Learning more CFGs

Computational Learning of Syntax

Alexander Clark

Department of Philosophy
King's College, London

LSA Summer Institute 2015, Chicago

Topics for today

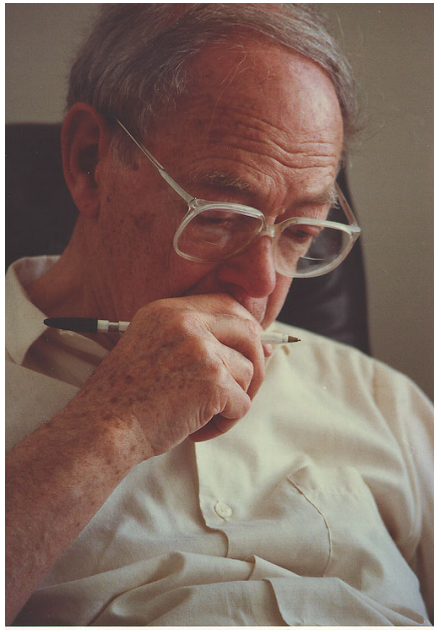
Staying with CFGs

- Beyond congruence classes
- The syntactic concept lattice
- Queries and probabilities
- Primal and Dual algorithms
- Some unlearnable languages

Distributional Learning

Zellig Harris (1949, 1951)

Here as throughout these procedures X and Y are substitutable if for every utterance which includes X we can find (or gain native acceptance for) an utterance which is identical except for having Y in the place of X



Analysis

- Computationally efficient.
- Infinite number of languages infinitely many of which are not regular (infinite and “hierarchically structured”)
- Uses only positive data
- Learns classes of words as well
- Simple proofs of correctness

Why does this work? (I)

Safe generalization

The substitutability property guarantees that we won't overgeneralise.

Why does this work? (I)

Safe generalization

The substitutability property guarantees that we won't overgeneralise.

- So if the language isn't substitutable, we need a better way of testing whether strings are congruent (or whatever).

Why does this work? (II)

Representational adequacy

The nonterminals can (must!) be congruence classes.

Why does this work? (II)

Representational adequacy

The nonterminals can (must!) be congruence classes.

- Suppose $N \stackrel{*}{\Rightarrow} u$ and $N \stackrel{*}{\Rightarrow} v$
- Assume that $S \stackrel{*}{\Rightarrow} lNr$
- Then $lur \in L$ and $lvr \in L$
- So $u \equiv v$
- Moreover, we only need one nonterminal for each congruence class.
- But if the language is not substitutable this no longer holds.

Why does this work? (III)

Efficient enumeration of substructures.

We can decompose the strings efficiently.

Why does this work? (III)

Efficient enumeration of substructures.

We can decompose the strings efficiently.

- If w is of length n
- then there are only $n(n-1)/2$ ways to decompose it into $l \cdot u \cdot r$

Why does this work? (IV)

Efficient parsing

We can test whether $w \in L(G)$ efficiently.

Why does this work? (IV)

Efficient parsing

We can test whether $w \in L(G)$ efficiently.

- If we can't parse efficiently we (probably/almost certainly) can't learn efficiently.

Classes of grammars

- The representation class is the class of all context free grammars.
- The learnable class is the class of substitutable CF languages.

Classes of grammars

- The representation class is the class of all context free grammars.
- The learnable class is the class of substitutable CF languages.
- The hypothesis class is rather hard to define: the algorithm sometimes outputs grammars that define languages that are not substitutable.
- There are languages which the learner will correctly learn some of the time, but not all of the time.
- Moreover whether a grammar defines a substitutable language is not a decidable property, so it may be *impossible* to restrict the learner.

Two “universals”?

(Both of these seem to be false.)

- Natural languages are representable by CFGs.
- Natural languages are substitutable.

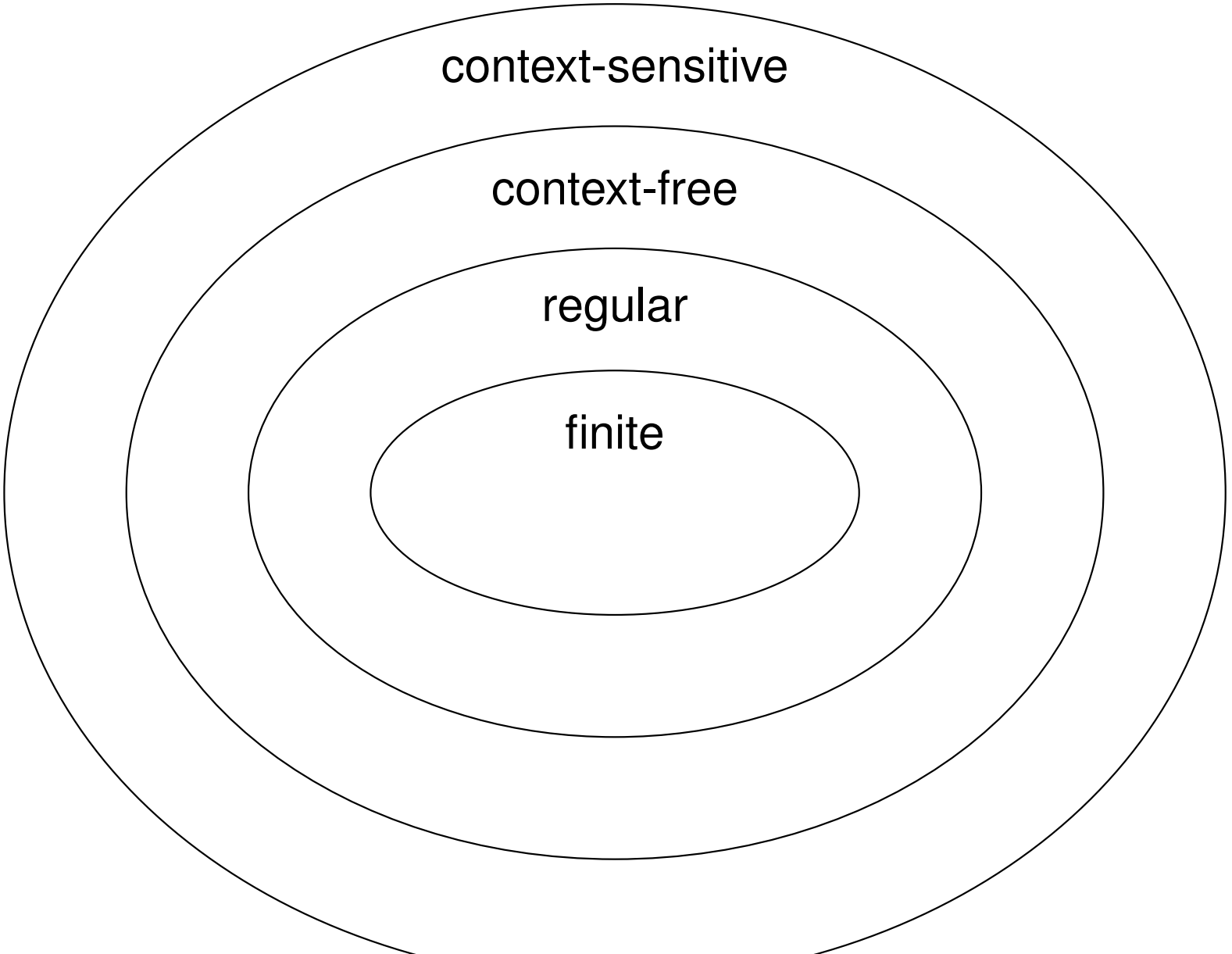
Tension

Chomsky, 1986

To achieve descriptive adequacy it often seems necessary to enrich the system of available devices, whereas to solve our case of Plato's problem we must restrict the system of available devices so that only a few languages or just one are determined by the given data. It is the tension between these two tasks that makes the field an interesting one, in my view.

