On Distributional Learning

Alexander Clark

Department of Computer Science
Royal Holloway, University of London
alexc@cs.rhul.ac.uk

15 May 2010
Cornell Workshop on Grammar Induction

Modelling assumptions

- Syntax – non-regular representations
- Proof plus simple examples
- Algorithmic issues
- Non-probabilistic paradigm

Outline

Computational learning theory

Two problems

- Information theoretic problems
- Computational complexity problems

Learning theory should help!

- Unfortunately, it has often been distorted and used to stifle research rather than guide it.
- If X is mathematically impossible, then don't try doing X . . .
- If you have a program that seems to do X, then it is probably actually doing something else *Y*

Two problems of grammar induction
First problem

Information theoretic problems

- Absence of negative data (Gold, 1967)
- VC-dimension (Vapnik, 1998)
- Sparsity, Noise etc.

We know how to attack these problems: MDL, NPB, MaxEnt
Not specific to language

Two problems of grammar induction
Second problem

Computational problems

Complexity of finding the best hypothesis

- Gold (1978), Kearns and Valiant (1989) ...
- Specific to certain classes of representation
- Often based on embedding cryptographic problems in learning problems

Tractable Cognition Thesis (van Rooij, 2008)

Human cognitive capacities are constrained by the fact that humans are finite systems with limited resources for computation.

Overview

	Inefficient	Efficient
Positive data and MQs	Gold (1967)	?
Stochastic data	Horning (1969) Angluin (1988) Chater and Vitanyi (2007)	?

These results suggest the presence of probabilistic data largely compensate for the absence of negative data. (Angluin, 1988)

Regular inference
A success story

Paradigm	Learnable class	
Positive Data	reversible languages	Angluin (1982)
Queries	regular languages	Angluin (1987)
Positive and Negative	regular languages	Oncina and Garcia (1992)
Stochastic data	acyclic PDFAs	Ron et al (1994),
	regular languages	Carrasco and Oncina (199
	regular languages	Clark and Thollard (2004)

- Positive data only chosen adversarially is unrealistically hard.
- Add Membership Queries (where the learner can ask whether a sentence is in the language)
- We will use this model as a place holder for more realistic probabilistic models (Clark and Lappin, 2009).

Why are DFAs learnable?

Objective representations

Example: $L = (abc)^* = \{\lambda, abc, abcabc \dots\}$

Residual languages

$u^{-1}L = \{v | uv \in L\}$
$a^{-1}L = \{bc, bcabc \dots\}$
$ab^{-1}L = \{c, cabc \dots\}$
$abc^{-1}L = L$
$b^{-1}L = \emptyset$

Why are DFAs learnable?

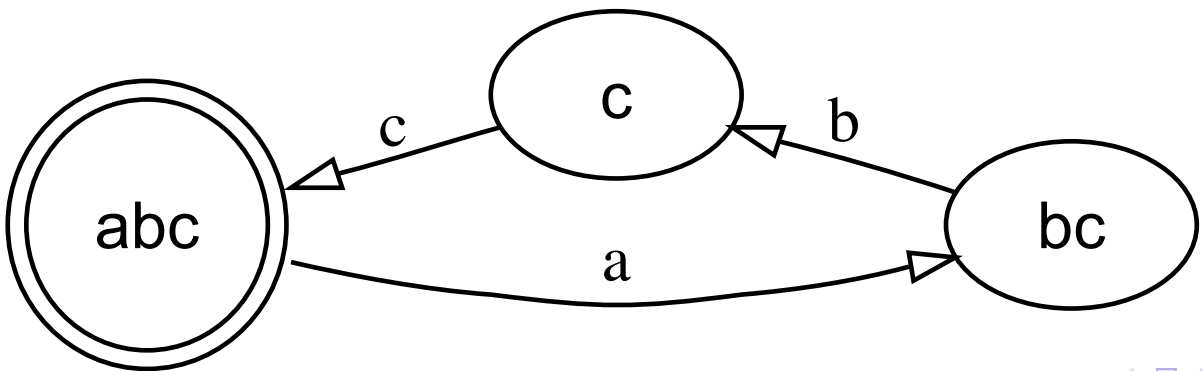
Objective representations

Example: $L = (abc)^* = \{\lambda, abc, abcabc \dots\}$

Residual languages

$u^{-1}L = \{v | uv \in L\}$
$a^{-1}L = \{bc, bcabc \dots\}$
$ab^{-1}L = \{c, cabc \dots\}$
$abc^{-1}L = L$
$b^{-1}L = \emptyset$

The minimal DFA has states which exactly correspond to residual languages.



“Empiricist” models

Slogan

The structure of the representation should be based on the structure of the language, not something arbitrarily imposed on it from outside.

- Identify some structure in the language
- Show how that structure can be observed
- Construct a representation based on that structure
- Richer structures will give you more powerful representations

Outline

Distributional learning

Zellig Harris, Olga Kulagina

Default assumption

Generalise in some way from a set of examples.

Natural algorithmic idea:

- Look at the doggy
- Look at the car
- Look at the biscuit
- Look at the blue car
- the doggy is over there
- the biscuit is over there
- ...

Question: what classes of languages can be learned using this approach?

Problems

A classic example from Chomsky:

- John is easy to please
- John is eager to please

Problems

A classic example from Chomsky:

- John is easy to please
- John is eager to please
- They are ready to eat

Problems

A classic example from Chomsky:

- John is easy to please
- John is eager to please
- They are ready to eat

Displaced constituents

- This is the book that John said that Mary had ...

Distributional learning

Several reasons to take distributional learning seriously:

- Cognitively plausible (Saffran et al. 1996)
- 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.”

Distribution

Example

That man over there is bothering me

Distribution

Example

That man over there is bothering me

Split

- A substring “man over”
- A context “That _ there is bothering me”

This is “observable”

Distribution

Classic idea from structuralist linguistics:

Context (or *environment*)

A context is just a pair of strings $(l, r) \in \Sigma^* \times \Sigma^*$.
$(l, r) \odot u = lur$
$f = (l, r)$.
Special context (λ, λ)

Given a language $L \subseteq \Sigma^*$.

