Computational Linguistics

8. Features and Unification, Language and Complexity

Xiaojing Bai

Tsinghua University

https://bxjthu.github.io/CompLing

At the end of this session you will

- know about feature structures and their representations;
- know how to unify feature structures and know when unification will fail;
- know how to integrate feature structures into a context free grammar and understand the benefits of doing so;
- know about the overloaded term ‘complexity’;
- understand that algorithms may be classified by their time and space complexity;
- know that formal grammars can be more or less complex as defined by their generative power.

Problems with CFGs

$$
\begin{aligned}
\text{S} &\rightarrow \text{NP VP} \\
\text{NP} &\rightarrow \text{N PP} \\
\text{NP} &\rightarrow \text{N} \\
\text{PP} &\rightarrow \text{P NP} \\
\text{VP} &\rightarrow \text{VP PP} \\
\text{VP} &\rightarrow \text{V VP} \\
\text{VP} &\rightarrow \text{V NP} \\
\text{VP} &\rightarrow \text{V} \\
\text{N} &\rightarrow \{\text{it, fish, rivers, pools,} \\
&\qquad \text{December, Scotland, they}\} \\
\text{P} &\rightarrow \{\text{in}\} \\
\text{V} &\rightarrow \{\text{can, fish}\}
\end{aligned}
$$

Problems with CFGs

Agreement

they fish rivers

? *it fish rivers*

Sub-categorization

they can can it

? *they fish fish it*

Long distance dependency

which dress did you say [ ] looks better with those shoes

what gluten-free products does that new company sell [ ]

? *what gluten free products does that new company sell cake*

S	$\rightarrow$	NP VP
NP	$\rightarrow$	N PP
NP	$\rightarrow$	N
PP	$\rightarrow$	P NP
VP	$\rightarrow$	VP PP
VP	$\rightarrow$	V VP
VP	$\rightarrow$	V NP
VP	$\rightarrow$	V
N	$\rightarrow$	{it, fish, rivers, pools, December, Scotland, they}
P	$\rightarrow$	{in}
V	$\rightarrow$	{can, fish}

How to solve these problems?

$$\begin{aligned}
\text{S} &\rightarrow \text{NP VP} \\
\text{VP} &\rightarrow \text{V NP} \\
\text{NP} &\rightarrow \text{N} \\
\text{V} &\rightarrow \{\text{can, fish}\} \\
\text{N} &\rightarrow \{\text{it, fish, rivers, pools, I, you,} \\
&\quad\ \text{December, Scotland, they}\}
\end{aligned}$$

How to solve these problems?

S $\longrightarrow$ NP VP

VP $\longrightarrow$ V NP

NP $\longrightarrow$ N

V $\longrightarrow$ {can, fish}

N $\longrightarrow$ {it, fish, rivers, pools, I, you, December, Scotland, they}

$$\text{S} \longrightarrow \text{NP}_{1st2ndsing}\ \ \text{VP}_{1st2ndsing}$$

$$\text{S} \longrightarrow \text{NP}_{3rdsing}\ \ \text{VP}_{3rdsing}$$

$$\text{S} \longrightarrow \text{NP}_{plural}\ \ \text{VP}_{plural}$$

How to solve these problems?

$$
\begin{aligned}
\text{S} &\rightarrow \text{NP VP} \\
\text{VP} &\rightarrow \text{V NP} \\
\text{NP} &\rightarrow \text{N} \\
\text{V} &\rightarrow \{\text{can, fish}\} \\
\text{N} &\rightarrow \{\text{it, fish, rivers, pools, I, you,} \\
&\quad\ \ \text{December, Scotland, they}\}
\end{aligned}
$$

$$
\begin{aligned}
\text{VP}_{1st2ndsing} &\rightarrow \text{V}_{1st2ndsing}\ \text{NP}_{1st2ndsing} \\
\text{VP}_{1st2ndsing} &\rightarrow \text{V}_{1st2ndsing}\ \text{NP}_{3rdsing} \\
\text{VP}_{1st2ndsing} &\rightarrow \text{V}_{1st2ndsing}\ \text{NP}_{plural} \\
\\
\text{VP}_{3rdsing} &\rightarrow \text{V}_{3rdsing}\ \text{NP}_{1st2ndsing} \\
\text{VP}_{3rdsing} &\rightarrow \text{V}_{3rdsing}\ \text{NP}_{3rdsing} \\
\text{VP}_{3rdsing} &\rightarrow \text{V}_{3rdsing}\ \text{NP}_{plural} \\
\\
\text{VP}_{plural} &\rightarrow \text{V}_{plural}\ \text{NP}_{1st2ndsing} \\
\text{VP}_{plural} &\rightarrow \text{V}_{plural}\ \text{NP}_{3rdsing} \\
\text{VP}_{plural} &\rightarrow \text{V}_{plural}\ \text{NP}_{plural}
\end{aligned}
$$

How to solve these problems?

$$\begin{aligned}
\text{S} &\longrightarrow \text{NP VP} \\
\text{VP} &\longrightarrow \text{V NP} \\
\text{NP} &\longrightarrow \text{N} \\
\text{V} &\longrightarrow \{\text{can, fish}\} \\
\text{N} &\longrightarrow \{\text{it, fish, rivers, pools, I, you,} \\
&\qquad \text{December, Scotland, they}\}
\end{aligned}$$

$$\begin{aligned}
\text{NP}_{1st2ndsing} &\longrightarrow \text{N}_{1st2ndsing} \\
\text{NP}_{3rdsing} &\longrightarrow \text{N}_{3rdsing} \\
\text{NP}_{plural} &\longrightarrow \text{N}_{plural}
\end{aligned}$$

