Learnability and Language Acquisition

Computational Learning of Syntax

Alexander Clark

Department of Philosophy
King's College, London

LSA Summer Institute 2015, Chicago

What is the course about?

- How to learn grammars from strings.
- The relation between these methods and
  - Language acquisition
  - Linguistic theory in general

Classic model

Three questions

1. What are the inputs? (the PLD)
2. What are the outputs? (the $\hat{G}$)
3. What conditions does the LAD have to satisfy?

Course Outline

`http://www.cs.rhul.ac.uk/home/alexc/lsa2015/index.html`

- Mon A first learning algorithm. Basic principles of learnability.
- Thur Distributional analysis. Learning Context-free grammars;
- Mon Mildly context sensitive grammars and beyond. Copying.
- Thu Strong learning; language acquisition and linguistic theory.

Topics for today

- The role of learnability in linguistics.
- A first learning algorithm.
- Basic principles of learning:
  - Inputs and Outputs
  - Convergence
  - Computational complexity

Central problem of linguistics

Chomsky's questions

1. What constitutes knowledge of a language?
2. How is this knowledge acquired by its speakers?

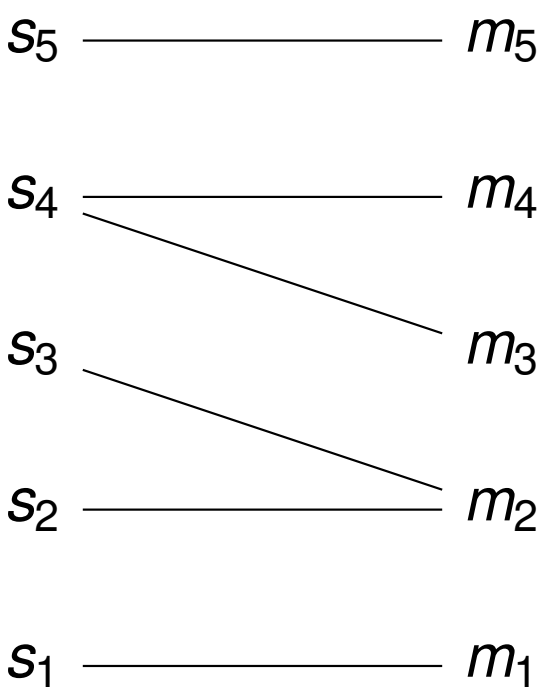
Jackendoff 2011

1. An account of speakers' ability to create and understand an unlimited number of sentences of their language(s). (knowledge of a language or competence)
2. An account of how speakers acquire knowledge of a language. (acquisition)
3. An account of how the human species acquired the ability to acquire the knowledge of a language. (evolution)