Distribution of a string

$C_L(u) = \{(l, r) | lur \in L\} = \{f | f \odot u \in L\}$

"Distributional Learning" models/exploits the distribution of strings;

Outline

Classic distributional learning

Congruence classes

$u \equiv v$ iff $C_L(u) = C_L(v)$
Write $[u]$ for class of u

Chomsky review of Greenberg (1959)

Let us say that two units A and B are substitutable$_1$ if there are expressions X and Y such that XAY and XBY are sentences of L; substitutable$_2$ if whenever XAY is a sentence of L then so is XBY and whenever XBY is a sentence of L so is XAY (i.e. A and B are completely mutually substitutable). These are the simplest and basic notions. (footnote 3. They are discussed by R. Carnap in The Logical Syntax of Language, 1934)

Example

$L = \{a^n b^n | n \geq 0\}$

$C_L(a) = \{(\lambda, b), (a, bb), (a, abbb) \dots\}$
$C_L(aab) = \{(\lambda, b), (a, bb), \dots\}$
$C_L(aaabb) = \{(\lambda, b), (a, bb), \dots\}$

English

$C_L(cat) = \{$ look at the _, the _ is on the mat $\dots\}$
$C_L(dog) = \{$ look at the _, the _ is on the mat $\dots\}$

Observation table

K a set of strings and *F* a set of contexts

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										
k_7										
k_6										
k_5										
k_4										
k_3										
k_2										
k_1										

Observation table

K a set of strings and *F* a set of contexts

	(λ, λ)	(a, λ)	(λ, b)
λ	■		
a			■
b		■	
ab	■		

Observation table

K a set of strings and *F* a set of contexts

	$(aaabb, bccc)$	$(\lambda, abbccc)$	(aa, bbc)	(abb, cc)
	(λ, λ)	$(aaabbc, \lambda)$	$(aaab, bccc)$	$(aa, bbbc)$	$(abbb, cc)$
bcc									■
aab							■		
$bbcc$	■							■	
bc	■							■	
abc	■								
$aabb$	■					■			
ab	■					■			
c	■		■						■
b					■				
a	■			■			■		
λ	■	■				■		■	

Observation table

K a set of strings and *F* a set of contexts

	(λ, λ)	$(aaabb, bccc)$	$(\lambda, abbccc)$	$(aa, bbbc)$	(aa, bbc)	(abb, cc)	$(aaabbc, \lambda)$	$(aaab, bccc)$	$(abbb, cc)$
bcc									■
aab								■	
abc		■							
$aabb$	■	■							
ab	■	■							
λ	■	■	■	■					
$bbcc$		■	■						
bc		■	■						
c		■			■				■
b							■		
a		■				■		■	

Relation to CFGs

Define

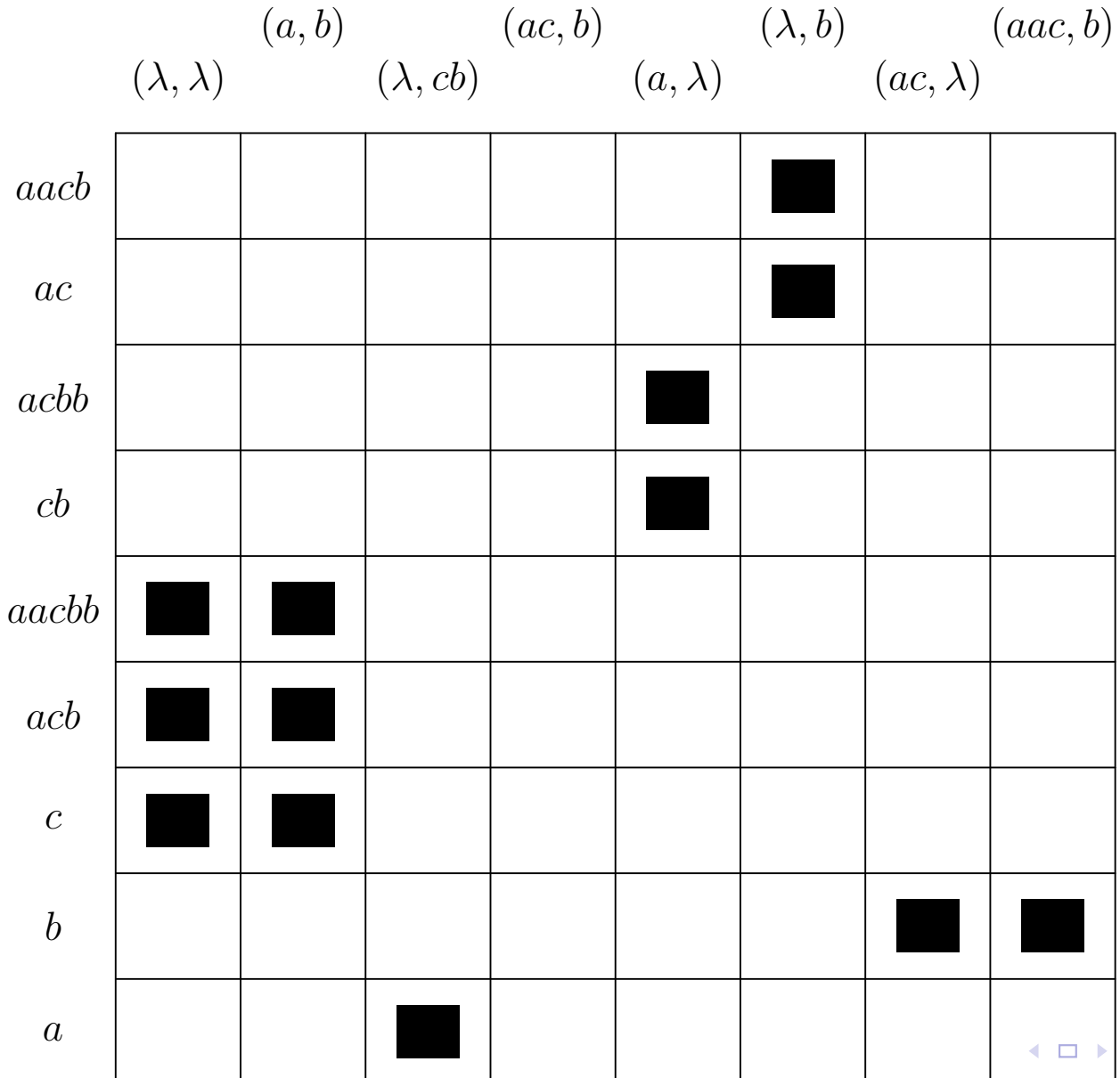
Given a CFG G for each non-terminal N

- Yield: $Y(N) = \{w | N \stackrel{*}{\Rightarrow} w\}$
- Contexts: $C(N) = \{(l, r) | S \stackrel{*}{\Rightarrow} lNr\}$.

Clearly $C(N) \odot Y(N) \subseteq L$
Each non-terminal will be a rectangle – but not necessarily maximal.

Substitutable

$L = \{a^n cb^n | n \geq 0\}$



Old concept

John Myhill, 1950 commenting on Bar-Hillel

I shall call a system *regular* if the following holds for all expressions μ, ν and all wffs ϕ, ψ each of which contains an occurrence of ν: If the result of writing μ for some occurrence of ν in ϕ is a wff, so is the result of writing μ for any occurrence of ν in ψ. Nearly all formal systems so far constructed are regular; ordinary word-languages are conspicuously not so.

Clark and Eyraud, 2005

A language is *substitutable* if $lur, lvr, l'ur' \in L$ means that $l'vr' \in L$.
Substitutable CF languages are polynomially learnable.

Why the delay?

Context free grammar

Suppose we have a grammar with non-terminals N, P, Q

- We have a rule $N \to PQ$
- This means that $Y(N) \supseteq Y(P)Y(Q)$.

Context free grammar

Suppose we have a grammar with non-terminals N, P, Q

- We have a rule $N \to PQ$
- This means that $Y(N) \supseteq Y(P)Y(Q)$.