How to solve these problems?

$$
\begin{aligned}
\text{S} &\longrightarrow \text{NP VP} \\
\text{VP} &\longrightarrow \text{V NP} \\
\text{NP} &\longrightarrow \text{N} \\
\text{V} &\longrightarrow \{\text{can, fish}\} \\
\text{N} &\longrightarrow \{\text{it, fish, rivers, pools, I, you,} \\
&\qquad \text{December, Scotland, they}\}
\end{aligned}
$$

$$
\begin{aligned}
\text{N}_{1st2ndsing} &\longrightarrow \{\text{I, you}\} \\
\text{N}_{3rdsing} &\longrightarrow \{\text{it, December, Scotland}\} \\
\text{N}_{plural} &\longrightarrow \{\text{fish, rivers, pools, they}\}
\end{aligned}
$$

$$
\begin{aligned}
\text{V}_{1st2ndsing} &\longrightarrow \{\text{fish, can}\} \\
\text{V}_{3rdsing} &\longrightarrow \{\text{fishes, cans, can}\} \\
\text{V}_{plural} &\longrightarrow \{\text{fish, can}\}
\end{aligned}
$$

Problems with CFGs

Agreement

they fish rivers

? *it fish rivers*

Sub-categorization

they can can it

? *they fish fish it*

Long distance dependency

which dress did you say [ ] looks better with those shoes

what gluten-free products does that new company sell [ ]

? *what gluten free products does that new company sell cake*

S	$\rightarrow$	NP VP
NP	$\rightarrow$	N PP
NP	$\rightarrow$	N
PP	$\rightarrow$	P NP
VP	$\rightarrow$	VP PP
VP	$\rightarrow$	V VP
VP	$\rightarrow$	V NP
VP	$\rightarrow$	V
N	$\rightarrow$	{it, fish, rivers, pools, December, Scotland, they}
P	$\rightarrow$	{in}
V	$\rightarrow$	{can, fish}

Grammar formalisms

- Constituent-based language models
- Dependency-based language models

- Constraint-based language models

  A more fine-grained way of representing and placing constraints on grammatical categories

A feature structure represented by an attribute-value matrix

$$\begin{array}{ll} \text{FEATURE}_1 & value_1 \\ \text{FEATURE}_2 & value_2 \\ \ldots & \ldots \\ \text{FEATURE}_n & value_n \end{array}$$

$$\begin{array}{ll} \text{CAT} & NP \\ \text{NUMBER} & sing \\ \text{PERSON} & 3rd \end{array}$$

$$\begin{array}{ll} \text{CAT} & NP \\ \text{AGREEMENT} & \left[\begin{array}{ll} \text{NUMBER} & sing \\ \text{PERSON} & 3rd \end{array} \right] \end{array}$$

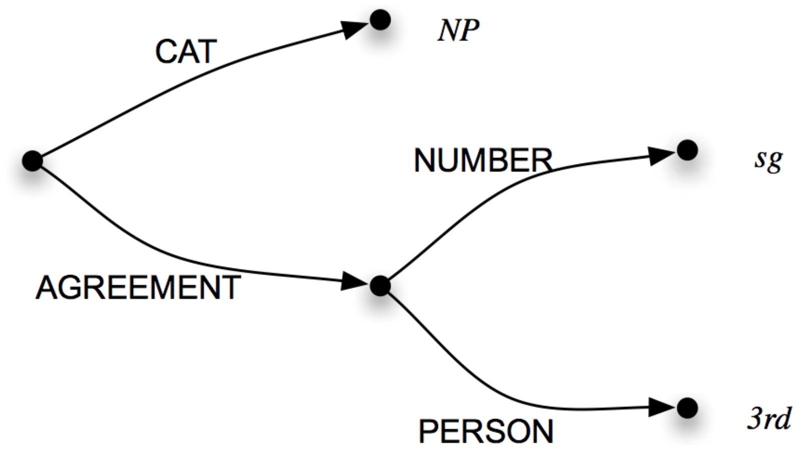
A feature structure represented by a directed acyclic graph

- Features as labeled edges
- Values as nodes
- Feature path

  A sequence of features through a feature structure leading to a particular value

  The < AGREEMENT NUMBER > path

  The < AGREEMENT PERSON > path



$$\begin{bmatrix} \text{CAT} & NP \\ \text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & sing \\ \text{PERSON} & 3rd \end{bmatrix} \end{bmatrix}$$

Reentrancy and reentrant structures

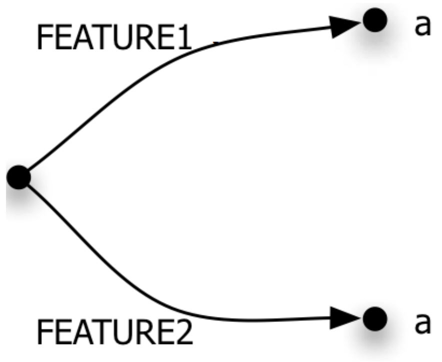
Reentrancy: A feature structure occurs more than once in an enclosing feature structure, i.e. there are two or more feature paths of reaching the same node in the directed acyclic graph.