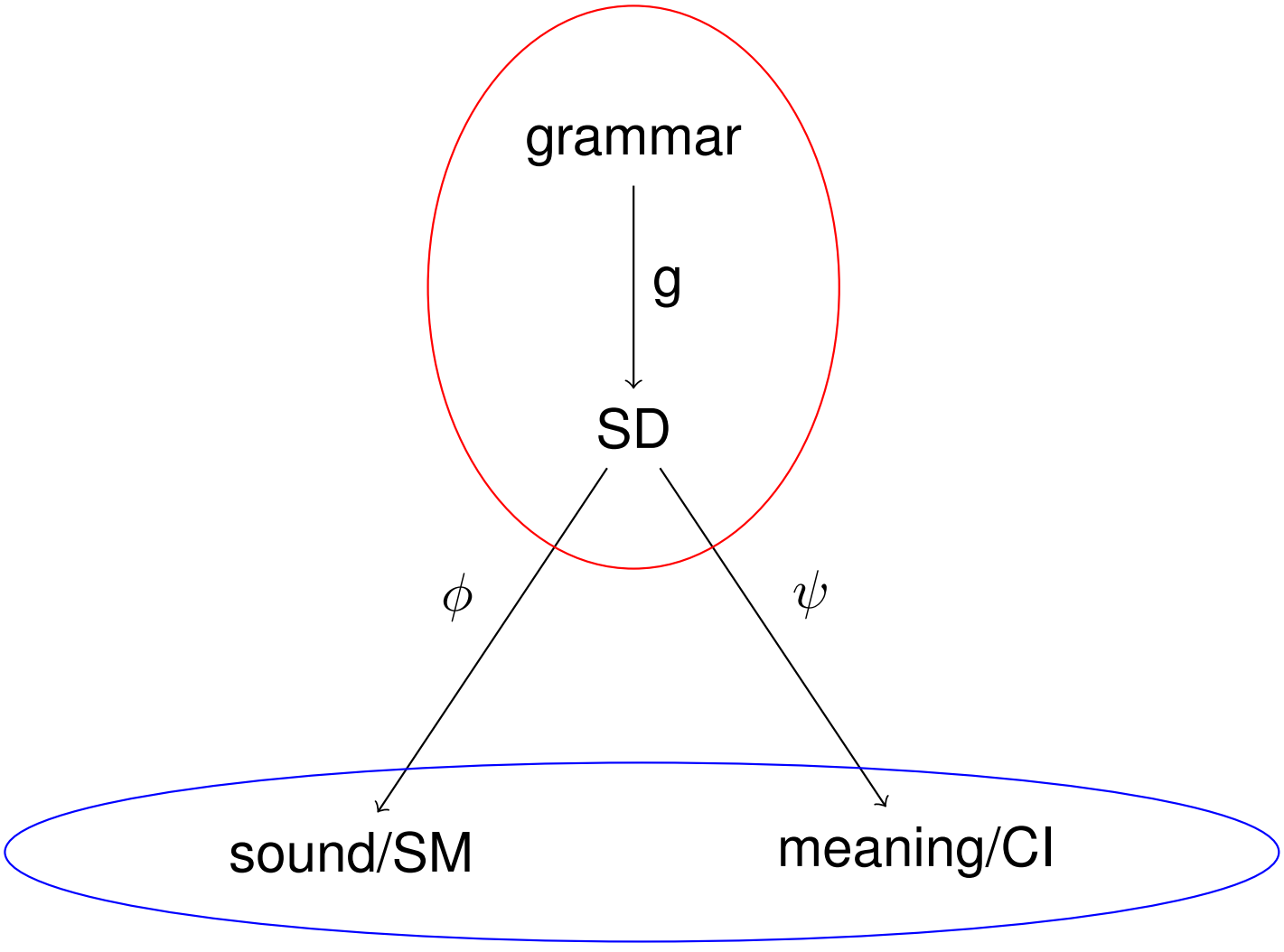
Knowledge of language

- Grammars (I-languages): finite generative devices
- Languages (E-languages): infinite sets of
  - sound/meaning pairs
  - sequences of acoustic categories/words/. . .

Sound/meaning pairs



Architecture



Inputs to the learning algorithm

Three classes of objects: strings, meanings, and trees.
Accessibility for the child:

- Strings – complete
- Meanings – partial
- Trees – no information at all

The scientific question

How do children acquire language?

- It's complicated.
- Isolate the possible contribution of one information source: *surface level distributional features*.
- We are going to take an idealized mathematical perspective.

A great quote from Partha Niyogi

Another aspect of the book is its focus on mathematical models where the relationship between various objects may be formally (provably) studied. A complementary approach is to consider the larger class of computational models where one resorts to simulations. Mathematical models with their equations and proofs, and computational models with their equations and simulations provide different and important windows of insight into the phenomena at hand.

In the first, one constructs idealized and simplified models but one can now reason precisely about the behavior of such models and therefore be very sure of one's conclusions. In the second, one constructs more realistic models but because of the complexity, one will need to resort to heuristic arguments and simulations. In summary, for mathematical models the assumptions are more questionable but the conclusions are more reliable - for computational models, the assumptions are more believable but the conclusions more suspect.

Two strategies

Mathematical model

- Define a specific learning algorithm.
- Prove that the algorithm is correct for some class of grammars/languages.

Computational models

- Natural language corpora (perhaps CHILDES)
- Run computer program.
- Evaluate the output in some way.

Weak learning

- We just consider a language as a set of strings L.
- We want to converge by learning a grammar that generates the same set of strings as L.
- We will just use positive data: strings of examples.

Important questions

- What are these strings of? what are the terminal symbols?
- What does the set of strings represent?
- Do we want to use a set rather than a distribution?

Notation

- Finite alphabet

$$\Sigma$$

- Set of all finite strings over this:

$$\Sigma^*$$

- Set of all nonempty finite strings over this:

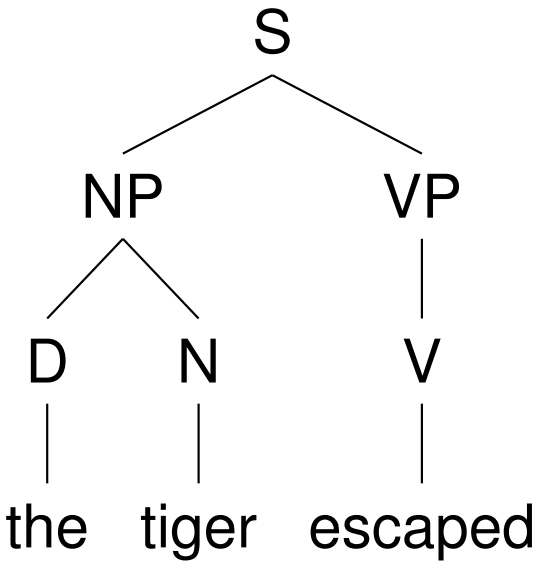
$$\Sigma^+$$

- Empty string λ
- A formal language:

$$L \subseteq \Sigma^*$$

(Normally $L \subset \Sigma^+$)

Context free grammars



Good questions

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

Context free grammars

Notation

A context free grammar G is a collection of finite sets:

- Σ a set of terminal symbols
- V a set of nonterminal symbols
- $S \in V$ a start symbol
- P a set of productions $N \to \alpha$ where
  - $N \in V$
  - α is a finite string of symbols from Σ and V

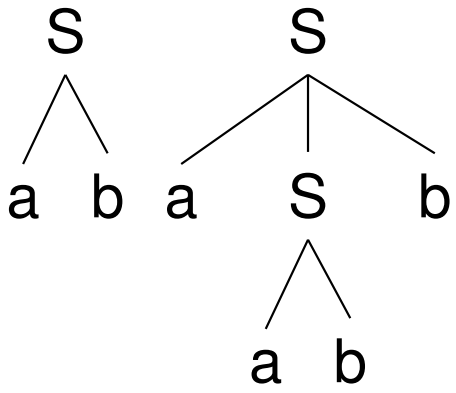
We write the derivation process using $N \stackrel{*}{\Rightarrow}_G w$, and

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

$$\mathcal{L}(G) = \mathcal{L}(G, S)$$

Trivial example

- Terminal symbols a, b
- One nonterminal S
- $S \to ab$ and $S \to aSb$



$$\mathcal{L}(G) = \{ab, aabb, aaabbb, \ldots\}$$

Propositional logic

$A, (\neg B), (A \vee (\neg B)), \ldots$

Alphabet

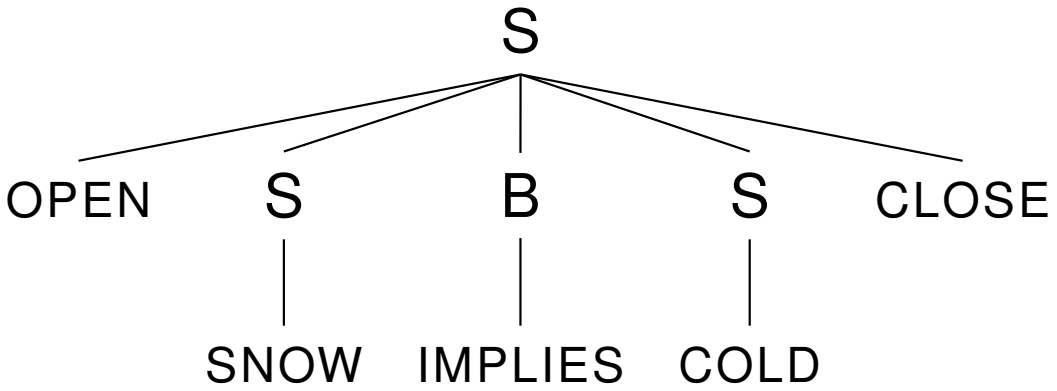
rain, snow, hot, cold, danger	$A_1, A_2, \ldots$
and, or, implies, iff	$\wedge, \vee, \rightarrow, \leftrightarrow$
not	$\neg$
open, close	$(,)$

- rain
- open snow implies cold close
- open snow implies open not hot close close

Grammar

- $\Sigma = \{\text{RAIN}, \text{SNOW}, \ldots, \text{OPEN}, \ldots \text{NOT}\}$
- $V = \{S, B\}$
- $P = \{S \rightarrow \text{RAIN}, S \rightarrow \text{SNOW}, S \rightarrow \text{OPEN } SBS \text{ CLOSE}, S \rightarrow \text{OPEN NOT} S \text{ CLOSE}, B \rightarrow \text{AND}, B \rightarrow \text{OR}, B \rightarrow \text{IMPLIES} \ldots\}$

Derivations



$$S \overset{*}{\Rightarrow}_G \text{OPEN SNOW IMPLIES COLD CLOSE}$$

Simple example, but still an infinite, non-regular language, which is hierarchically structured.

Classic model

Three questions

1. What are the inputs?: Strings
2. What are the outputs?: Context-free grammars
3. What conditions does the LAD have to satisfy?:
   - 3.1 Correctness
   - 3.2 Efficiency

Correctness of Weak Learners

- The learner must produce a grammar that is correct, if the PLD is big enough.
  - If the input is drawn from L_1 then it should produce a grammar for L_1
  - If the input is drawn from L_2 then it should produce a grammar for L_2
  - ...
- For every language L in some class of languages $\mathcal{L}$ the learner produces the right answer.
- This is called the *learnable class* for a learner.

Distributional learning

Several reasons to take distributional learning seriously:

- Cognitively plausible (Saffran et al. 1996, Mintz, 2002)
- It works in practice: large scale lexical induction (Curran, J. 2003)
- Linguists use it as a constituent structure test (Carnie, A, 2008)
- Historically, PSGs were intended to be the output from distributional learning algorithms.

