL}$*	*consulted before beginning the works*	

Chesi & Canal (2019)

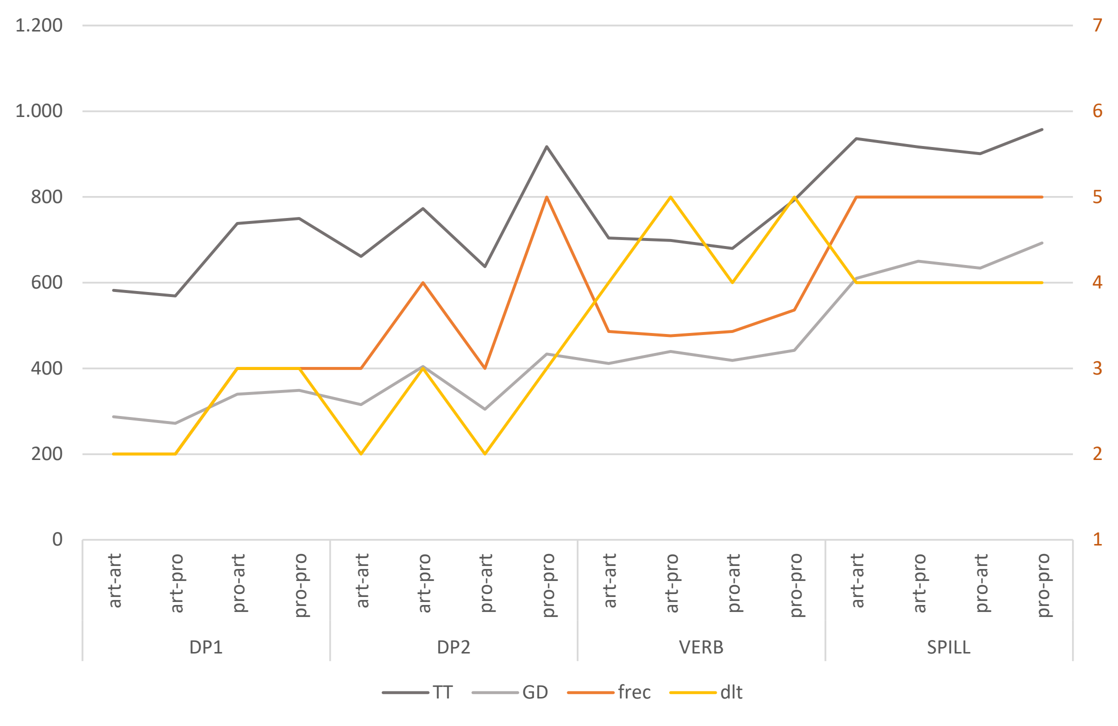
condition	Art$_1$-Art$_2$	Pro$_1$-Pro$_2$	Art$_1$-Pro$_2$	Pro$_1$-Art$_2$
Similarity-based prediction	hard	hard	medium	medium
Intervention-based prediction	hard	hard	medium	medium
Top-down prediction (FRC) – H1	hard	hard	medium	medium
Top-down prediction (FRC) – H2	hard	hardest	medium	hard
Memory-load prediction – A1	hard	hard	hard	hard
Memory-load prediction – A2	harder	hard	hard	harder
Memory-load prediction – A3	hard	harder	harder	hard
ACT-R-based prediction	hard	hard	hard	hard

Chesi & Canal (2019)





Chesi & Canal (2019)



Conclusion

- We rephrased the **intervention-based** idea (Friedmann et al. 2009) in T**op-Down** terms, trying to reconcile the formal account of intervention **(what**) with processing evidence (**when** and **how**)

- What permits to express the exact **complexity cost** is a **Top-down** (that in the end produce a **left-right**) derivation (this way the model fitting can be directly compared with other complexity metrics, e.g. SPLT, Gibson 1998)

- The special role of intervention has been expressed in terms of **interference** at **retrieval** (e.g. Van Dyke & McElree 2006)

Further development

- Feature structures (and actual cues) need to be further refined (other features, e.g. animacy, Kidd et al. 2007, and semantic selection, Gordon et al. 2004, should be considered)
- The counterintuitive idea that Subject “is harder” to retrieve than Object in ORs should receive experimental support
- Is it a purely privative system (+/- F) enough?
- Doing away with LIFO structure which is computationally OK, but psycholinguistically odd (cf. content-adressable memory).

Crucial concepts of this course

- What’s a formal grammar and why do we need to specify it
  - Rewriting rules and recursion
  - Restrictions on rule format and generative power (Chomsky's hierarchy)
  - Equivalence between grammars, finite state automata and push-down automata
  - Where natural languages are located in Chomsky’s Hierarchy
- What’s a computation
  - Problem space and its (algorithmic) exploration
  - Complexity calculus
  - Parsing algorithms (Earley)
- What’s a Top-Down derivation
  - A reconciling view of Competence and Performance
  - Reconstruction and islands
  - Predictions and phases
  - Complexity and intervention (possibly in terms of retrieval)

Possible questions

- What's a phrase structure grammar and how do we read rules such as "**NP → D N**"?
- Why **Regular Grammars** (RG) are less powerful than **Context-Free Grammars** (CFG)? What does it mean $a^n b^n$? It this property generable by a CFG? ***XX***? XX^R?
- What are the **Finite State Automata** (FSA) and **Push-Down Automata** (PDA) and which languages they recognize?
- How do we determine if a string is generable or not with a specific grammar? (pumping lemmas)
- What are the dimensions used to express **computational complexity**?
- How do we use context-free grammar in **parsing**?
- What's the major difference between **RG/CFG** and **minimalism**?
- How do you express complexity in terms of **intervenience**?