Backwards

Objectively define a collection of sets of strings X, Y, Z
Suppose $X \supseteq YZ$
Then we add a rule $X \to YZ$.

Congruence classes
Partition of the strings

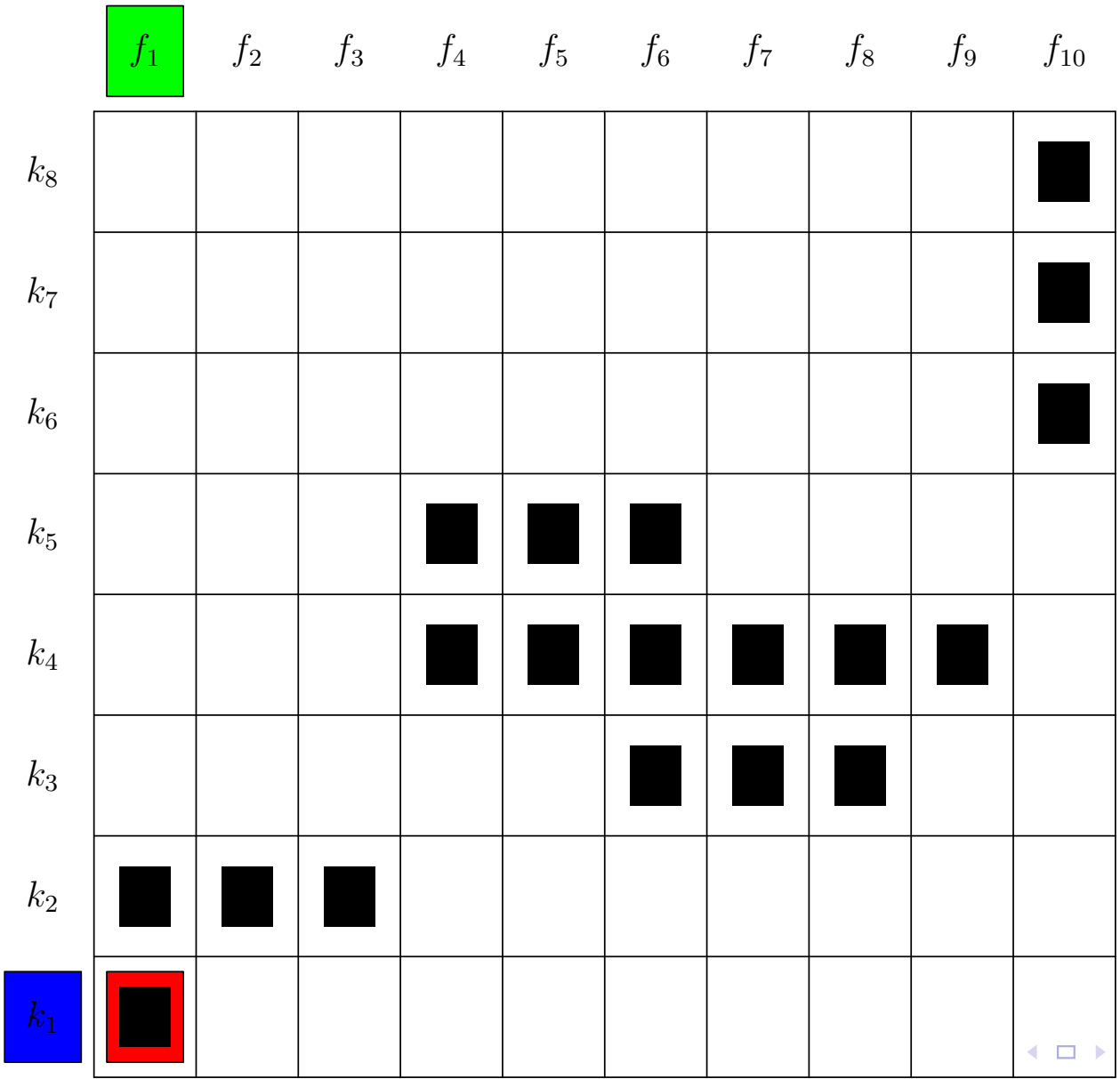
Congruence classes have nice properties!