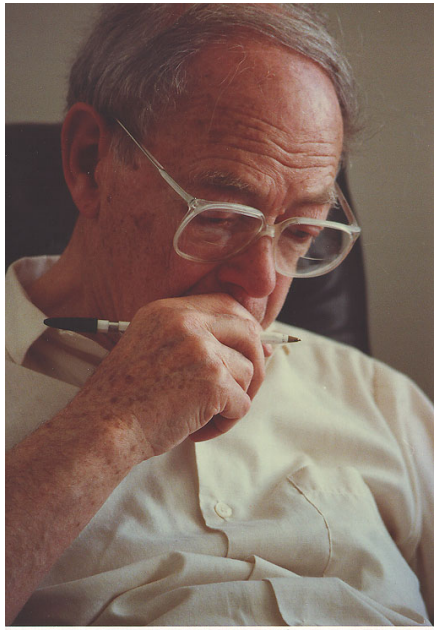
Chomsky (1968/2006)

“The concept of "phrase structure grammar" was explicitly designed to express the richest system that could reasonable be expected to result from the application of Harris-type procedures to a corpus.”

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



Example

Is 'cat' substitutable for 'dog'?

- The cat is over there.
- I want a dog for Christmas.
- I want a Siamese cat for Christmas.
- Put a cat-flap in the door to the kitchen.
- An Alsatian is a breed of dog.
- He continues to dog my footsteps.
- I would rather have a dog than a cat as a pet.

- The dog is over there.
- I want a cat for Christmas.
- I want a Siamese dog for Christmas.
- Put a dog-flap in the door to the kitchen.
- An Alsatian is a breed of cat.
- He continues to cat my footsteps.
- I would rather have a dog than a dog as a pet.
- I would rather have a cat than a dog as a pet.

Example

Distribution of “cat” in English

Infinite set of full contexts that ’cat’ can appear in :
“the □ is over there”
“I want a □ for Christmas”
. . .

- We can observe the distribution simply by looking at positive examples.
- We can see a similarity in distribution of “cat” and “dog”.
- Distributional learning is based on this idea.

Distributional Learning

[?]

- Look at the dog
- Look at the cat
- That cat is crazy
- That dog is crazy

English counterexample

- I can swim
- I may swim
- I want a can of beer
- *I want a may of beer

English counterexample

- She is Italian
- She is a philosopher
- She is an Italian philosopher
- *She is an a philosopher philosopher

Logic example

Propositional logic is *substitutable*:

- open rain and cold close
- open rain implies cold close
- open snow implies open not hot close
- open snow and open not hot close

Substitutable Languages

- $lur \in L$
- $lvr \in L$
- $l'ur' \in L$
- $\Rightarrow l'vr' \in L$

Formally

Definition

A language L is substitutable if for all strings $l, r, l', r' \in \Sigma^*$ and for all strings $u, v \in \Sigma^+$,

$$lur, lvr, l'ur' \in L \Rightarrow l'vr' \in L$$

Not all substitutable languages are context-free

Formally

The Syntactic Congruence

Two nonempty strings u, v are congruent ($u \equiv_L v$) if for all $l, r \in \Sigma^*$

$lur \in L \Leftrightarrow lvr \in L$

Complete mutual substitutability!

Congruences

A congruence is a relation which is well-behaved with respect to the structure of the space.

- The syntactic congruence is a congruence because if $u \equiv v$ then $u \cdot w \equiv v \cdot w$.
- It is a congruence with respect to the concatenation of strings.

For any strings u, v

$$[uv] \supseteq [u][v]$$

Proof

Suppose $u' \in [u], v' \in [v]$

- $uv \in L$
- $u'v \in L$
- $u'v' \in L$

Alternative Definition

L is substitutable if
$lur \in L, lvr \in L \Rightarrow u \equiv_L v$

- If we see two strings that share one context in common, then we assume they share all contexts.

A very restrictive property

Substitutable language

$\{a^n c b^n \mid n > 0\}$

Not a substitutable language

$\{a, aa\}$
(Since a and aa both occur in the context $\square$ but are not congruent)

Not a substitutable language

$\{a^n b^n \mid n > 0\}$
Since a and aab both occur in the context $\square b$.

Algorithm

- There is a very simple algorithm for learning all substitutable languages which are also context-free languages.
- We will explain it with an illustrative example.