- Non-reentrant

$$\begin{bmatrix} \text{FEATURE}_1 & a \\ \text{FEATURE}_2 & a \end{bmatrix}$$

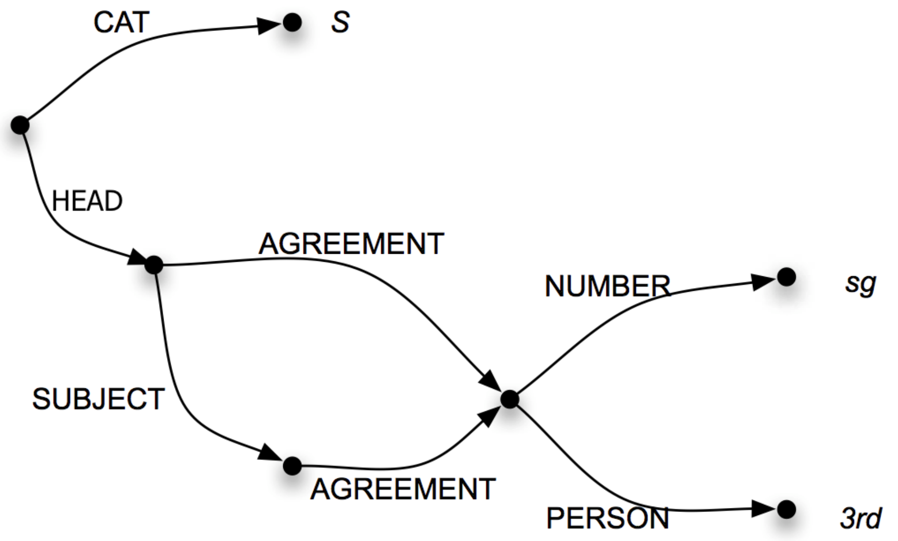
- Reentrant

$$\begin{bmatrix} \text{FEATURE}_1 & [1]a \\ \text{FEATURE}_2 & [1] \end{bmatrix}$$



Reentrancy and reentrant structures

$$\begin{bmatrix} \text{CAT} & S \\ \text{HEAD} & \begin{bmatrix} \text{AGREEMENT} & [1]\begin{bmatrix}\text{NUMBER} & sing \\ \text{PERSON} & 3rd\end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix}\text{AGREEMENT} & [1]\end{bmatrix}\end{bmatrix}\end{bmatrix}$$



Unification of feature structures

- A merger of the original two structures into one larger structure
- A union of all the information stored in each of the original structures
- Compatibility

$$\begin{bmatrix} \text{NUMBER} & sing \end{bmatrix} \sqcup \begin{bmatrix} \text{NUMBER} & sing \end{bmatrix} = \begin{bmatrix} \text{NUMBER} & sing \end{bmatrix}$$

$$\begin{bmatrix} \text{NUMBER} & sing \end{bmatrix} \sqcup \begin{bmatrix} \text{NUMBER} & plural \end{bmatrix} Fails!$$

$$\begin{bmatrix} \text{NUMBER} & sing \end{bmatrix} \sqcup \begin{bmatrix} \text{NUMBER} & [] \end{bmatrix} = \begin{bmatrix} \text{NUMBER} & sing \end{bmatrix}$$

$$\begin{bmatrix} \text{NUMBER} & sing \end{bmatrix} \sqcup \begin{bmatrix} \text{PERSON} & 3rd \end{bmatrix} = \begin{bmatrix} \text{NUMBER} & sing \\ \text{PERSON} & 3rd \end{bmatrix}$$

Unification of feature structures

$$
\begin{bmatrix}
\text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & sing \end{bmatrix} \\[2em]
\text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & sing \end{bmatrix} \end{bmatrix}
\end{bmatrix}
$$

$$
\sqcup \begin{bmatrix} \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{PERSON} & 3rd \\ \text{NUMBER} & sing \end{bmatrix} \end{bmatrix} \end{bmatrix}
$$

$$
= \begin{bmatrix}
\text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & sing \end{bmatrix} \\[2em]
\text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{PERSON} & 3rd \\ \text{NUMBER} & sing \end{bmatrix} \end{bmatrix}
\end{bmatrix}
$$

Unification of reentrant feature structures

$$\begin{bmatrix} \text{AGREEMENT} & [1]\begin{bmatrix} \text{NUMBER} & sing \\ \text{PERSON} & 3rd \end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & [1] \end{bmatrix} \end{bmatrix}$$

$$\sqcup \begin{bmatrix} \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{PERSON} & 3rd \\ \text{NUMBER} & sing \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \text{AGREEMENT} & [1]\begin{bmatrix} \text{NUMBER} & sing \\ \text{PERSON} & 3rd \end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & [1] \end{bmatrix} \end{bmatrix}$$

Unification of reentrant feature structures

$$\begin{bmatrix} \text{AGREEMENT} & [1] \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & [1] \end{bmatrix} \end{bmatrix}$$

$$\sqcup \begin{bmatrix} \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{PERSON} & 3rd \\ \text{NUMBER} & sing \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \text{AGREEMENT} & [1] \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & [1] \begin{bmatrix} \text{PERSON} & 3rd \\ \text{NUMBER} & sing \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Failed unification of feature structures

$$\begin{bmatrix} \text{AGREEMENT} & [1]\begin{bmatrix} \text{NUMBER} & sing \\ \text{PERSON} & 3rd \end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & [1] \end{bmatrix} \end{bmatrix}$$

$$\sqcup \begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & sing \\ \text{PERSON} & 3rd \end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & plural \\ \text{PERSON} & 3rd \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Fails!

Feature structures in the grammar

Augmenting the ordinary CFGs rules with attachments that specify feature structures for the constituents of the rules, along with appropriate unification operations that express constraints on those constituents.