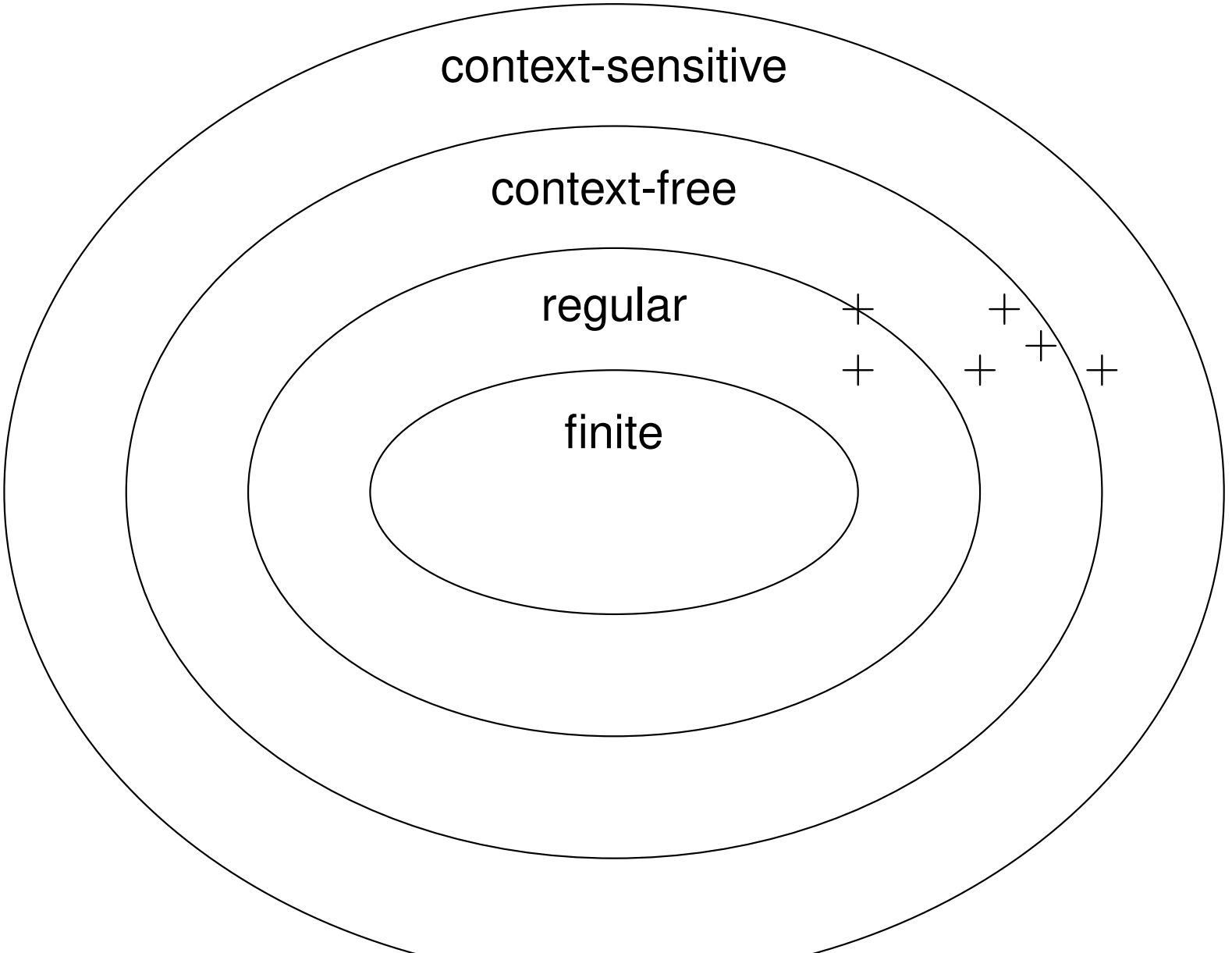
Standard strategy

Restrict hypothesis class



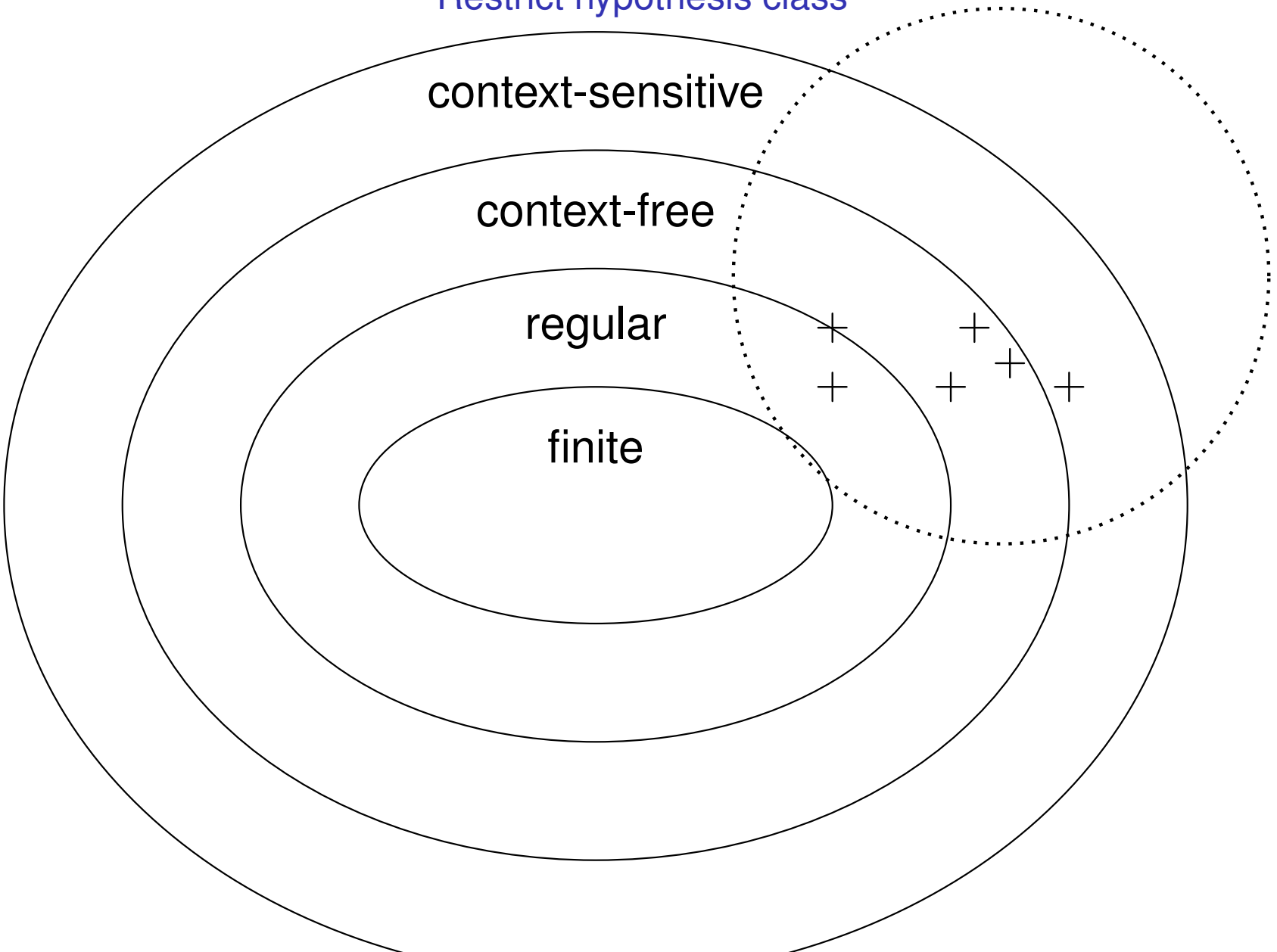
Standard strategy

Restrict hypothesis class



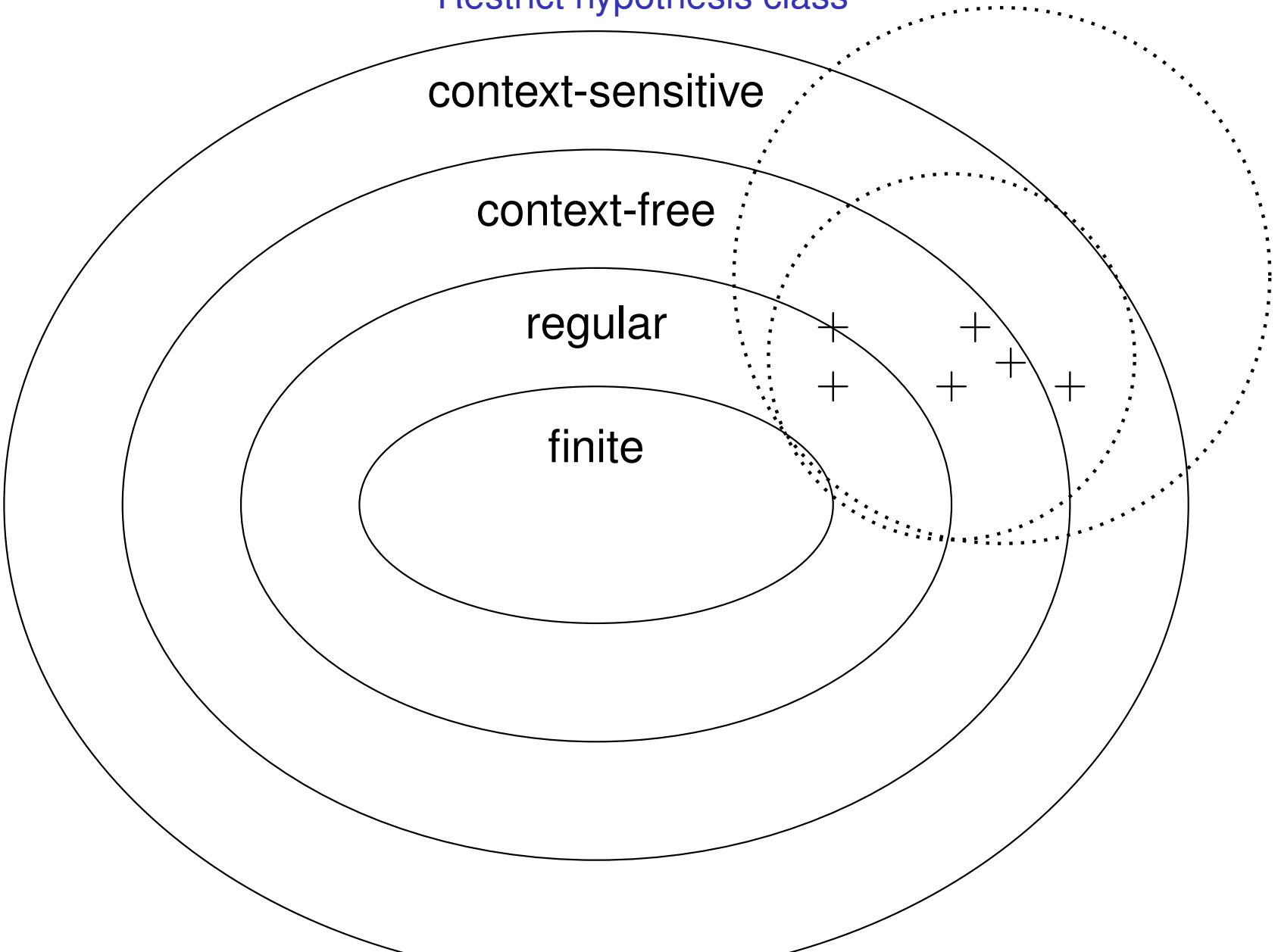
Standard strategy

Restrict hypothesis class



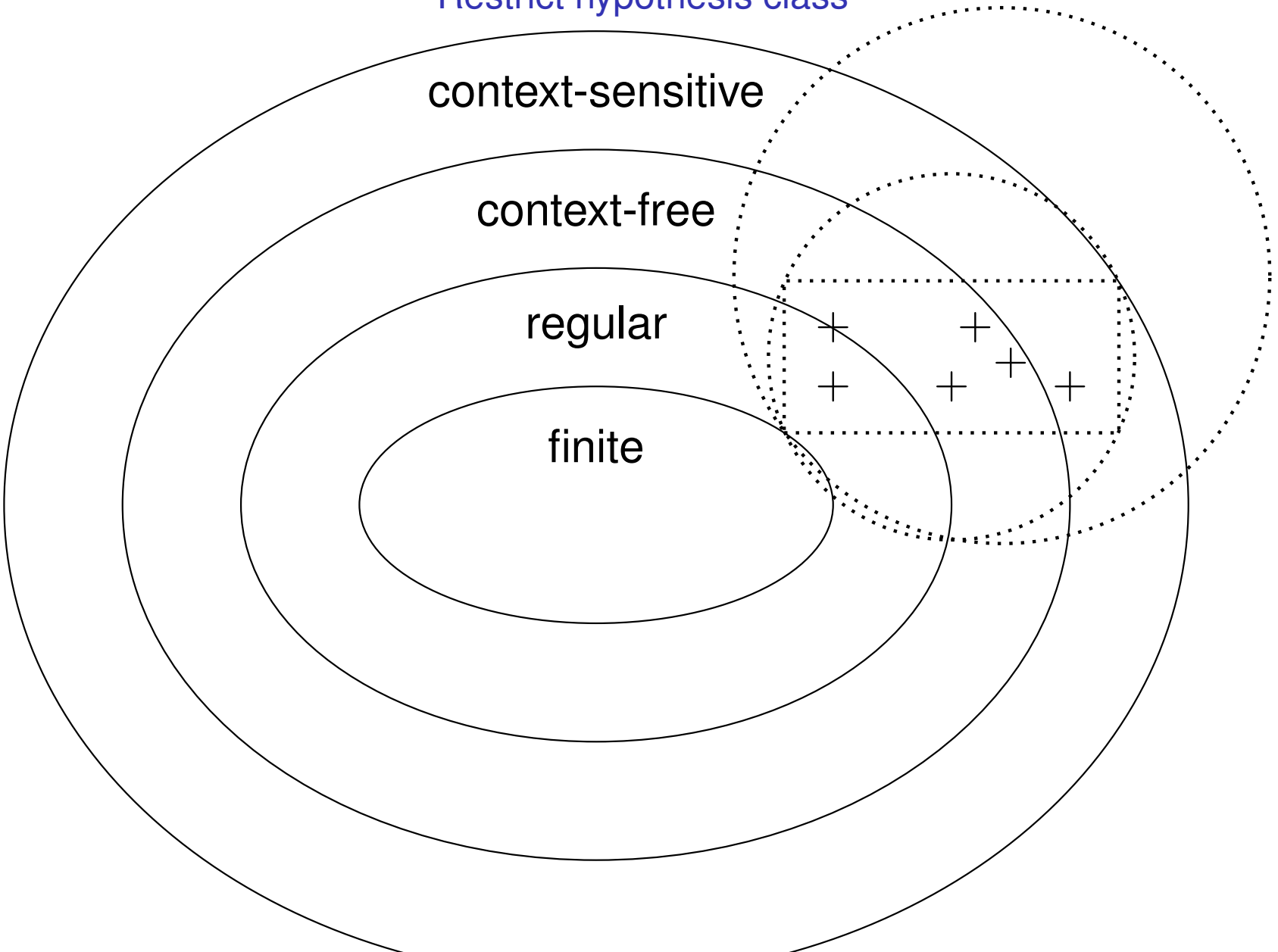
Standard strategy

Restrict hypothesis class



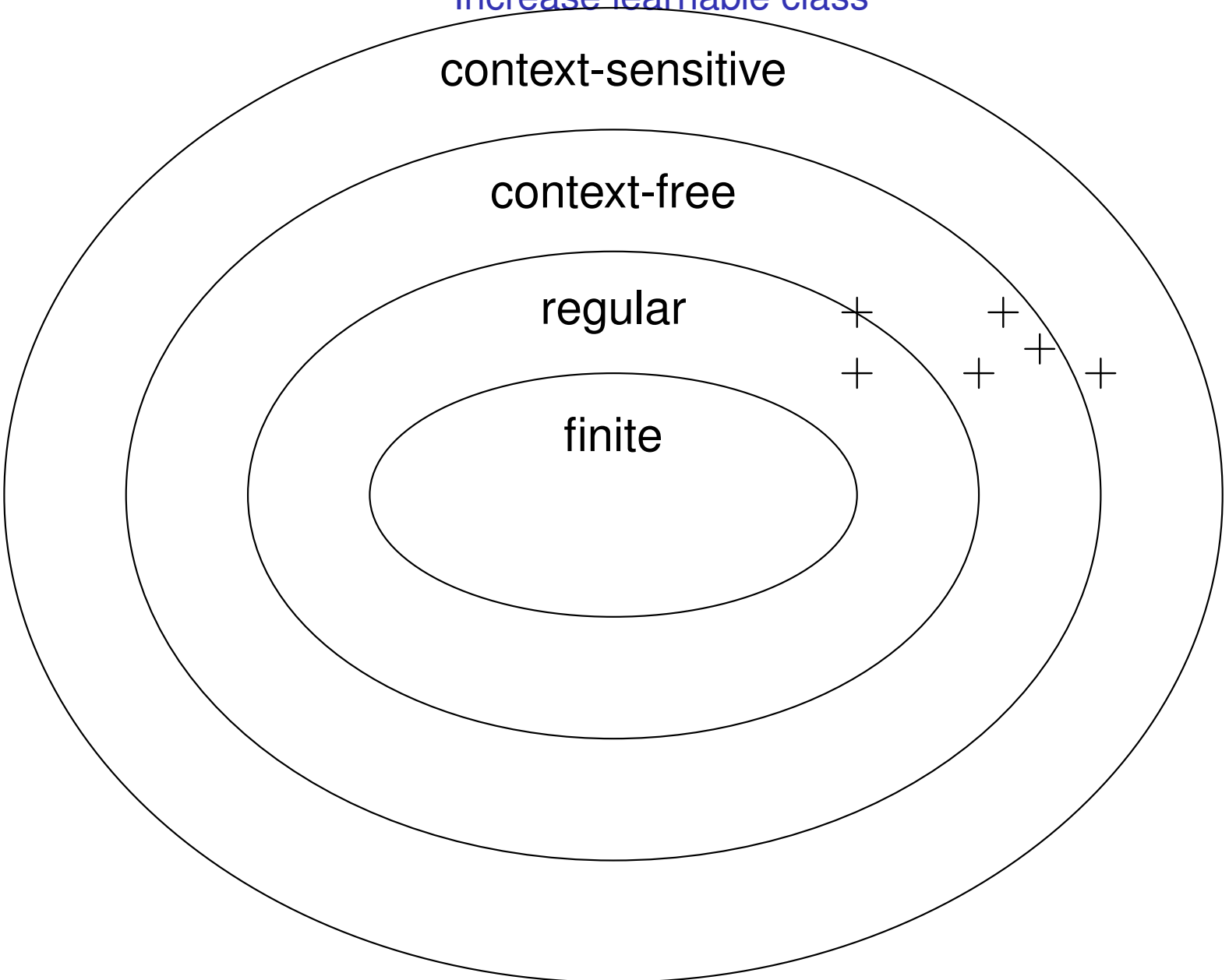
Standard strategy

Restrict hypothesis class



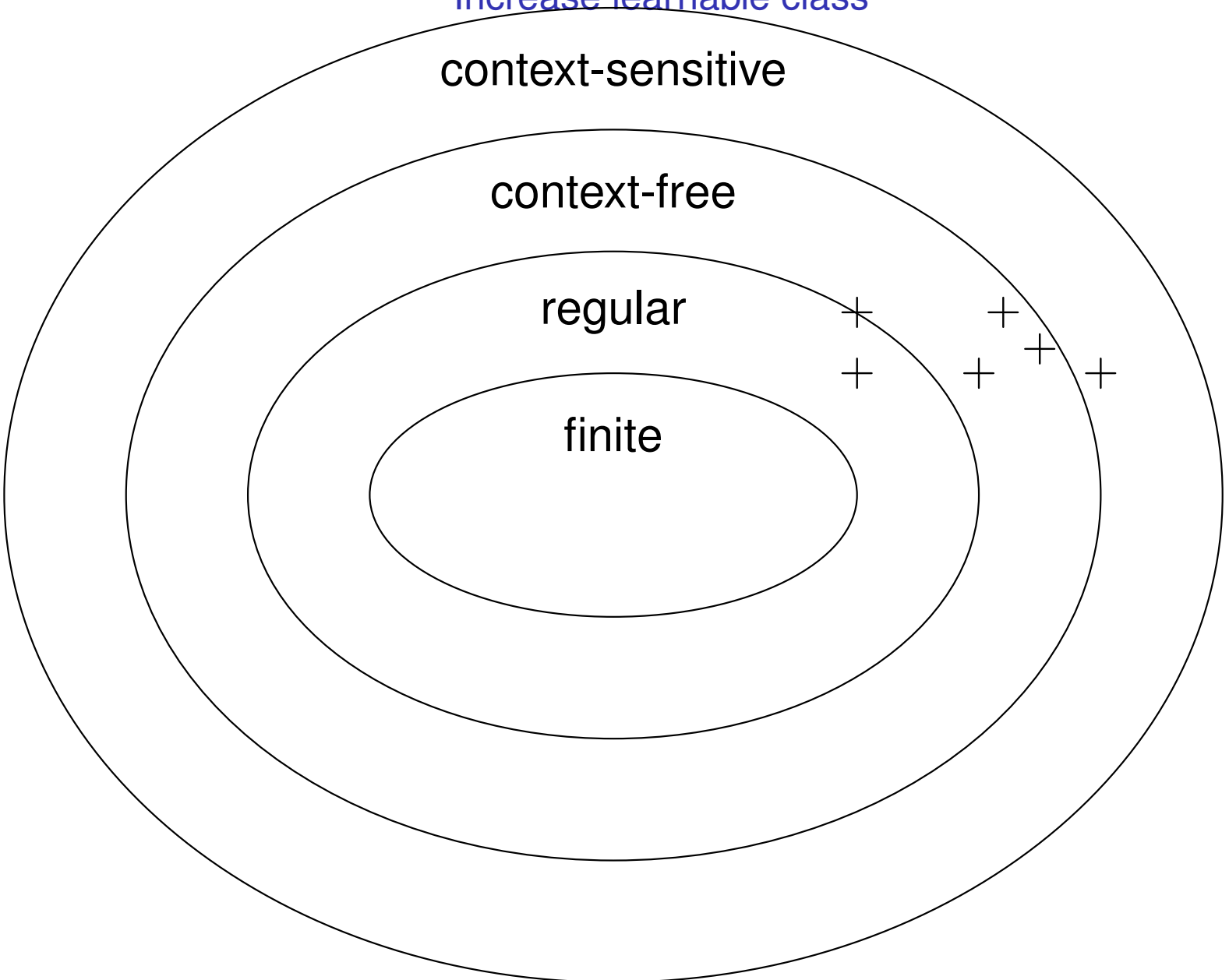
Alternative strategy

Increase learnable class



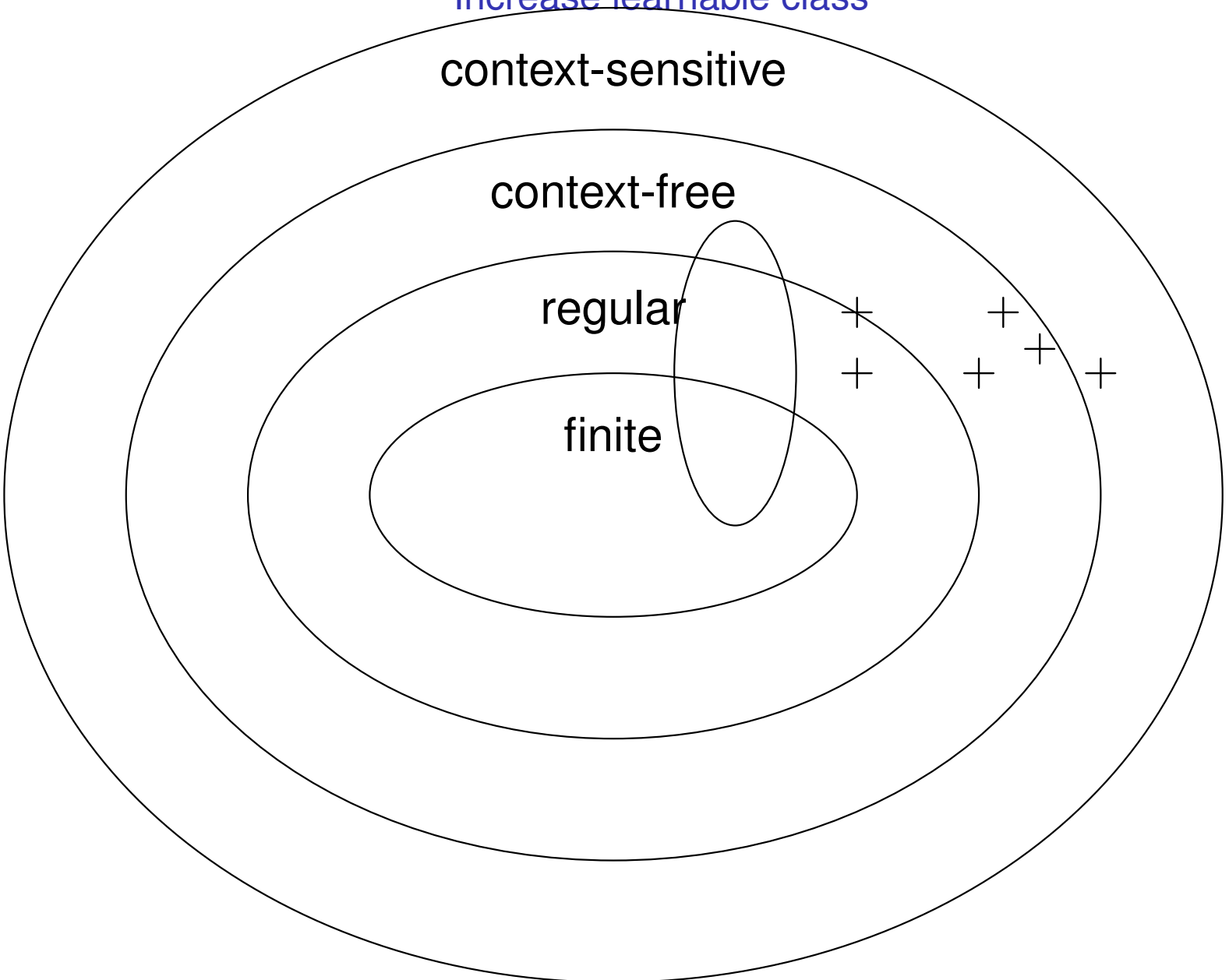
Alternative strategy

Increase learnable class