Example

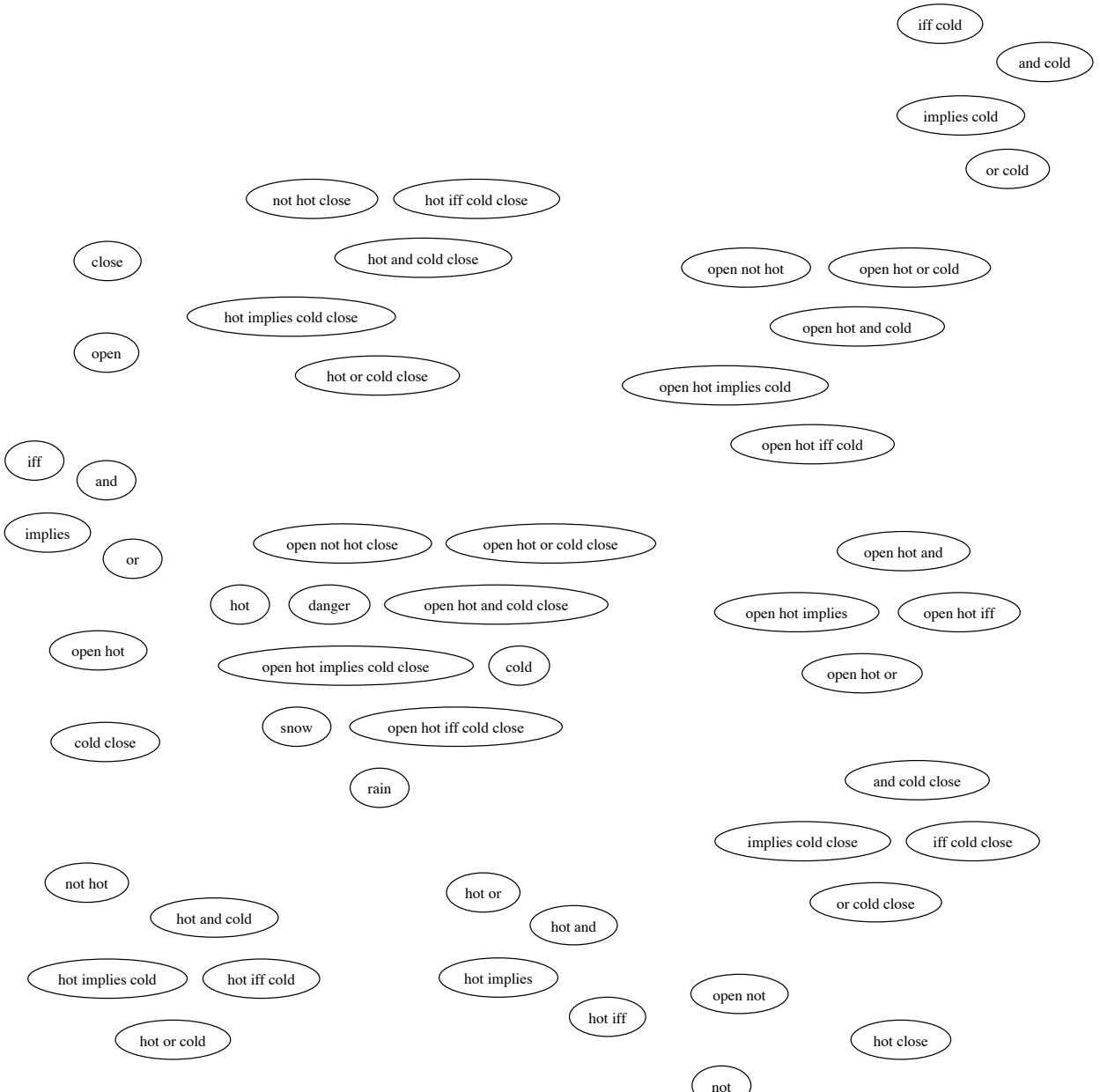
Input data $D \subseteq L$

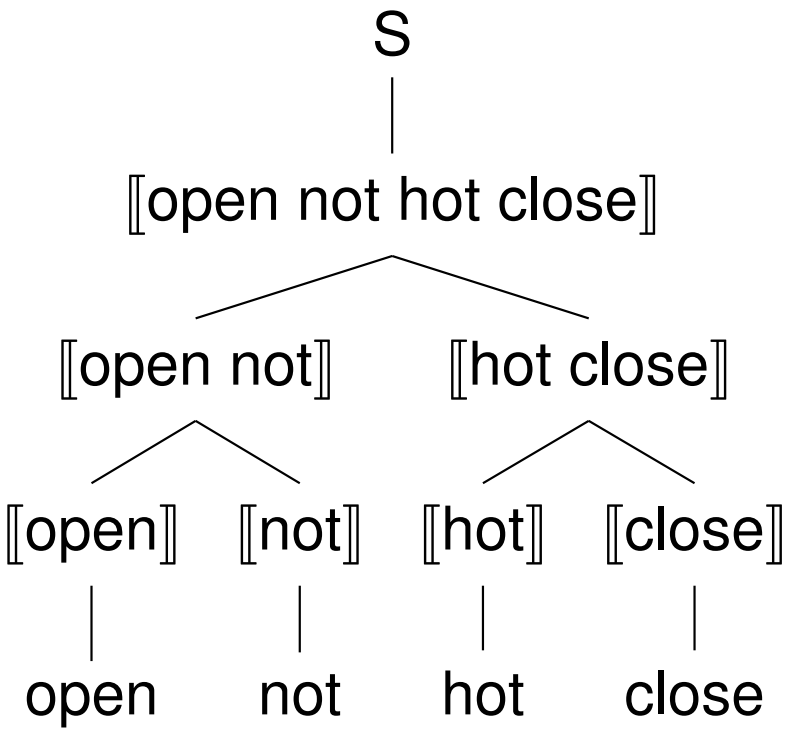
- hot
- cold
- open hot or cold close
- open not hot close
- open hot and cold close
- open hot implies cold close
- open hot iff cold close
- danger
- rain
- snow