$[u][v] \subseteq [uv]$
$[uv] \rightarrow [u][v]$

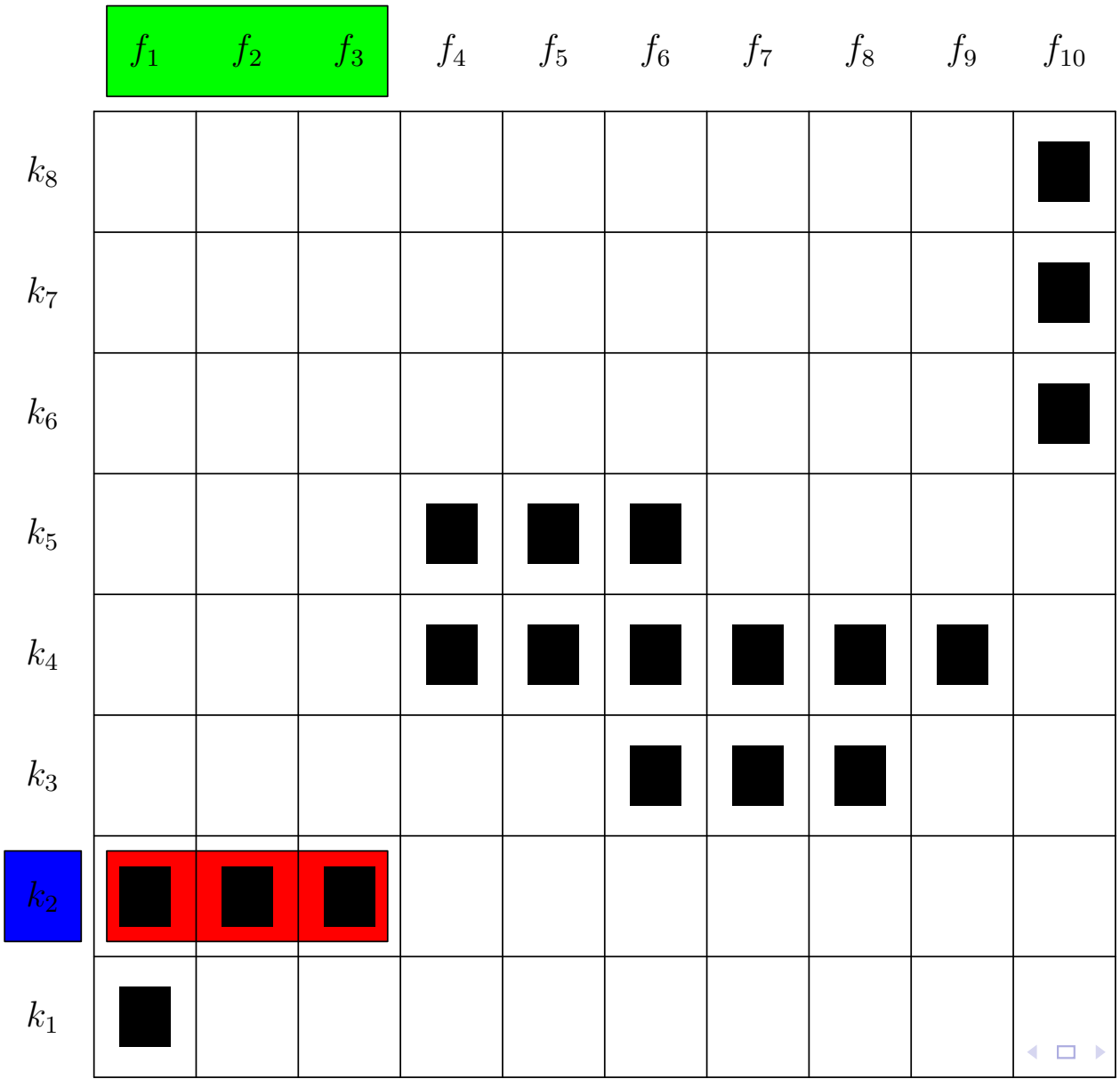
$L = \{a^n b^n | n \geq 0\}$

$[a] = \{a\}$
$[abb] = \{abb, aabbb, \dots\}$
$[a][aab] = \{aabb, aaabbb \dots\} \subseteq [aabb] = [ab]$
So we have a rule $[ab] \rightarrow [a][aab]$