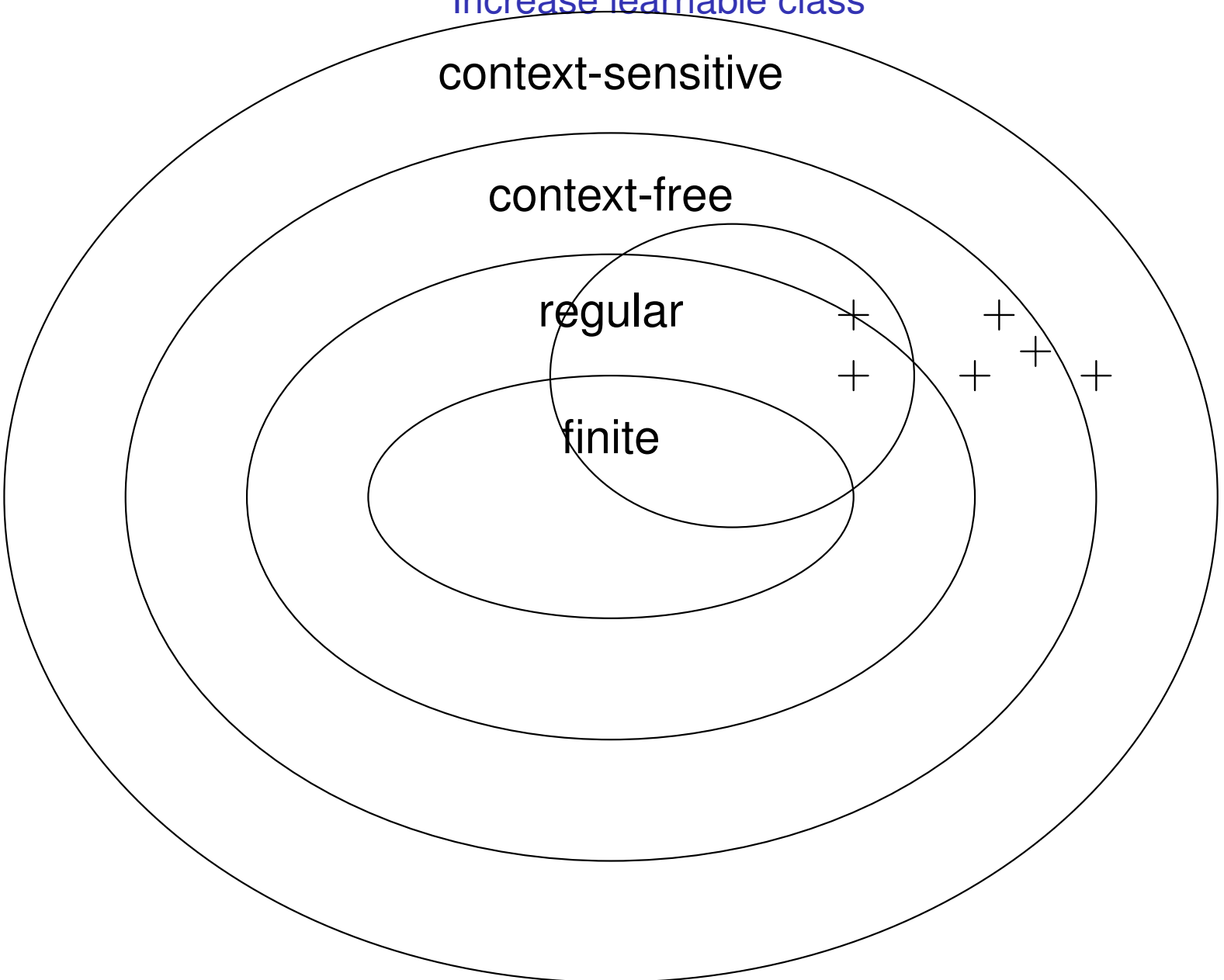
Alternative strategy

Increase learnable class



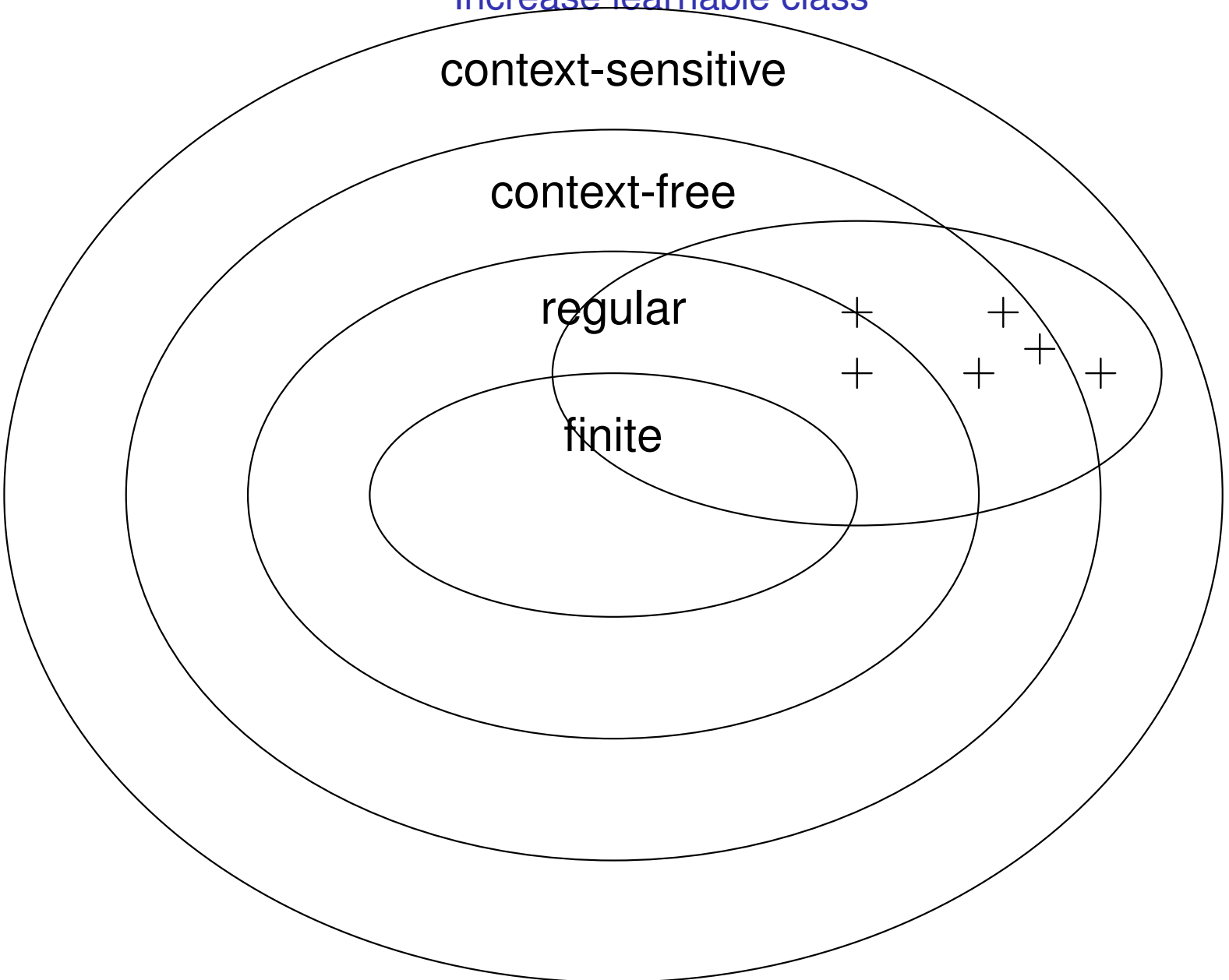
Alternative strategy

Increase learnable class

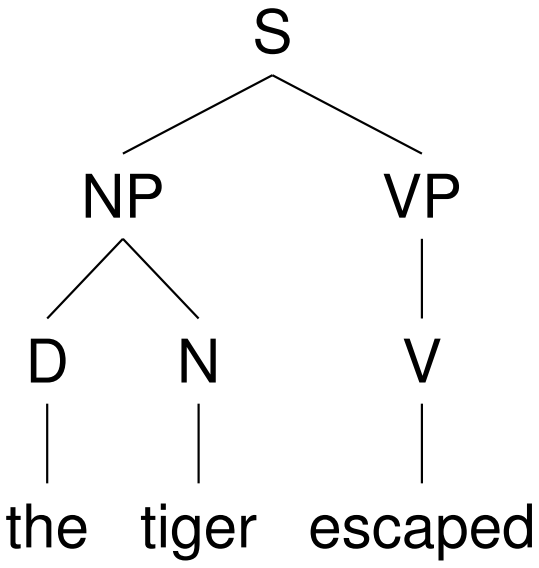


Alternative strategy

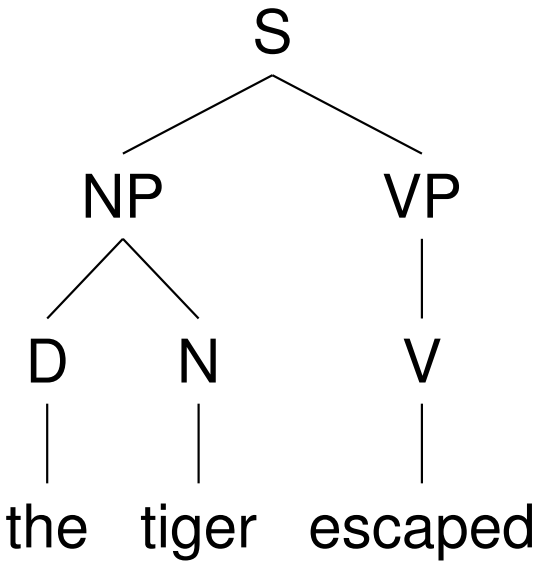
Increase learnable class



Context free grammars



Context free grammars



Good questions

- What does the symbol NP mean?
- What does a production NP $\rightarrow$ D N mean?

A partial answer

- A nonterminal is/represents a congruence class

$$[u]$$

- A production represents an inclusion relation between these classes:

$$[u] \supseteq [v][w]$$

Distribution

Full context

Context (or *environment*)

A context is just a sentence with a gap $l\square r$.
We use $\odot$ for the wrap operation:

$$(l\square r) \odot u = lur$$

Distribution of a string

Given a language L
$u^{\triangleright} = \{l\square r | lur \in L\}$

Limitations of congruence classes

Complete substitutability is too strong:

- can: I can run fast, I want a can of beans
- may: I may run fast, I love Paris in May
- would: I would run fast.
- jar: I want a jar of beans. I seem to jar my elbow often.

Congruence classes form a partition; whereas we must have categories that overlap.

Special congruence classes

- (0) The class of strings that don't occur as substrings anywhere in the language. ($u^{\triangleright} = \emptyset$).
- (1) The class of strings that are congruent to the empty string ($[\lambda]$).

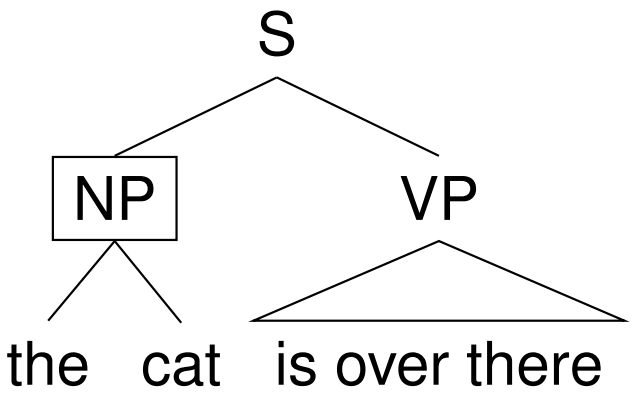
Special congruence classes

- (0) The class of strings that don't occur as substrings anywhere in the language. ($u^{\triangleright} = \emptyset$).
- (1) The class of strings that are congruent to the empty string ($[\lambda]$).

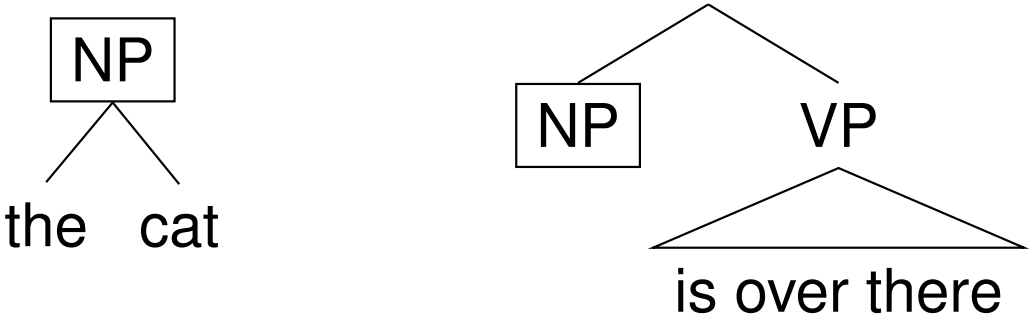
$L = \{ab, aabb, aaabbb, \dots\}$

- What is 0?
- What is 1?