$\beta_0 \rightarrow \beta_1 \ldots \beta_n$

{*set of constraints*}

< β_i feature path > = atomic value

< β_i feature path > = < β_j feature path >

Application: agreement

[1] *This flight serves breakfast.*
[2] *Does this flight serve breakfast?*
[3] *Do these flights serve breakfast?*

S $\rightarrow$ NP VP		S $\rightarrow$ Aux NP VP	
	< NP AGREEMENT > = < VP AGREEMENT >		< Aux AGREEMENT > = < NP AGREEMENT >
NP $\rightarrow$ Det Nominal			
	< Det AGREEMENT > = < Nominal AGREEMENT >		
	< NP AGREEMENT > = < Nominal AGREEMENT >		
Aux $\rightarrow$ do		Aux $\rightarrow$ does	
	< Aux AGREEMENT NUMBER > = *plural*		< Aux AGREEMENT NUMBER > = *sing*
	< Aux AGREEMENT PERSON > = *3rd*		< Aux AGREEMENT PERSON > = *3rd*

Application: head features

VP	$\rightarrow$	Verb NP
		< VP AGREEMENT > = < Verb AGREEMENT >
VP	$\rightarrow$	Verb NP
		< VP HEAD > = < Verb HEAD >
NP	$\rightarrow$	Det Nominal
		< Det AGREEMENT > = < Nominal AGREEMENT >
		< NP AGREEMENT > = < Nominal AGREEMENT >
NP	$\rightarrow$	Det Nominal
		< Det HEAD AGREEMENT > = < Nominal HEAD AGREEMENT >
		< NP HEAD > = < Nominal HEAD >
Nominal	$\rightarrow$	Noun
		< Nominal AGREEMENT > = < Noun AGREEMENT >
Nominal	$\rightarrow$	Noun
		< Nominal HEAD > = < Noun HEAD >
Noun	$\rightarrow$	*flights*
		< Noun HEAD AGREEMENT NUMBER > = *plural*

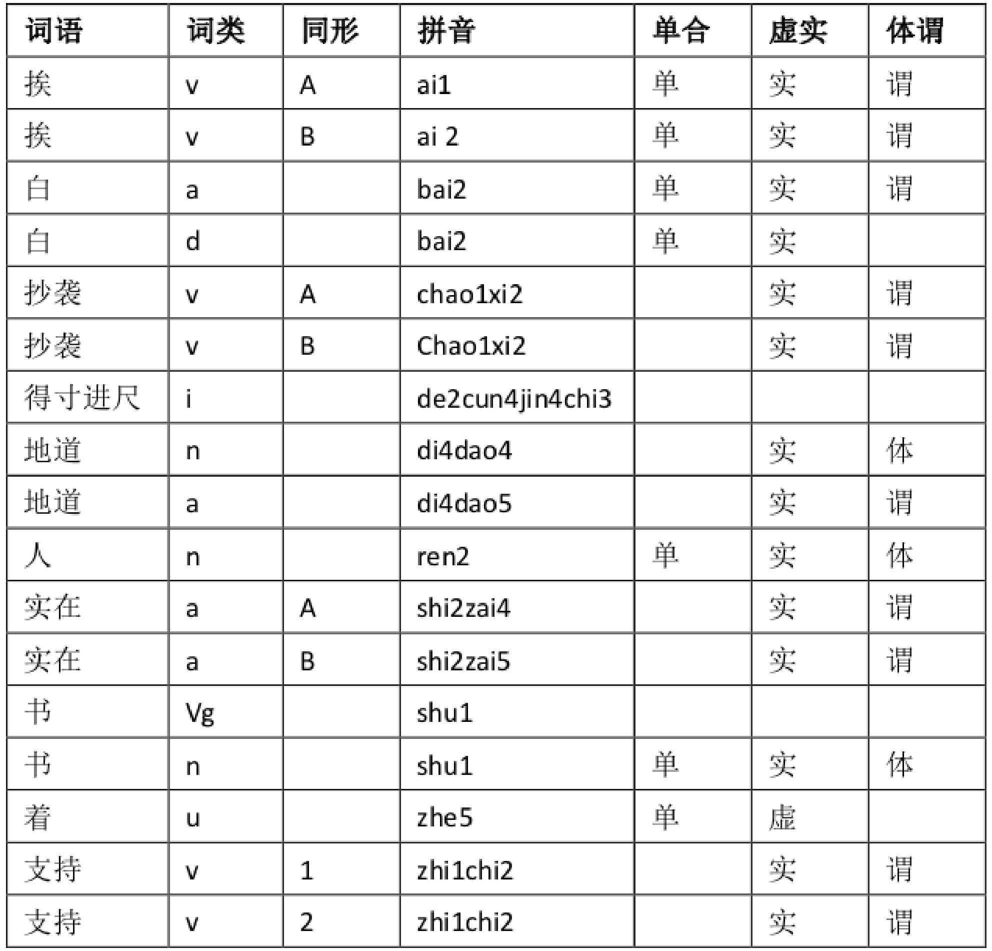
The features for most grammatical categories are copied from one of the children to the parent.

The child that provides the features is called the head of the phrase, and the features copied are referred to as head features.