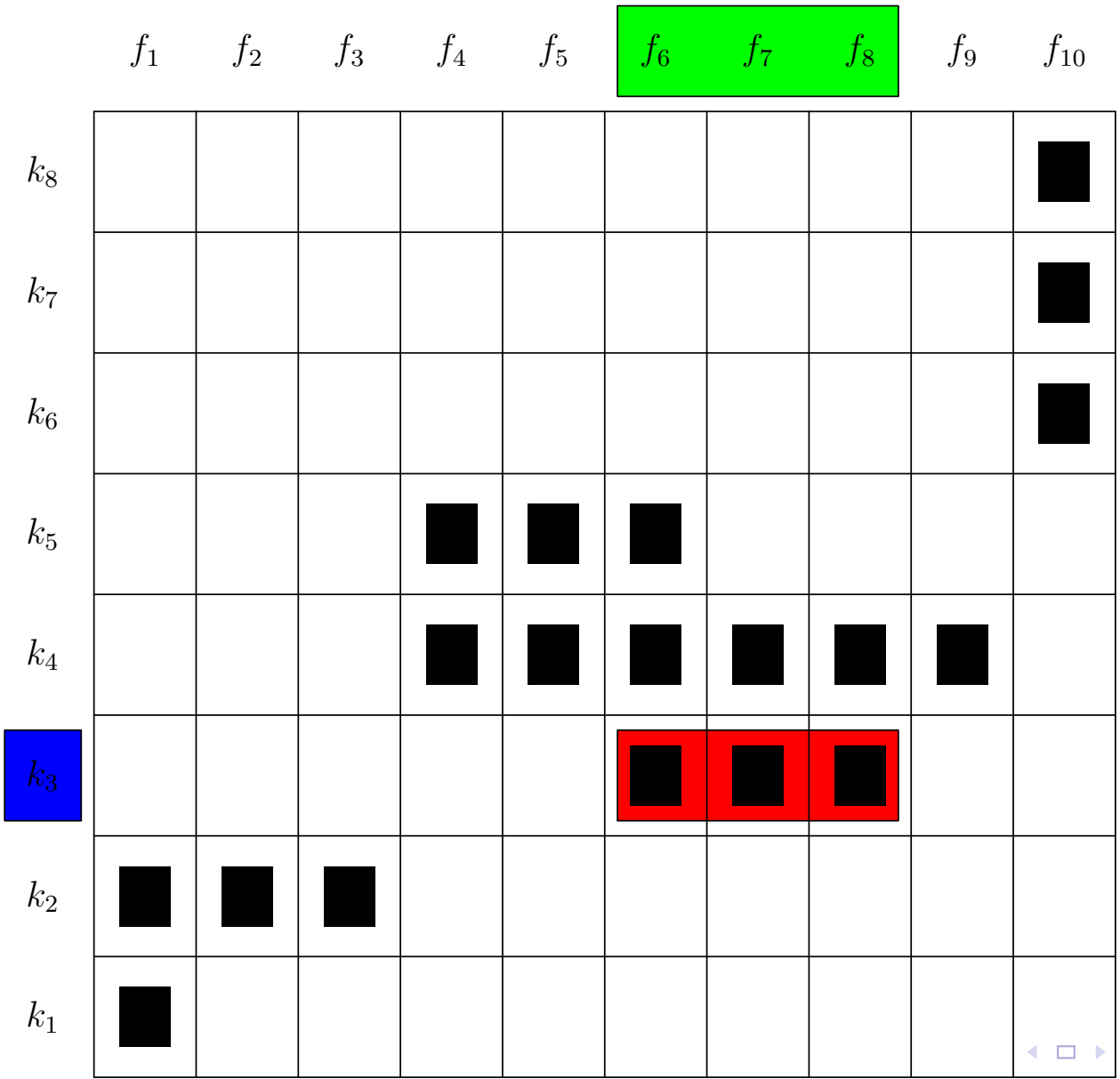
Example
Artificial



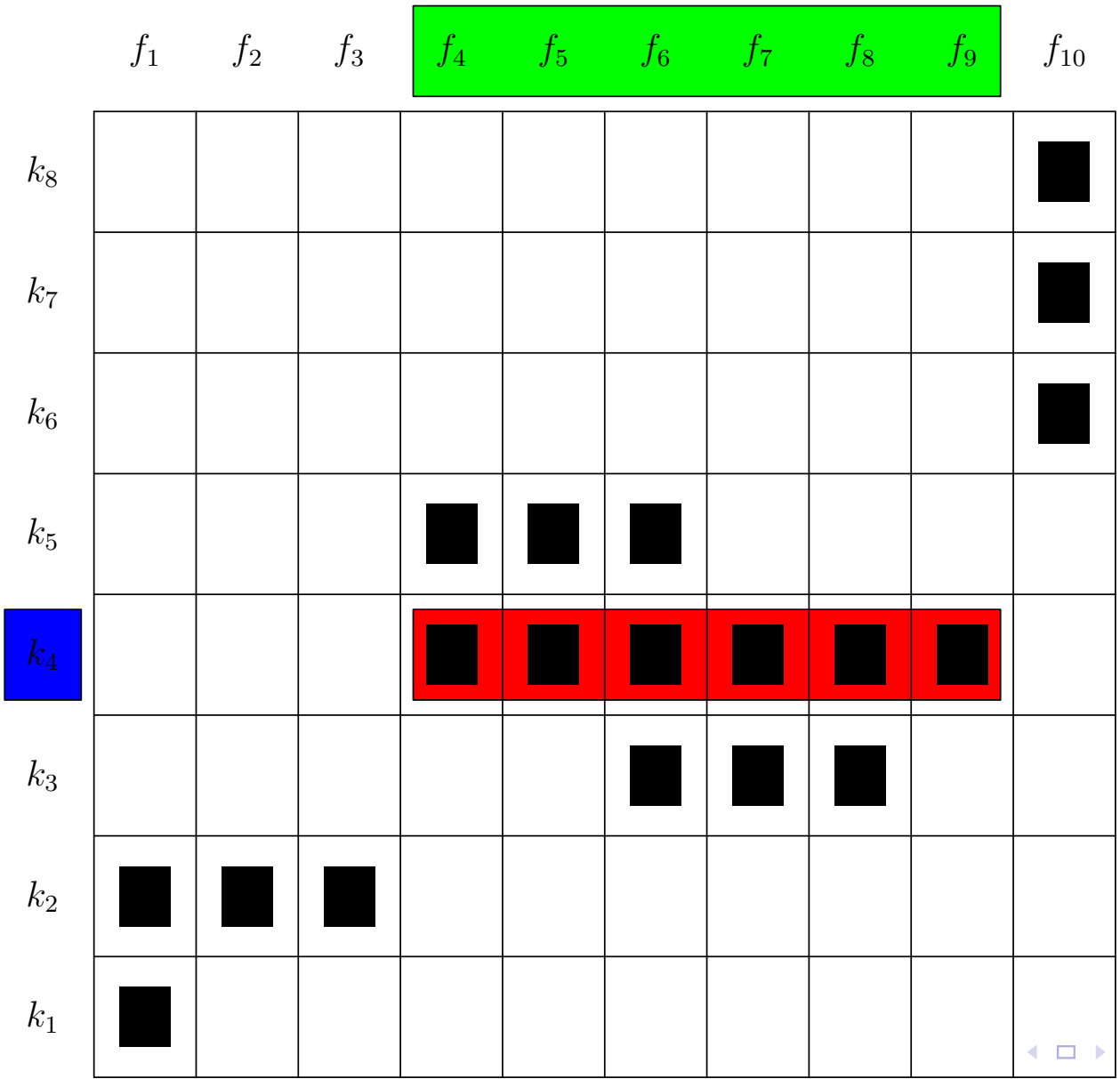
Example
Artificial



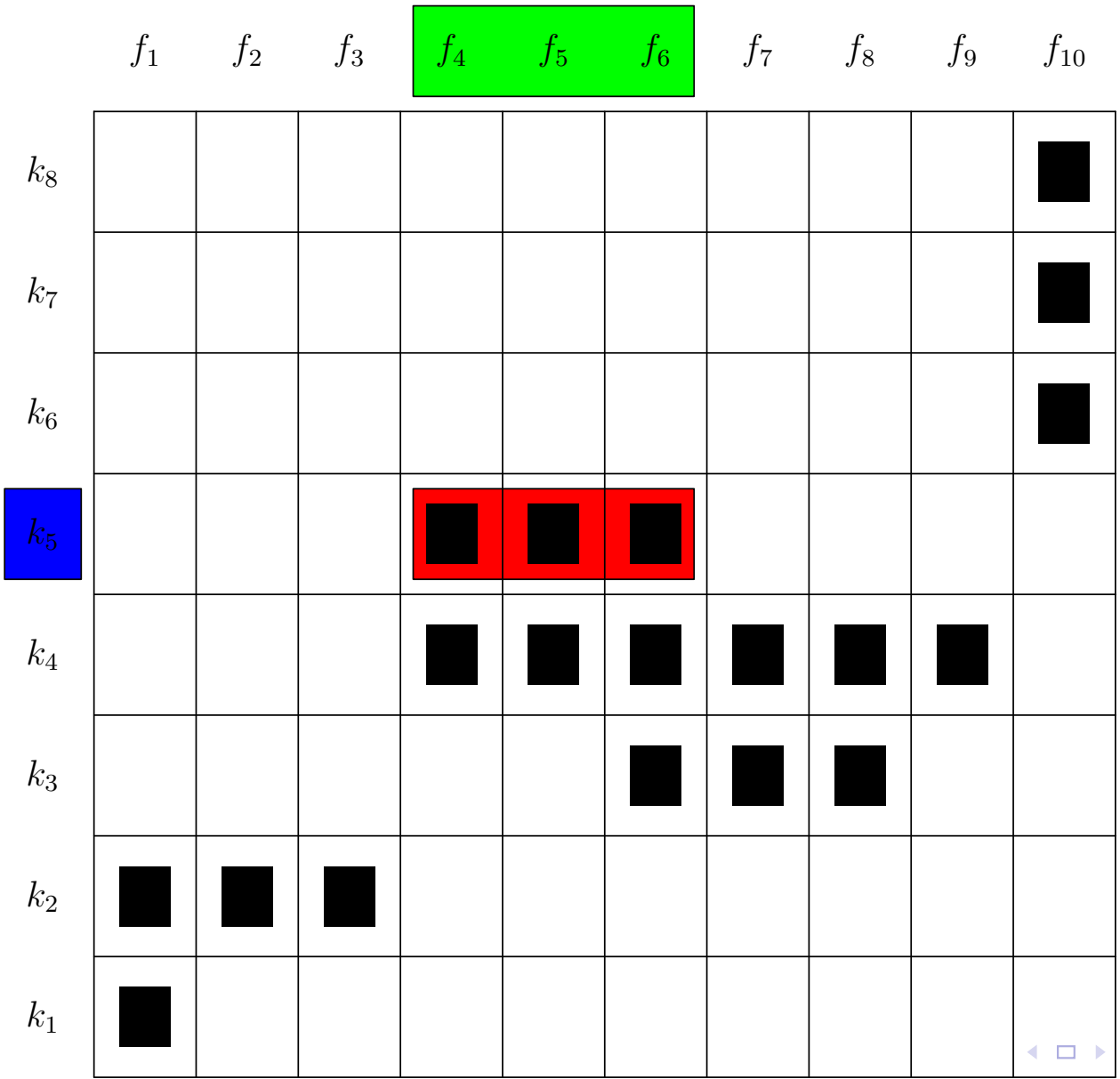
Example
Artificial



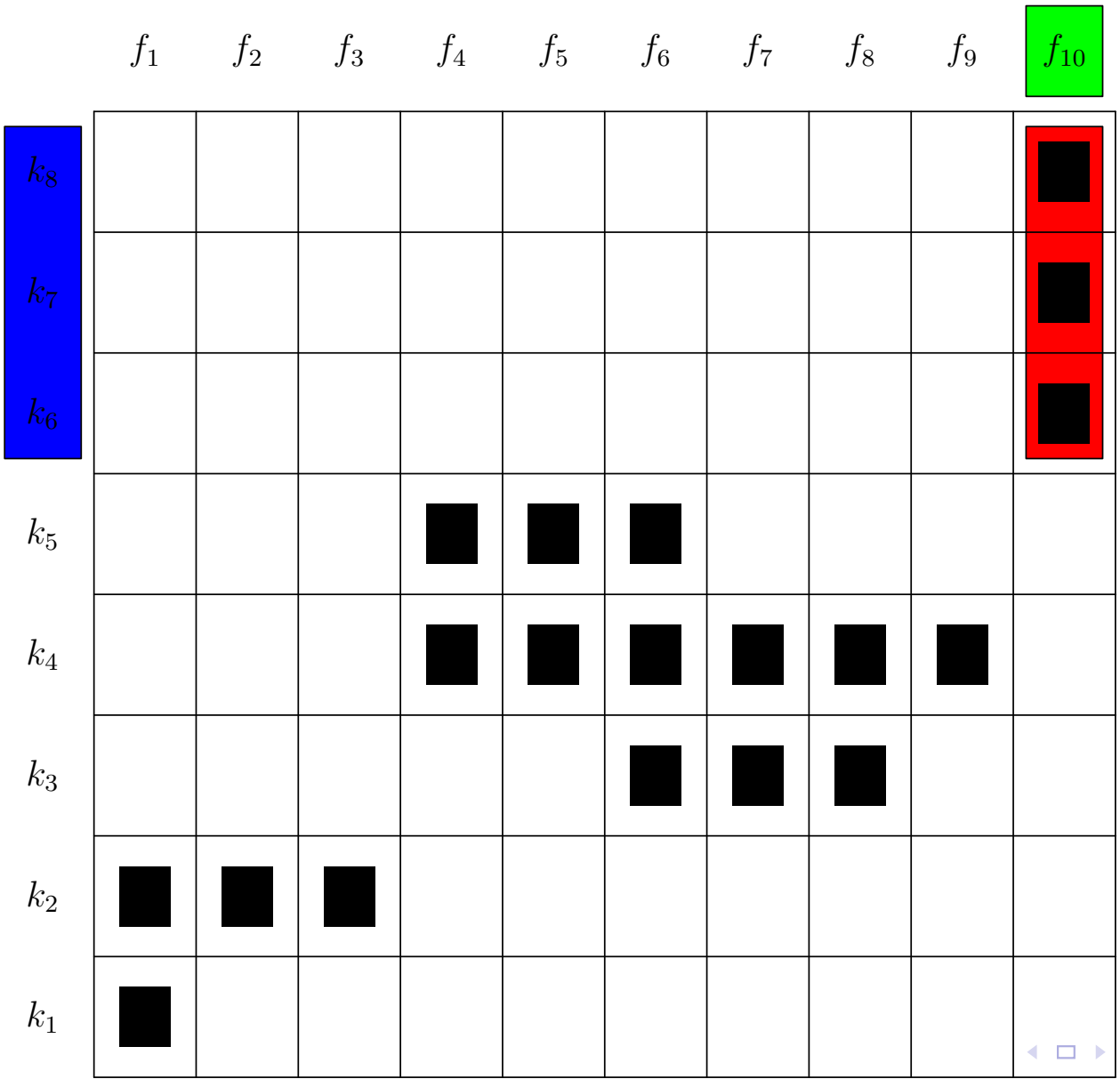
Example
Artificial



Example
Artificial



Example
Artificial



Example

$L = \{a^n b^n | n \geq 0\}$

Grammar

- $S \rightarrow [ab], S \rightarrow [\lambda]$

Example

$L = \{a^n b^n | n \geq 0\}$

Grammar

- $S \rightarrow [ab], S \rightarrow [\lambda]$
- $[a] \rightarrow a, [b] \rightarrow b, [\lambda] \rightarrow \lambda$

Example

$L = \{a^n b^n | n \geq 0\}$

Grammar

- $S \rightarrow [ab], S \rightarrow [\lambda]$
- $[a] \rightarrow a, [b] \rightarrow b, [\lambda] \rightarrow \lambda$
- $[ab] \rightarrow [aab][b], [ab] \rightarrow [a][b], [ab] \rightarrow [a][abb]$
- $[aab] \rightarrow [a][ab]$, $[abb] \rightarrow [ab][b]$

Example

$L = \{a^n b^n | n \geq 0\}$

Grammar

- $S \rightarrow [ab], S \rightarrow [\lambda]$
- $[a] \rightarrow a, [b] \rightarrow b, [\lambda] \rightarrow \lambda$
- $[ab] \rightarrow [aab][b], [ab] \rightarrow [a][b], [ab] \rightarrow [a][abb]$
- $[aab] \rightarrow [a][ab]$, $[abb] \rightarrow [ab][b]$
- Plus $[ba] \rightarrow [b][ba] \ldots$
- Plus $[a] \rightarrow [\lambda][a] \ldots$

Problem

Hard to tell whether $u \equiv_L v$

If we can figure out the congruence classes, we can just write down a grammar.

Congruence class results

Positive data alone

$lur \in L$ and $lvr \in L$ implies $u \equiv_L v$
Polynomial result from positive data. (Clark and Eyraud, 2005)
k-l substitutable languages, (Yoshinaka 2008)

Stochastic data

If data is generated from a PCFG
PAC-learn unambiguous NTS languages, Clark (2006)

Membership queries

An efficient query-learning result Clark (under submission)
Pick a finite set of contexts F
Test if $C_L(u) \cap F = C_L(v) \cap F$

Limitations

One symbol per congruence class is too limited

- Class of CFGs is quite small – deterministic, unambiguous
- Just won't work for natural languages.
- Congruence classes are very many and very close together
- Exact substitutability is rare – e.g. cat/dog