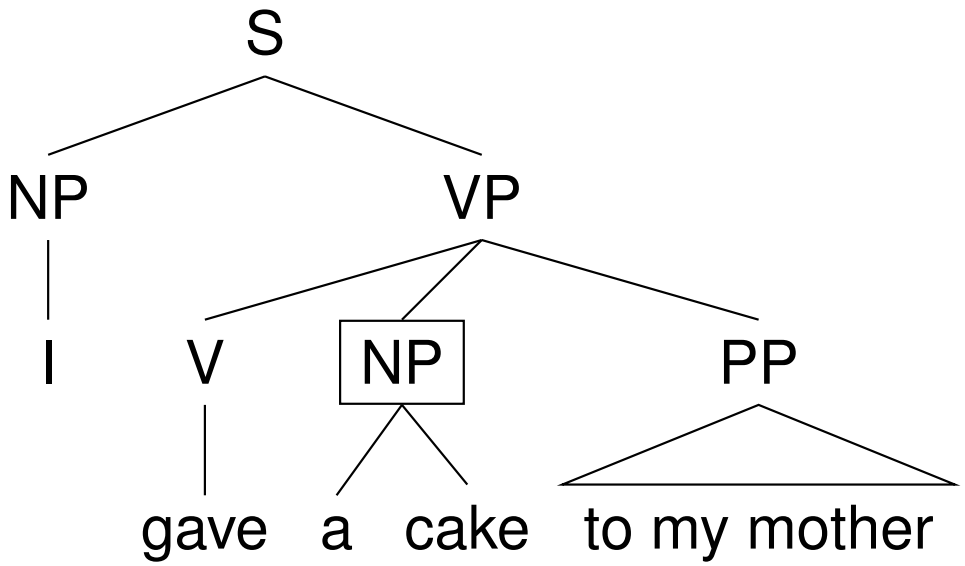
Trees



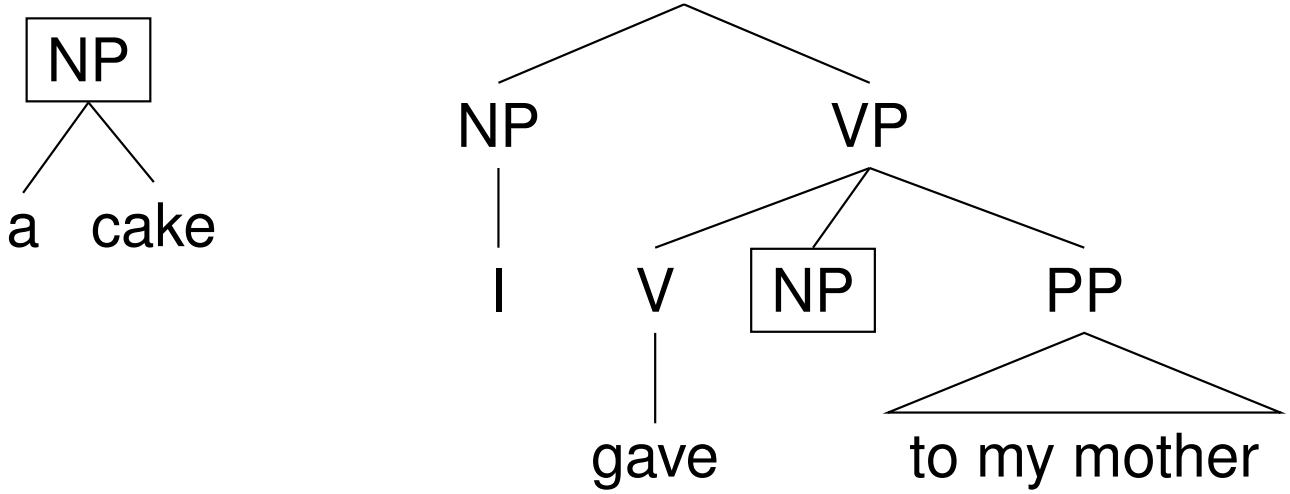
Trees



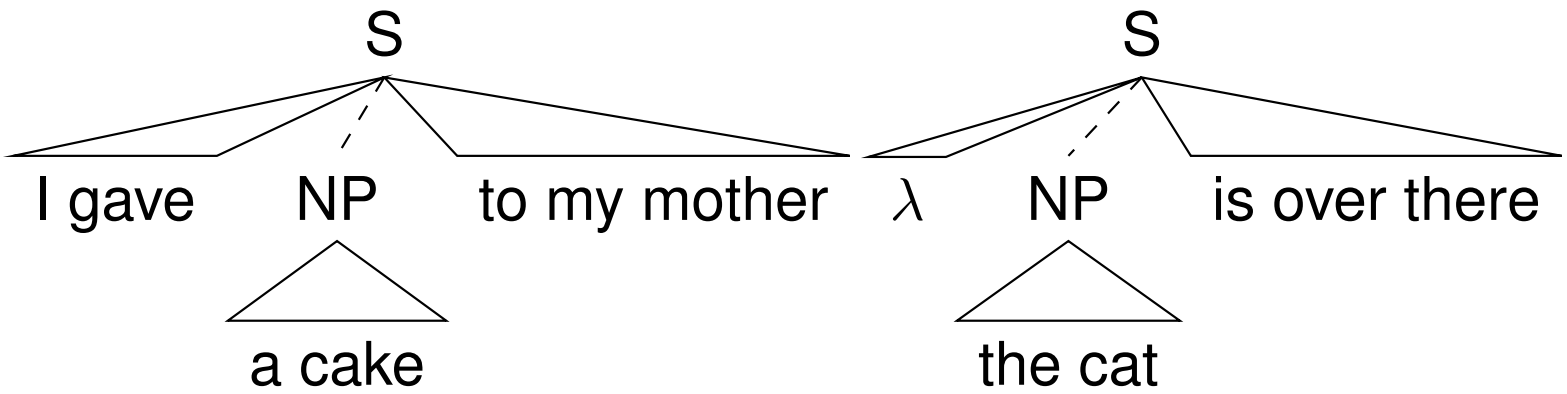
Trees



Trees



Contexts



Definition

Filling the hole

$l \Box r \odot u = lur$

A factorisation of a language L

C is a set of contexts; S is a set of strings

$$C \odot S \subseteq L$$

Context free grammars

Contexts and yields

$\mathcal{L}(G, N) = \{w \in \Sigma^* \mid N \stackrel{*}{\Rightarrow} w\}$

$\mathcal{C}(G, N) = \{l \Box r \mid S \stackrel{*}{\Rightarrow} lNr\}.$

Nonterminals in a context-free grammar

$$\mathcal{C}(G, N) \odot \mathcal{L}(G, N) \subseteq L$$

Contexts and yields of nonterminals

Yield of a non-terminal

$\mathcal{L}(NP)$ is the set of all strings w such that $NP \stackrel{*}{\Rightarrow} w$

$\mathcal{L}(NP) = \{$ the cat, the dog, some blue boxes $\ldots \}$

Contexts of a non-terminal

$\mathcal{C}(NP)$ is the set of all contexts (l, r) such that $S \stackrel{*}{\Rightarrow} lNPr$

$\mathcal{C}(NP) = \{ \square$ is over there, I want $\square$, Put it on $\square \ldots \}$

Any string in $\mathcal{L}(NP)$ can occur in any context in $\mathcal{C}(NP)$

A difficult question

Suppose we have some string, say 'the cat', which is in $\mathcal{L}(NP)$

Question

What is the relationship between the distribution of 'the cat'
the cat$^{\triangleright}$
and the contexts or distribution of NP: $\mathcal{C}(NP)$??

- If $NP \stackrel{*}{\Rightarrow}$ the cat
  then $\mathcal{C}(NP) \subseteq$ the cat$^{\triangleright}$

- If $NP \stackrel{*}{\Rightarrow}$ the cat
  then $\mathcal{C}(NP) \subseteq$ the cat$^{\triangleright}$

I installed the cat flap yesterday. The cat food has run out.

Key step

Congruence classes

Strings that have exactly the same distribution.
Based on the distribution of an **individual** string.

Syntactic concepts

Sets of strings that have similar distributions.
Based on the distribution of a **set** of strings.

Distribution

Full context

Distribution of a string in a language

$u^{\triangleright} = \{l \square r | lur \in L\}$

Distribution of a set of strings

W is a set of strings:

$W^{\triangleright} = \{l \square r |$ for all $u \in W, lur \in L\}$

Example

$$\text{CAN}^{\triangleright} = \{\dots\}$$

$$\text{MAY}^{\triangleright} = \{\dots\}$$

$$\{\text{MAY}, \text{CAN}\}^{\triangleright} = \text{CAN}^{\triangleright} \cap \text{MAY}^{\triangleright}$$

Example

$$\text{CAN}^{\triangleright} = \{\dots\}$$

$$\text{MAY}^{\triangleright} = \{\dots\}$$

$$\{\text{MAY}, \text{CAN}\}^{\triangleright} = \text{CAN}^{\triangleright} \cap \text{MAY}^{\triangleright}$$

$$\{\text{MAY}, \text{CAN}\}^{\triangleright} \subset \text{CAN}^{\triangleright}$$

In general if $X \subseteq Y$ then $Y^{\triangleright} \subseteq X^{\triangleright}$.

Dual map

Given a context $l \square r$ we can think of the set of strings that occur in that context.

$$(l \square r)^{\triangleleft} = \{u \in \Sigma^* : lur \in L\}$$

Dual map

Given a context $l \square r$ we can think of the set of strings that occur in that context.

$$(l \square r)^{\triangleleft} = \{u \in \Sigma^* : lur \in L\}$$

If C is a set of contexts:

$$C^{\triangleleft} = \{u \in \Sigma^* \mid \forall l \square r \in C, lur \in L\} \quad (1)$$

Circularity

Take a set of strings W, perhaps only a single string $\{w\}$

- $W^{\triangleright}$ is a set of contexts
- $W^{\triangleright\triangleleft}$ is a set of strings
- $W^{\triangleright\triangleleft\triangleright}$ is a set of contexts
- $W^{\triangleright\triangleleft\triangleright\triangleleft}$ is a set of strings
- ...

Circularity

Take a set of strings W, perhaps only a single string $\{w\}$

- $W^{\triangleright}$ is a set of contexts
- $W^{\triangleright\triangleleft}$ is a set of strings
- $W^{\triangleright\triangleleft\triangleright}$ is a set of contexts
- $W^{\triangleright\triangleleft\triangleright\triangleleft}$ is a set of strings
- ...

Facts

$$W^{\triangleright} = W^{\triangleright\triangleleft\triangleright}$$

$$W \subseteq W^{\triangleright\triangleleft} = W^{\triangleright\triangleleft\triangleright\triangleleft}$$

Closed sets of strings

- A set of strings W is closed if $W = W^{\triangleright\triangleleft}$; these are generalisations of the congruence classes.
- We will also call closed sets of strings *syntactic concepts*.
- $[w]$ is the set of all strings that have exactly the same distribution as w.
- $W^{\triangleright\triangleleft}$ is the set of all strings whose distribution includes the shared distribution of W ($W^{\triangleright}$).

Example

- could$^{\triangleright} = \{\ldots, \text{I}\square\text{have been a contender}, \ldots\}$
- could$^{\triangleright\triangleleft} = \{\text{could}, \text{would}, \text{might}, \ldots\}$
- But might$^{\triangleright} = \text{could}^{\triangleright} \cup \{\square\text{is right}\} \cup \ldots$

Example

- could$^{\triangleright} = \{\ldots, \text{I}\square\text{have been a contender}, \ldots\}$
- could$^{\triangleright\triangleleft} = \{\text{could}, \text{would}, \text{might}, \ldots\}$
- But might$^{\triangleright} = \text{could}^{\triangleright} \cup \{\square\text{is right}\} \cup \ldots$

Even for single words closed sets of strings may be bigger than congruence classes.

Lambek, 1958

"We shall assign type n to all expressions which can occur in any context in which all proper names can occur."

- Let N be the set of proper names (not necessarily closed)
- $N^{\triangleright}$ is then the set of contexts that all proper nouns can occur in.
- $N^{\triangleright\triangleleft}$ is the set of all strings which can occur in any of $N^{\triangleright}$
- n is assigned to $N^{\triangleright\triangleleft}$ which is a closed set.

Lambek, 1958

"We shall assign type n to all expressions which can occur in any context in which all proper names can occur."

- Let N be the set of proper names (not necessarily closed)
- $N^{\triangleright}$ is then the set of contexts that all proper nouns can occur in.
- $N^{\triangleright\triangleleft}$ is the set of all strings which can occur in any of $N^{\triangleright}$
- n is assigned to $N^{\triangleright\triangleleft}$ which is a closed set.