Applications

- Agreement
- Head features
- Subcategorization
- Long-distance dependencies

Feature structures in language knowledge bases



词语	词类	同形	拼音	单合	虚实	体谓
挨	v	A	ai1	单	实	谓
挨	v	B	ai 2	单	实	谓
白	a		bai2	单	实	谓
白	d		bai2	单	实	
抄袭	v	A	chao1xi2		实	谓
抄袭	v	B	Chao1xi2		实	谓
得寸进尺	i		de2cun4jin4chi3			
地道	n		di4dao4		实	体
地道	a		di4dao5		实	谓
人	n		ren2	单	实	体
实在	a	A	shi2zai4		实	谓
实在	a	B	shi2zai5		实	谓
书	Vg		shu1			
书	n		shu1	单	实	体
着	u		zhe5	单	虚	
支持	v	1	zhi1chi2		实	谓
支持	v	2	zhi1chi2		实	谓

副词用法词典

Complexity: an overloaded term

It may refer to:

- the computational ‘expense’ of an algorithm;
- the generative power of a grammar; or
- the human processing difficulty of a sentence.

The complexity of algorithms

Comparing the efficiency of algorithms: how will they behave in the worst case?

Two particular measures:

1. the amount of time it takes for the algorithm to run to completion;
2. the amount of space (computer memory) it will take to hold the instructions that are waiting to be executed while the algorithm is running.

The time complexity of algorithms

Scenario 1: How many items are on my 11.11 shopping list?

```
+ 旺仔大礼包
+ Thermos保温杯
+ 森田面膜
+ 欧莱雅卸妆水
+ 沃隆每日坚果
+ 良品铺子猪肉脯
```

The counting algorithm

- Start with a count of zero item;
- While there are items left to count:
  - Point to the next item on the list; [0.25s for each]
  - Increase the count by 1. [0.25s for each]

The time cost of the algorithm: $0.5n$ seconds

The time complexity of algorithms

Scenario 2: How many items are on the following two shopping lists?

```
+ 旺仔大礼包
+ Thermos保温杯
+ 森田面膜
+ 欧莱雅卸妆水
+ 沃隆每日坚果
+ 良品铺子猪肉脯
```

```
+ MacBook VGA转接头
+ 中国哲学简史
+ 旺仔大礼包
+ 沙宣洗护套装
+ 云南白药牙膏
+ 良品铺子猪肉脯
```

The condensing algorithm

- Start with a count of 6 items;
- While there are items left my friend's list:
    - Point to the next item on my friend's list; [0.25s for each]
    - Compare the current item on my friend's list to all the items on my original list; if no match is found, then add the current item to my list and increase the count by 1. [0.5s for each]

The time cost of the algorithm: $(0.25 + 0.5n) * n$
or $0.25n + 0.5n^2$ seconds

The time complexity of algorithms

Big O notation of time complexity (or asymptotic complexity)

In computer science, big O notation is used to classify algorithms according to how their running time or space requirements grow as the input size grows.

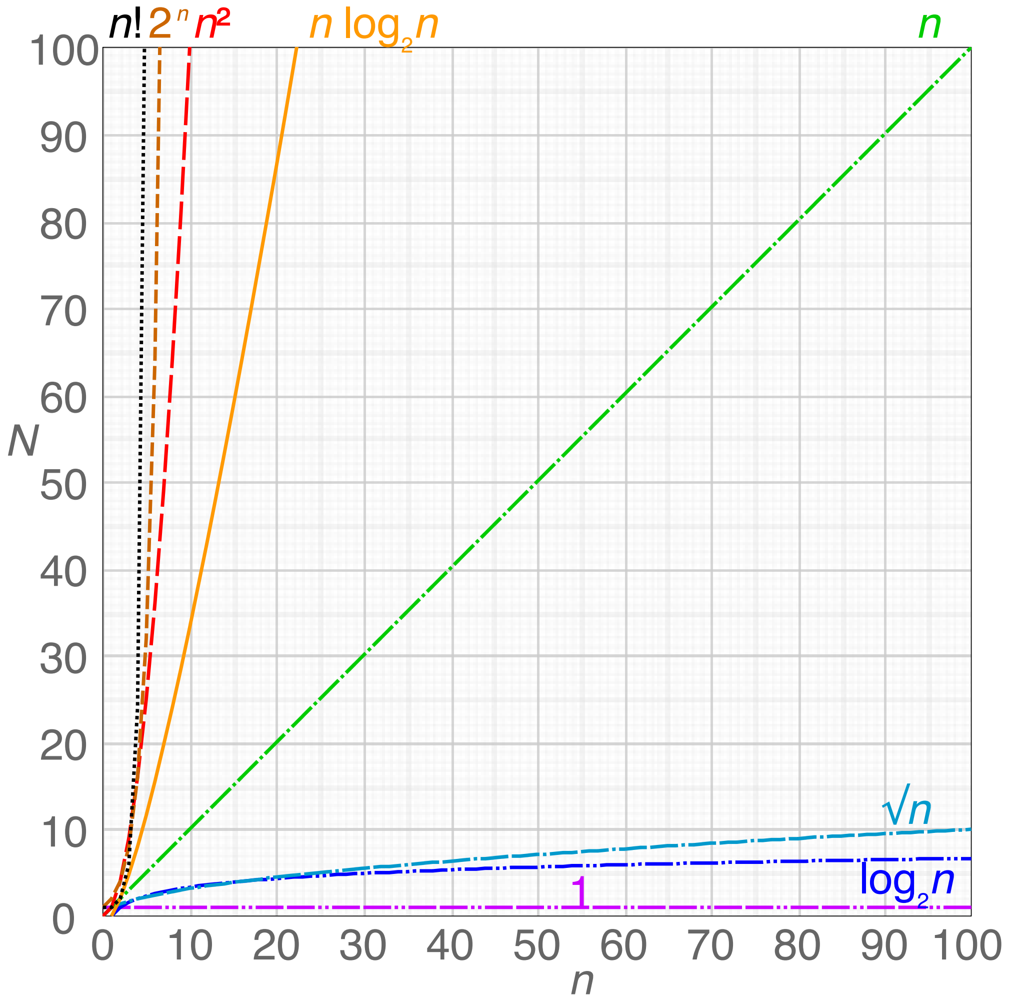
- The counting algorithm: a time complexity of $O(n)$
- The condensing algorithm: a time complexity of $O(n^2)$