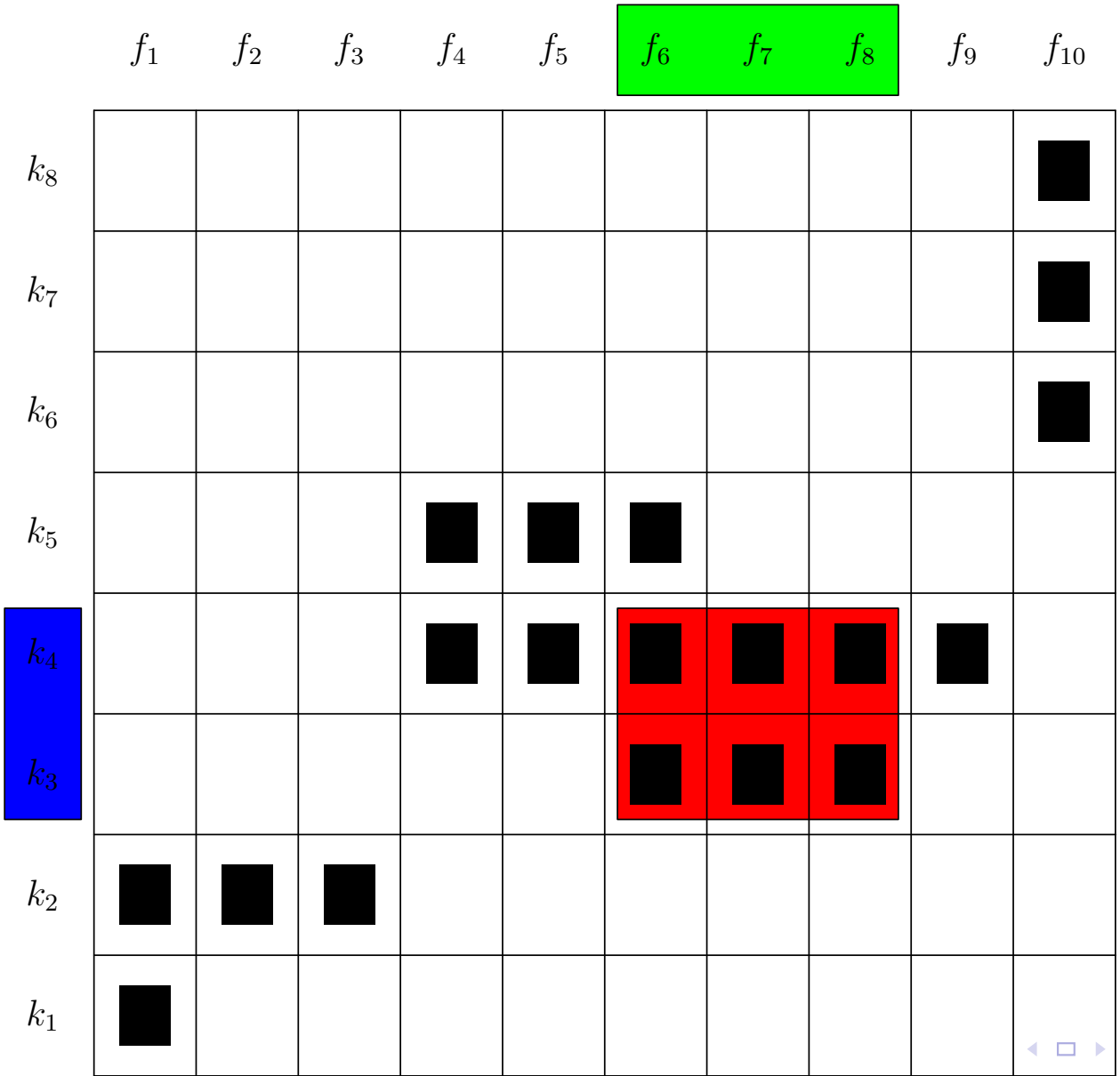
Limitations

One symbol per congruence class is too limited

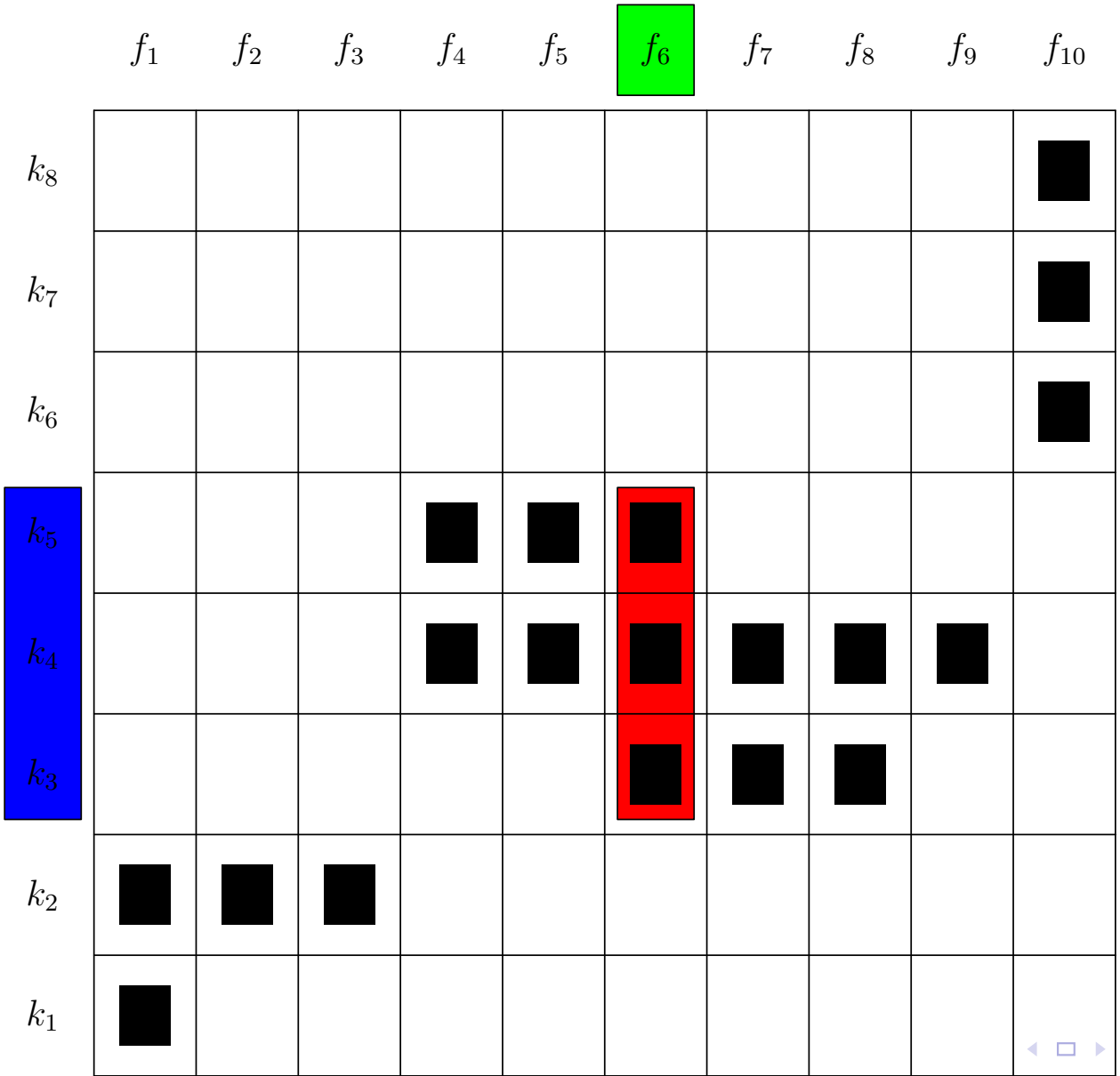
- Class of CFGs is quite small – deterministic, unambiguous
- Just won't work for natural languages.
- Congruence classes are very many and very close together
- Exact substitutability is rare – e.g. cat/dog
- Learning model assumes that either they are identical or they are completely unrelated.
- Need to have a more powerful representation that represents the structure of the congruence classes
- Languages aren't context free