Historically the types in Lambek grammar were intended to be closed sets of strings.

Maximal decompositions into contexts and substrings

Definition

A *syntactic concept* is a pair $\langle W, C \rangle$ where

- where W is a set of strings, and C is a set of contexts
- $C \odot W \subseteq L$
- C and W are both maximal

In this case $W = C^{\triangleleft}$ and $C = W^{\triangleright}$ and W is closed.

Special concepts

- Top: $\top$, Σ^*.
- Bottom, zero, $\emptyset^{\triangleright\triangleleft}$ which is usually empty.
- Unit: $\lambda^{\triangleright\triangleleft}$ which is usually just λ

Special concepts

- Top: $\top$, Σ^*.
- Bottom, zero, $\emptyset^{\triangleright\triangleleft}$ which is usually empty.
- Unit: $\lambda^{\triangleright\triangleleft}$ which is usually just λ
- L the language itself!

Special concepts

- Top: $\top$, Σ^*.
- Bottom, zero, $\emptyset^{\triangleright\triangleleft}$ which is usually empty.
- Unit: $\lambda^{\triangleright\triangleleft}$ which is usually just λ
- L the language itself!

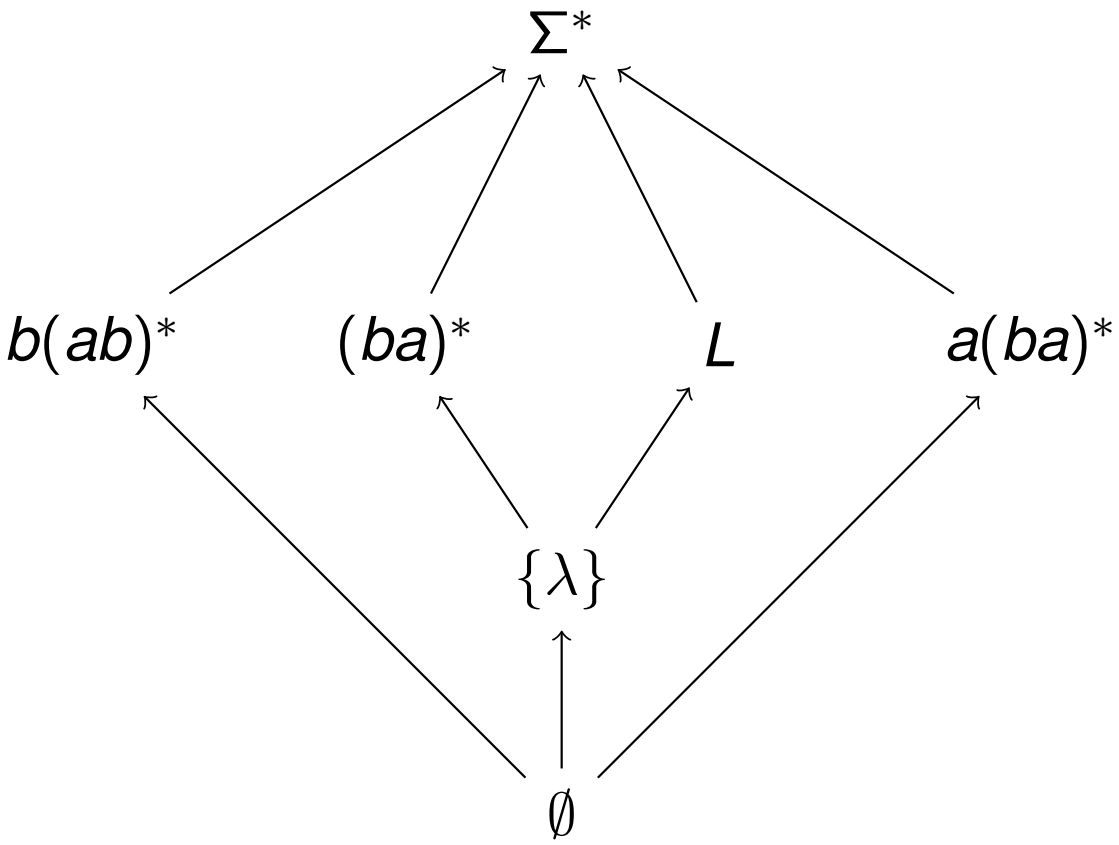
Why is L closed?

$$\square \in L^{\triangleright}$$

So everything in $L^{\triangleright\triangleleft}$ must be in $\square^{\triangleleft} = L$.

Language is regular iff lattice is finite

$L = (ab)^*$



Lexical Ambiguity

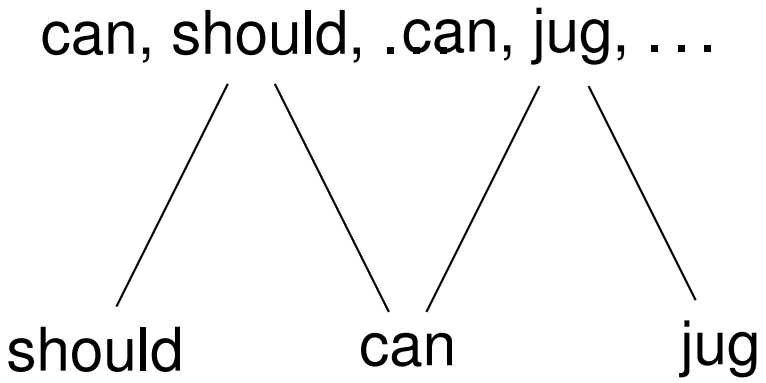
Example

CAN is both a modal auxiliary and a count noun (and a verb, ...)

Assume the only rules introducing it are:

- AUX $\rightarrow$ CAN
- N $\rightarrow$ CAN

$$\text{CAN}^{\triangleright} = \mathcal{C}(\text{AUX}) \cup \mathcal{C}(\text{N})$$


can, should, .can, jug, . . .
should
can
jug

Correctness of productions

$$X \to YZ$$

$$\mathcal{L}(X) \supseteq \mathcal{L}(Y) \cdot \mathcal{L}(Z)$$

Correctness of productions

$$X \to YZ$$

$$\mathcal{L}(X) \supseteq \mathcal{L}(Y) \cdot \mathcal{L}(Z)$$

If $X \supseteq Y \cdot Z$ then

$$X^{\triangleright\triangleleft} \supseteq Y^{\triangleright\triangleleft} \cdot Z^{\triangleright\triangleleft}$$

Productions

Observation

If $X \supseteq Y \cdot Z$ then $X^{\triangleright\triangleleft} \supseteq Y^{\triangleright\triangleleft} \cdot Z^{\triangleright\triangleleft}$

Productions

Observation

If $X \supseteq Y \cdot Z$ then $X^{\triangleright\triangleleft} \supseteq Y^{\triangleright\triangleleft} \cdot Z^{\triangleright\triangleleft}$

Add production

$[\![X]\!] \to [\![Y]\!][\![Z]\!]$

[Clark, 2013]

Mergeable nonterminals

If

$$\mathcal{L}(G, M)^{\triangleright\triangleleft} = \mathcal{L}(G, N)^{\triangleright\triangleleft}$$

then we can merge M and N without increasing the language defined by G,

[Clark, 2013]

Mergeable nonterminals

If

$$\mathcal{L}(G, M)^{\triangleright\triangleleft} = \mathcal{L}(G, N)^{\triangleright\triangleleft}$$

then we can merge M and N without increasing the language defined by G,

Minimal grammars correspond to maximal factorisations

A grammar without mergable nonterminals will have nonterminals that correspond to syntactic concepts.

All CFLs

The minimal grammar will have nonterminals that are closed sets of strings.

Substitutable CFLs

- The minimal grammar will have nonterminals that are congruence classes.

All CFLs

The minimal grammar will have nonterminals that are closed sets of strings.

Substitutable CFLs

- The minimal grammar will have nonterminals that are congruence classes.
- In a substitutable language the congruence classes are all closed sets of strings and nearly all closed sets of strings are congruence classes.

Learning model

Gold style positive examples plus MQs

Membership queries

The learner can query whether a string w is in the language or not.

Controversial!

Learning model

Gold style positive examples plus MQs

Membership queries

The learner can query whether a string w is in the language or not.

Controversial!

Two justifications:

- Direct justification
- Probabilistic justification

Membership queries to probabilistic learning

Membership queries

- Possible string w
- Possible context $l\Box r$
- The learner can ask whether $lwr \in L$

Corresponds to the ability of the child to interact and generate new utterances.

Membership queries to probabilistic learning

Simplified version of probabilistic learning

- A string w that occurs with probability $> \epsilon_1$
- A context $l\Box r$ that occurs with probability $> \epsilon_2$
- Then lwr should occur more frequently than $f(\epsilon_1, \epsilon_2)$
- If we observe $\mathcal{O}(\frac{1}{f(\epsilon_1 \epsilon_2)})$ examples without seeing lwr then lwr is probably ungrammatical.

"Indirect negative evidence?"

Computationally efficient probabilistic learning results

A problem in statistical learning of no linguistic interest

All Regular languages

Clark and Thollard (2004)

Some Context-free languages

Clark (2006)
Luque and Infante-Lopez (2010)
Shibata and Yoshinaka (2013)

Context-sensitive results?

No efficient ones yet

Computationally efficient probabilistic learning results

A problem in statistical learning of no linguistic interest

All Regular languages

Clark and Thollard (2004)

Some Context-free languages

Clark (2006)
Luque and Infante-Lopez (2010)
Shibata and Yoshinaka (2013)

Context-sensitive results?

No efficient ones yet
Inefficient: Angluin (1988), Chater and Vitányi (2007)

Representing a concept

We need to be able to represent these concepts finitely: there are two approaches:

- (dual approach) Pick a finite set of contexts C to represent $C^{\triangleleft}$
- (primal approach) Pick a finite set of strings W to represent $W^{\triangleright\triangleleft}$

These give two closely related learning algorithms.

Representing a concept

We need to be able to represent these concepts finitely: there are two approaches:

- (dual approach) Pick a finite set of contexts C to represent $C^{\triangleleft}$
- (primal approach) Pick a finite set of strings W to represent $W^{\triangleright\triangleleft}$

These give two closely related learning algorithms.
Explain a new word to someone: MLE: *bare*.