One production for each example

- $S \rightarrow$ hot
- $S \rightarrow$ cold
- $S \rightarrow$ open hot or cold close
- $S \rightarrow$ open not hot close
- $S \rightarrow$ open hot and cold close
- $S \rightarrow$ open hot implies cold close
- $S \rightarrow$ open hot iff cold close
- $S \rightarrow$ danger
- $S \rightarrow$ rain
- $S \rightarrow$ snow

Nonterminal for each substring




S
⟦open not hot close⟧
⟦open not⟧
⟦hot close⟧
⟦open⟧
⟦not⟧
⟦hot⟧
⟦close⟧
open
not
hot
close

A trivial grammar

Input data D

$D = \{w_1, w_2, \ldots, w_n\}$ are nonempty strings.

Binarise this every way

One nonterminal $[\![w]\!]$ for every substring w.

- $[\![a]\!] \to a$
- $S \to [\![w]\!]$, $w \in D$
- $[\![w]\!] \to [\![u]\!][\![v]\!]$ when $w = u \cdot v$

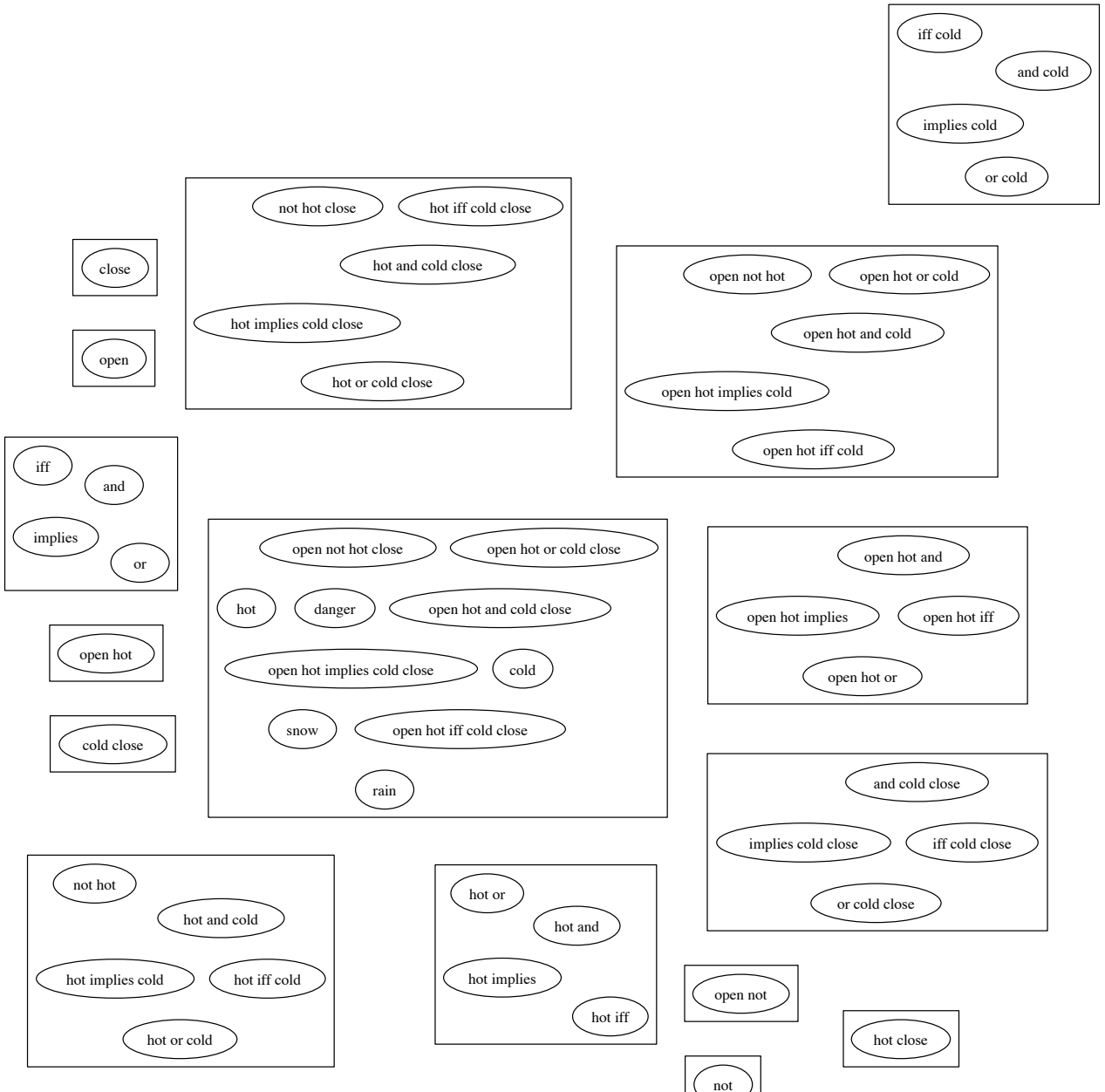
This grammar does not generalize at all! It just generates D.