Notes: The time cost of an algorithm is usually a complicated formula. We are only interested in the most significant term when stating its order, which means the term involving the highest exponent of n. For example, the time (or the number of steps) it takes to complete a problem of size n might be found to be $T(n) = 4n^2 - 2n + 2$. As n grows large, the n^2 term will come to dominate, so that all other terms can be neglected. An algorithm with a time complexity of $O(c^n)$, where c is a constant, is said to be intractable.

Big O notation

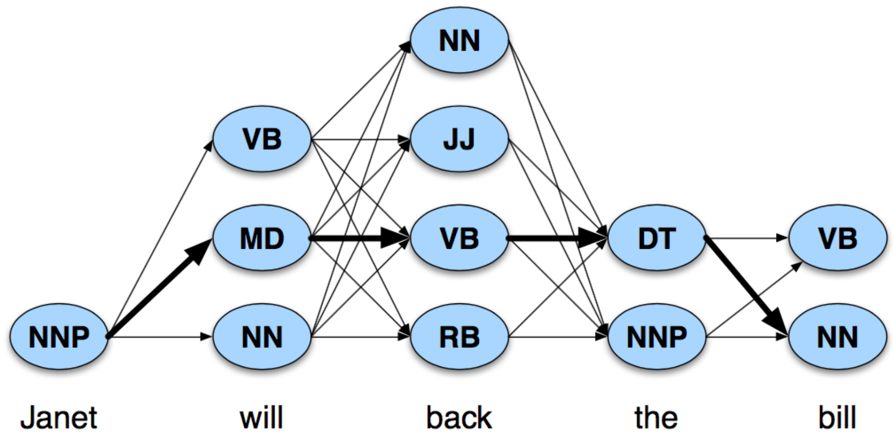
Graphs of functions commonly used in the analysis of algorithms, showing the number of operations N versus input size n for each function

More on Big O notation at wiki



Recap: An example

E.g. Janet will back the bill



Janet/NNP will/MD back/VB the/DT bill/NN

Viterbi algorithm

- Andrew Viterbi, 1967
- A dynamic programming algorithm
- Maximize the probabilities
- Find the most likely sequence of tags
- Algorithm: exponential complexity

  With a tagset of N tags, for a sequence of M words, there are, in the worse case, N^M possible paths.
- Efficiency: N^M vs. $N^2 \times M$

The space complexity of algorithms

The stack memory: Stacks in computing architectures are regions of memory where data is added or removed in a last-in-first-out (LIFO) manner.

Named after the plate-stacking devices

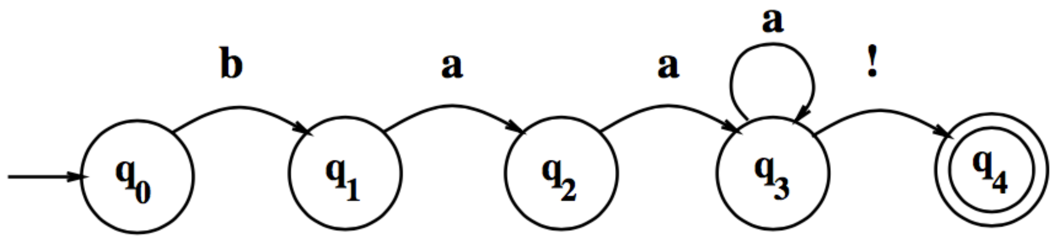
- Pushing: placing an item on the stack
- Popping: removing an item off the stack

The counting/condensing algorithm:

- a space complexity of $O(1)$ stack depth = 1
- a space complexity of $O(n)$ stack depth = n

The complexity of a grammar

- The complexity of the most efficient algorithm to decide whether a string belongs to a regular language is $O(n)$, where n is the length of the string.



- The complexity of many algorithms to decide whether a string belongs to a context-free language is $O(n^a)$, where a is a whole number. The algorithms are therefore tractable.