Dual algorithms

Representation

Each nonterminal in the grammar is represented by a small ($\leq k$) set of contexts:

- Small set of contexts A
- defines a nonterminal $[\![A]\!]$
- which we want to generate all the strings in $A^{\triangleleft}$

Dual algorithms

Representation

Each nonterminal in the grammar is represented by a small ($\leq k$) set of contexts:

- Small set of contexts A
- defines a nonterminal $[\![A]\!]$
- which we want to generate all the strings in $A^{\triangleleft}$

Special case: the start symbols S is always represented by the single context $\{\Box\}$.

Finite context property

Given a nonterminal N:

Can we pick out a finite set of contexts C such that the set of strings generated by N is just the set of strings that can occur in all of the contexts C?

Finite context property

Given a nonterminal N:

Can we pick out a finite set of contexts C such that the set of strings generated by N is just the set of strings that can occur in all of the contexts C?

- If we can do this for every nonterminal in the grammar, then the grammar has the Finite Context Property (FCP).
- If we can do it using at most s contexts for each nonterminal, then it has the s-FCP.

Toy example

that cat

Toy example

that cat

Maybe $C = \{\square \text{IS CRAZY}\}$?

- that cat
- Alex's hamster
- that dog is nice but that cat

Toy example

that cat

Maybe $C = \{\text{I LOVE}\square\}$?

- that cat
- Alex's hamster
- that dog but I hate all cats

Toy example

that cat

$C = \{\square$IS CRAZY, I LOVE$\square\}$?

- that cat
- Alex's hamster

Productions

Correctness of a production

Suppose we have nonterminals $[\![A]\!], [\![B]\!], [\![C]\!]$:
$[\![A]\!] \to [\![B]\!][\![C]\!]$ is correct iff $A^\triangleleft \supseteq B^\triangleleft C^\triangleleft$

- The correctness of each production can be judged independently.
- If it is not correct then there are some strings u and v such that $u \in B^\triangleleft$, and $v \in C^\triangleleft$ and some context $l \Box r \in A$ such that $luvr$ is not in L.
- Given a set of strings K, we can test the correctness relative to K using MQs.
- The larger K is the better the test is.

Trivial example

$L = \{a^n b^n \mid n > 0\} = \{ab, aabb, \dots\}$

Suppose we have nonterminals $[\![\Box]\!]$, $[\![\Box b]\!]$, $[\![a\Box]\!]$.

Candidate production

$[\![\Box]\!] \to [\![\Box b]\!][\![a\Box]\!]$

Trivial example

$L = \{a^n b^n \mid n > 0\} = \{ab, aabb, \dots\}$

Suppose we have nonterminals $[\![\Box]\!]$, $[\![\Box b]\!]$, $[\![a\Box]\!]$.

Candidate production

$[\![\Box]\!] \to [\![\Box b]\!][\![a\Box]\!]$

Counterexample:

- $aab \in \{\Box b\}^\triangleleft$
- $abb \in \{a\Box\}^\triangleleft$
- But $aababb$ is NOT in $\Box^\triangleleft = L$

English example

NP $\rightarrow$ D N

Nonterminals

- A: □is over there
- B □cat is over there
- C the □is over there

$A \rightarrow BC$

English example

NP $\rightarrow$ D N

Nonterminals

- A: □is over there ,I saw □
- B □cat is over there ,I saw □cat
- C the □is over there ,I saw the □

$A \rightarrow BC$

Overview

Maintain a set of contexts F and a set of strings K: construct a grammar $\mathcal{G}(K, L, F)$:

- Define nonterminals using small sets of contexts.
- Use strings to eliminate incorrect rules.

Overview

Maintain a set of contexts F and a set of strings K: construct a grammar $\mathcal{G}(K, L, F)$:

- Define nonterminals using small sets of contexts.
- Use strings to eliminate incorrect rules.

Monotonicity wrt contexts

If $F \subseteq G$ are two sets of contexts then
$\mathcal{L}(\mathcal{G}(K, L, F)) \subseteq \mathcal{L}(\mathcal{G}(K, L, G))$

Anti-Monotonicity with strings

If $J \subseteq K$ then $\mathcal{L}(\mathcal{G}(J, L, F)) \supseteq \mathcal{L}(\mathcal{G}(K, L, F))$

Primal

Representation

Each nonterminal in the grammar is represented by a small ($\leq k$) set of strings:

- Small set of strings W
- defines a nonterminal $[\![W]\!]$
- which we want to generate all the strings in $W^{\triangleright\triangleleft}$

Primal

Representation

Each nonterminal in the grammar is represented by a small ($\leq k$) set of strings:

- Small set of strings W
- defines a nonterminal $[\![W]\!]$
- which we want to generate all the strings in $W^{\triangleright\triangleleft}$

The larger the set W, the larger the set of strings $W^{\triangleright\triangleleft}$.

Use contexts to eliminate incorrect rules.

Primal algorithms

$$[\![A]\!] \to [\![B]\!][\![C]\!]$$

$$A^{\triangleright\triangleleft} \supseteq B^{\triangleright\triangleleft} C^{\triangleright\triangleleft}$$

Primal algorithms

$$[\![A]\!] \to [\![B]\!][\![C]\!]$$

$$A^{\triangleright\triangleleft} \supseteq B^{\triangleright\triangleleft} C^{\triangleright\triangleleft}$$

Incorrect if there is some context $l \Box r \in A^{\triangleright}$ and strings $u \in B, v \in C$ such that

- $luvr$ is not in L,

Theorem

This algorithm learns all context-free languages that have the FCP:

- Efficiently
- Correctly
- Weakly
- Using MQs.

Theorem

This algorithm learns all context-free languages that have the FCP:

- Efficiently
- Correctly
- Weakly
- Using MQs.

Discussion

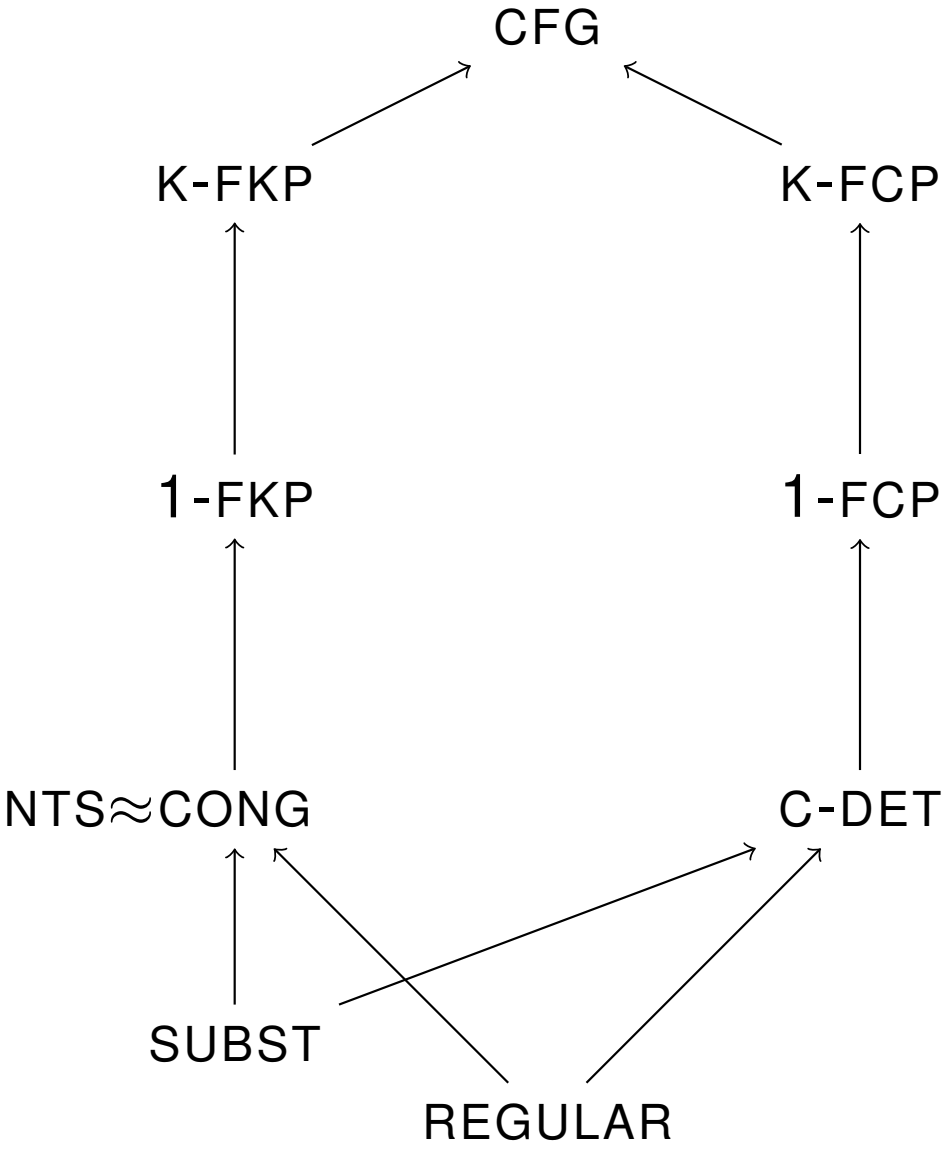
Is this really how children learn language?

Learnable languages

Weak learning, with MQs

- All finite languages
- All regular languages
- Not quite all context-free languages

Diagram



Unlearnable languages

Example

$L = \{a^n b^m \mid n \neq m\}$

$\{a, b, aab, abbb, aaaaabbb, \ldots\}$

Unlearnable languages

Example

$L = \{a^n b^m \mid n \neq m\}$

$\{a, b, aab, abbb, aaaaabbb, \ldots\}$

Why?

To represent this we need to use concepts that cannot be referred to by a finite number of contexts, or a finite number of strings.

Do these occur in natural language?

Bibliography I


Clark, A. (2013).
The syntactic concept lattice: Another algebraic theory of the context-free languages?
Journal of Logic and Computation.