Nonterminals

Nonterminals correspond to congruence classes:

- If u is a substring then we have a nonterminal $[\![u]\!]$
- $[\![u]\!]$ should generate u and all strings congruent to u: we want: $\mathcal{L}(G, [\![u]\!]) = [u]$
- So if we observe lur and lvr then we know that
  - $u \equiv v$,
  - $[u] = [v]$
  - So we want $\mathcal{L}(G, [\![u]\!]) = \mathcal{L}(G, [\![v]\!])$
  - So we either merge $[\![u]\!]$ and $[\![v]\!]$ or add productions $[\![u]\!] \to [\![v]\!]$ and $[\![v]\!] \to [\![u]\!]$.

Nonterminal for each cluster



Productions

Observation

If $w = u \cdot v$ then $[w] \supseteq [u] \cdot [v]$

Add production

$[\![w]\!] \to [\![u]\!][\![v]\!]$

Consequence

If L is substitutable, then

$$\mathcal{L}(G, [\![w]\!]) \subseteq [w]$$

$$\mathcal{L}(G) \subseteq L$$

Convergence

- As D increases the grammar will increase, and the language defined may increase (cannot decrease).
- When D is sufficiently big, the grammar will generate the whole language.

Theorem [**?**]

- If the language is a substitutable context-free language, then the hypothesis grammar will converge to a correct grammar.
- Efficient; provably correct

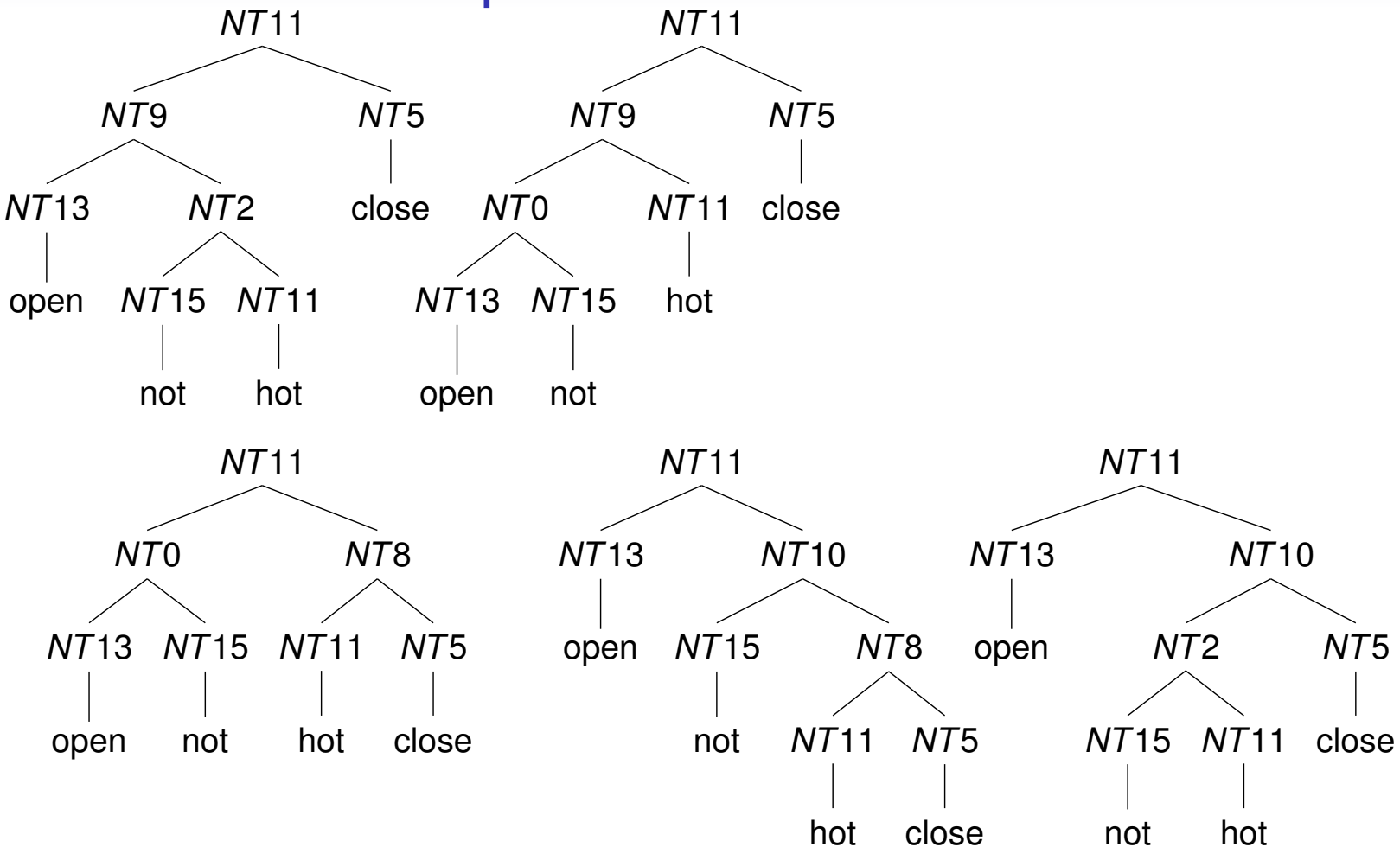
Efficiency in two senses

- Computational efficiency: the amount of computation time grows slowly as a function of the size of the input data.
- Sample efficiency: the amount of data needed to produce a correct hypothesis grows slowly as a function of the size of the target grammar.