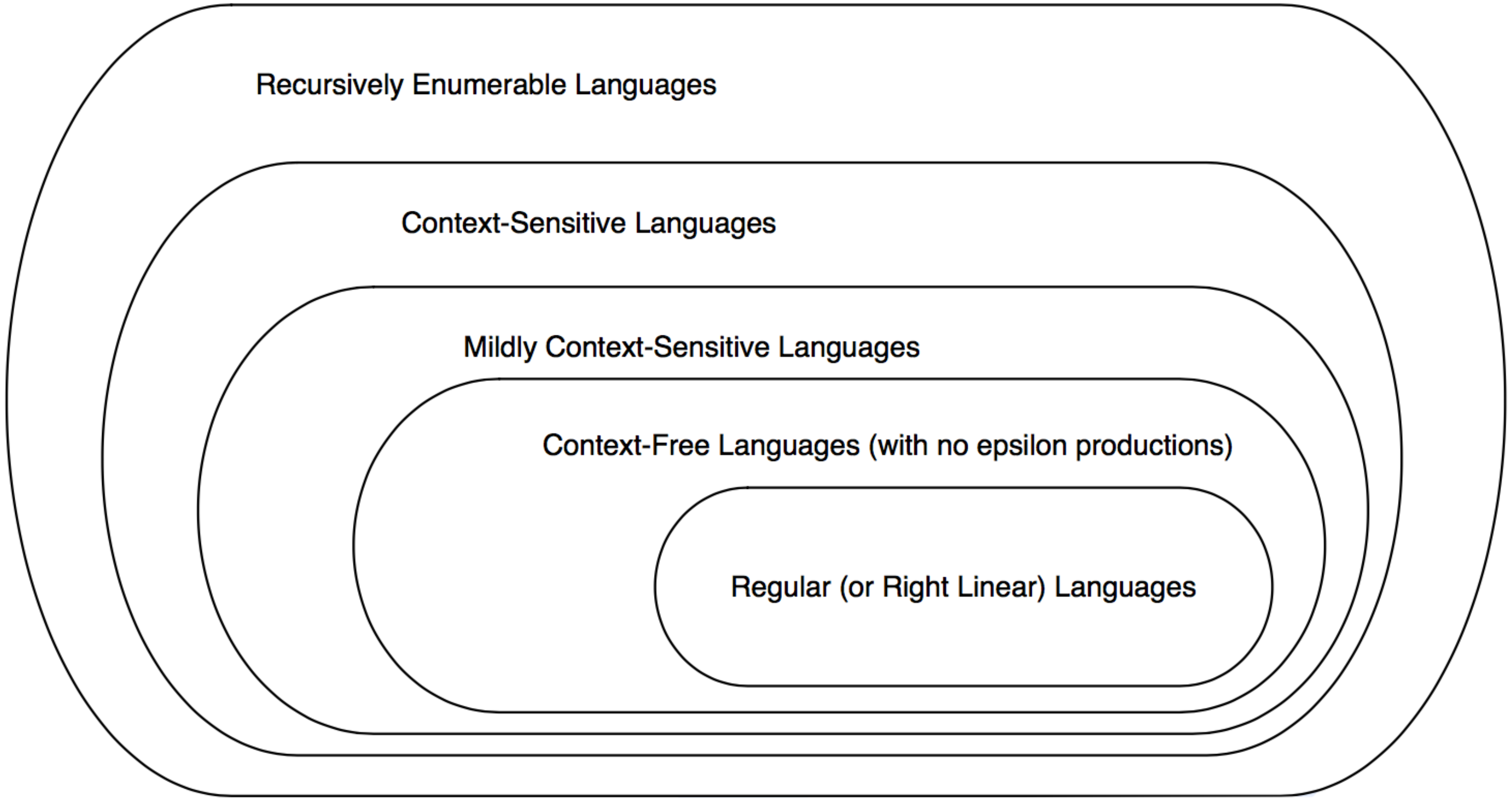
Chomsky hierarchy

- A theoretical tool to compare the generative power or complexity of formal mechanisms
- A theoretical tool to better understand human parsing: What makes individual constructions or sentences hard to understand?

Generative power

- One grammar is of greater generative power than another if it can define a language that the other cannot define.
- The set of languages describable by grammars of greater power subsumes the set of languages describable by grammars of lesser power.

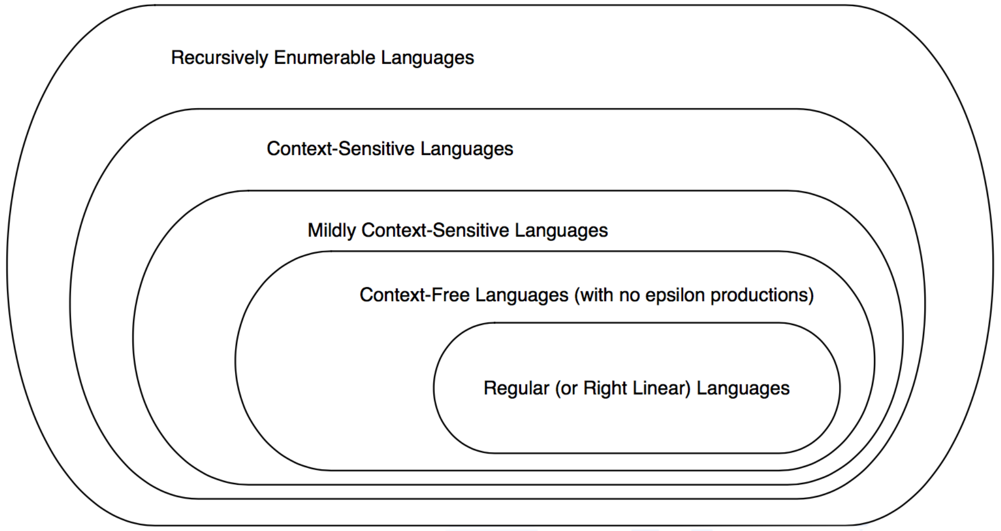
Chomsky hierarchy


Recursively Enumerable Languages
Context-Sensitive Languages
Mildly Context-Sensitive Languages
Context-Free Languages (with no epsilon productions)
Regular (or Right Linear) Languages

Chomsky hierarchy

Type	**Language**	**Rule**	**Complexity**	**Automaton**
0	Recursively Enumerable	$\alpha \to \beta$ (where $\alpha \neq \epsilon$)	Intractable	Turing machine
1	Context-Sensitive	$\alpha A \beta \to \alpha\gamma\beta$ (where $\gamma \neq \epsilon$)	$O(c^n)$, intractable	Linear-bounded
-	Mildly Context-Sensitive			
2	Context-Free	$A \to \gamma$	$O(n^a)$, polynomial	Push down
3	Regular	$A \to xB$ and $A \to x$	$O(n)$, linear	Finite state