Critique

- Natural languages aren't substitutable
- Congruence classes are inappropriate
- Natural languages aren't context-free
- Only weak learning – doesn't learn right structures

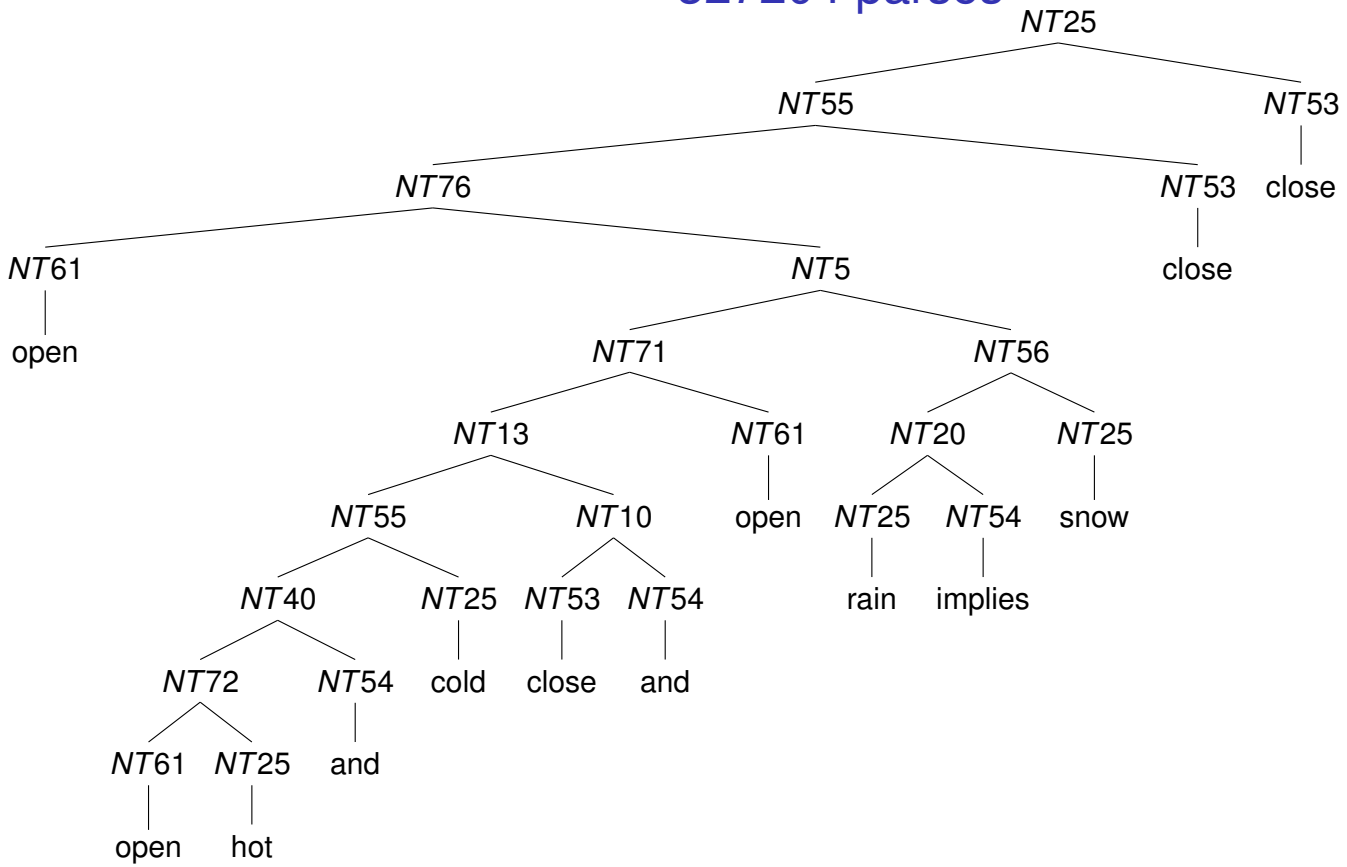
open not hot close



Larger data set: 92 nonterminals, 435 Productions

open open hot and cold close and open rain implies snow close close

327204 parses



Bibliography I

Clark, A. and Eyraud, R. (2007).
Polynomial identification in the limit of substitutable context-free languages.
Journal of Machine Learning Research, 8:1725–1745.

Harris, Z. (1964).
Distributional structure.
In Fodor, J. A. and Katz, J. J., editors, *The structure of language: Readings in the philosophy of language*, pages 33–49. Prentice-Hall.