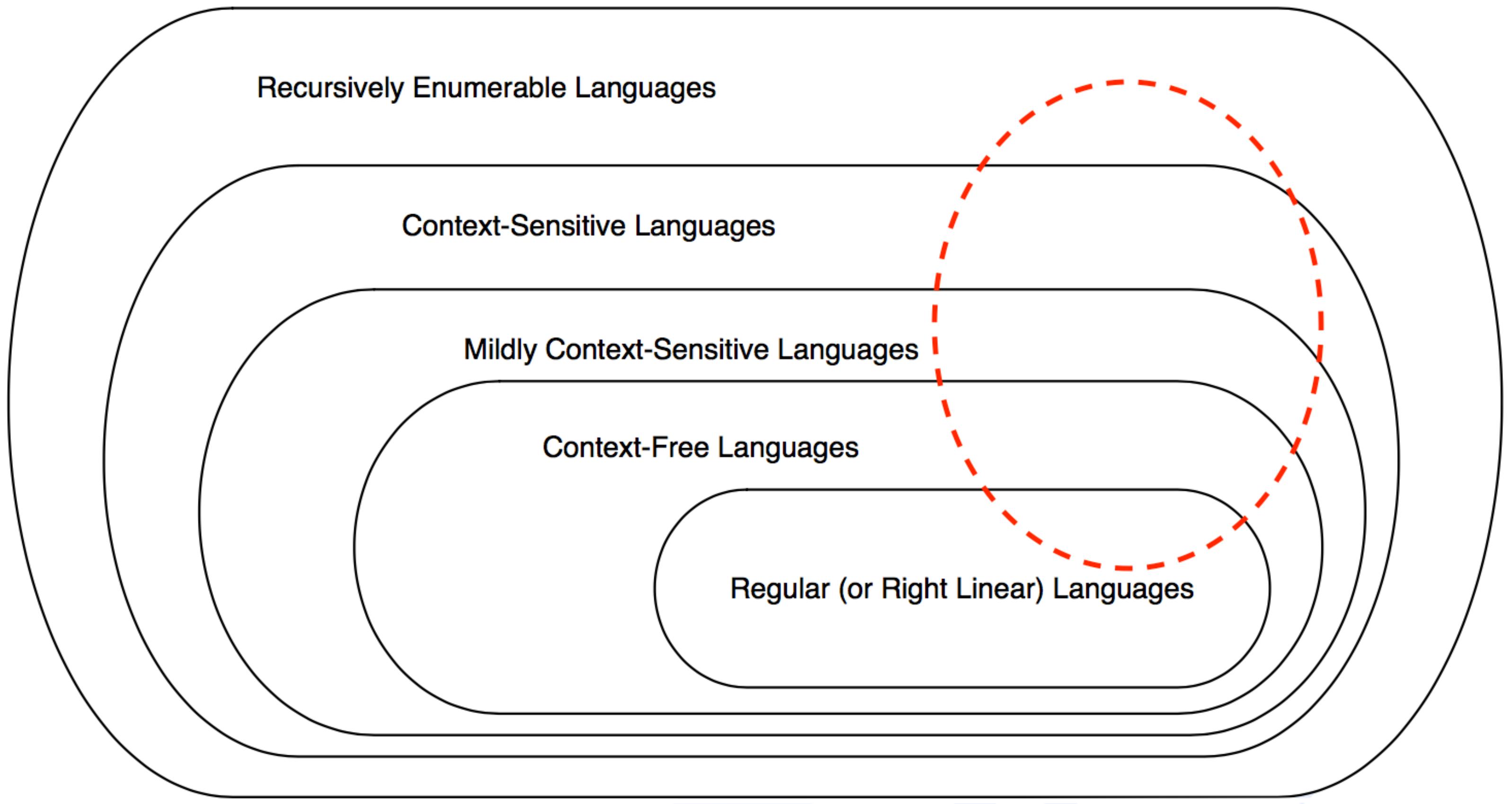
Meaning of symbols:
x: terminal
A and B: non-terminal
α and β: terminal, non-terminal, or empty
γ: terminal or non-terminal


Recursively Enumerable Languages
Context-Sensitive Languages
Mildly Context-Sensitive Languages
Context-Free Languages (with no epsilon productions)
Regular (or Right Linear) Languages

Chomsky hierarchy

- Is some part of natural language representable by a certain class of grammars?
- Which type of rules can be used to write computational grammars for this part of natural language?
- Which type of automata can be used to process the rules?
- Where does a language keeps its complexity?

Chomsky hierarchy


Recursively Enumerable Languages
Context-Sensitive Languages
Mildly Context-Sensitive Languages
Context-Free Languages
Regular (or Right Linear) Languages

At the end of this session you will

- know about feature structures and their representations;
- know how to unify feature structures and know when unification will fail;
- know how to integrate feature structures into a context free grammar and understand the benefits of doing so;
- know about the overloaded term ‘complexity’;
- understand that algorithms may be classified by their time and space complexity;
- know that formal grammars can be more or less complex as defined by their generative power.

Practical 8

- Use the Stanford Parser to parse sentences and draw trees [sample code]
- Building Feature Based Grammars
    - Subcategorization [sample code]
    - Long-distance dependencies [sample code]

Homework

- Read/Review (Quiz 7 on Nov. 21, 2018)
  - J+M_13 (13.1)
  - J+M_second_edition_16
- Practice
  - Practical 8
  - http://www.nltk.org/book/ch09.html

Next session

Meaning Representation and Vector Semantics