Outline

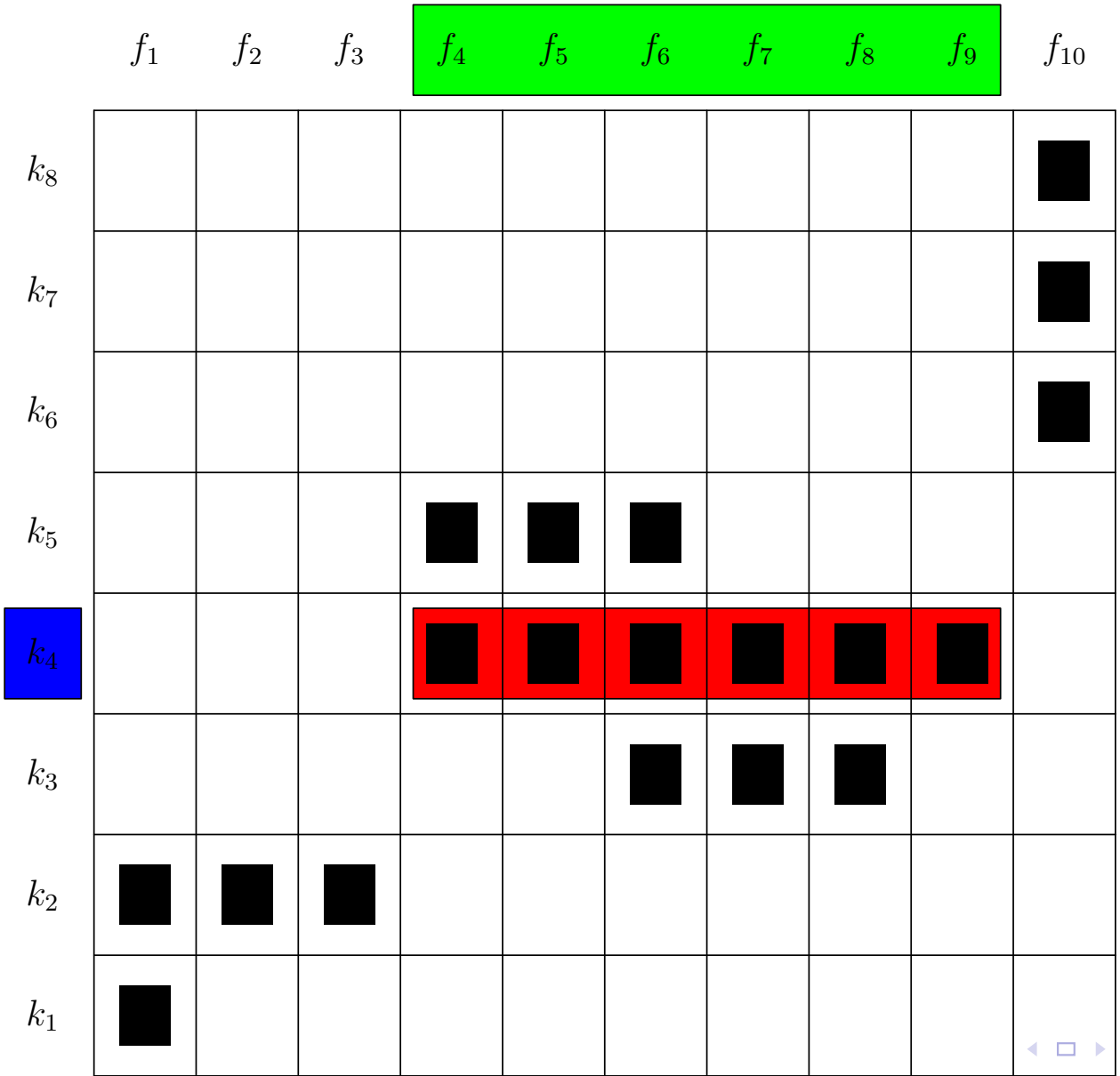
Concepts
Maximal rectangles



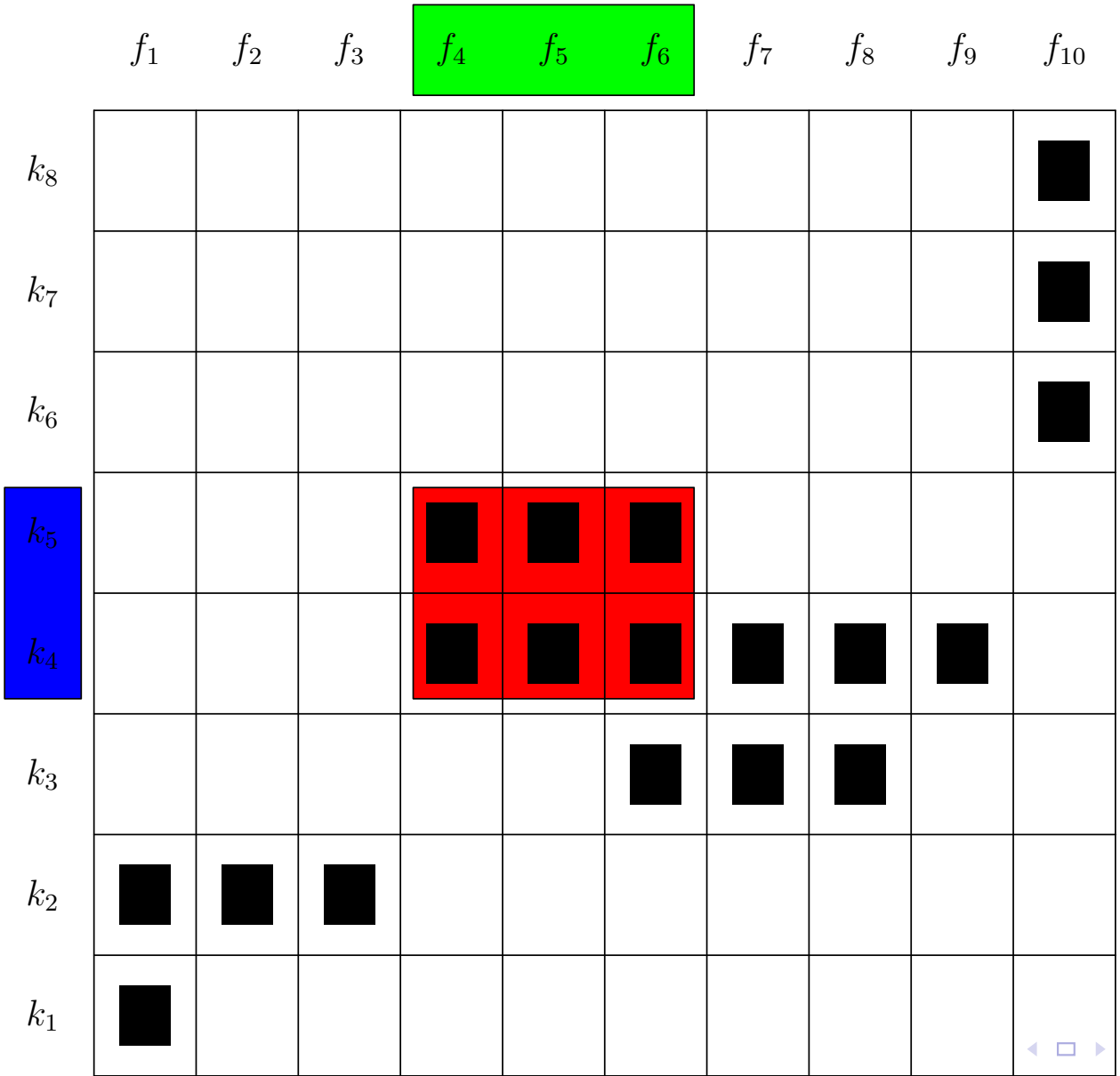
Concepts
Maximal rectangles



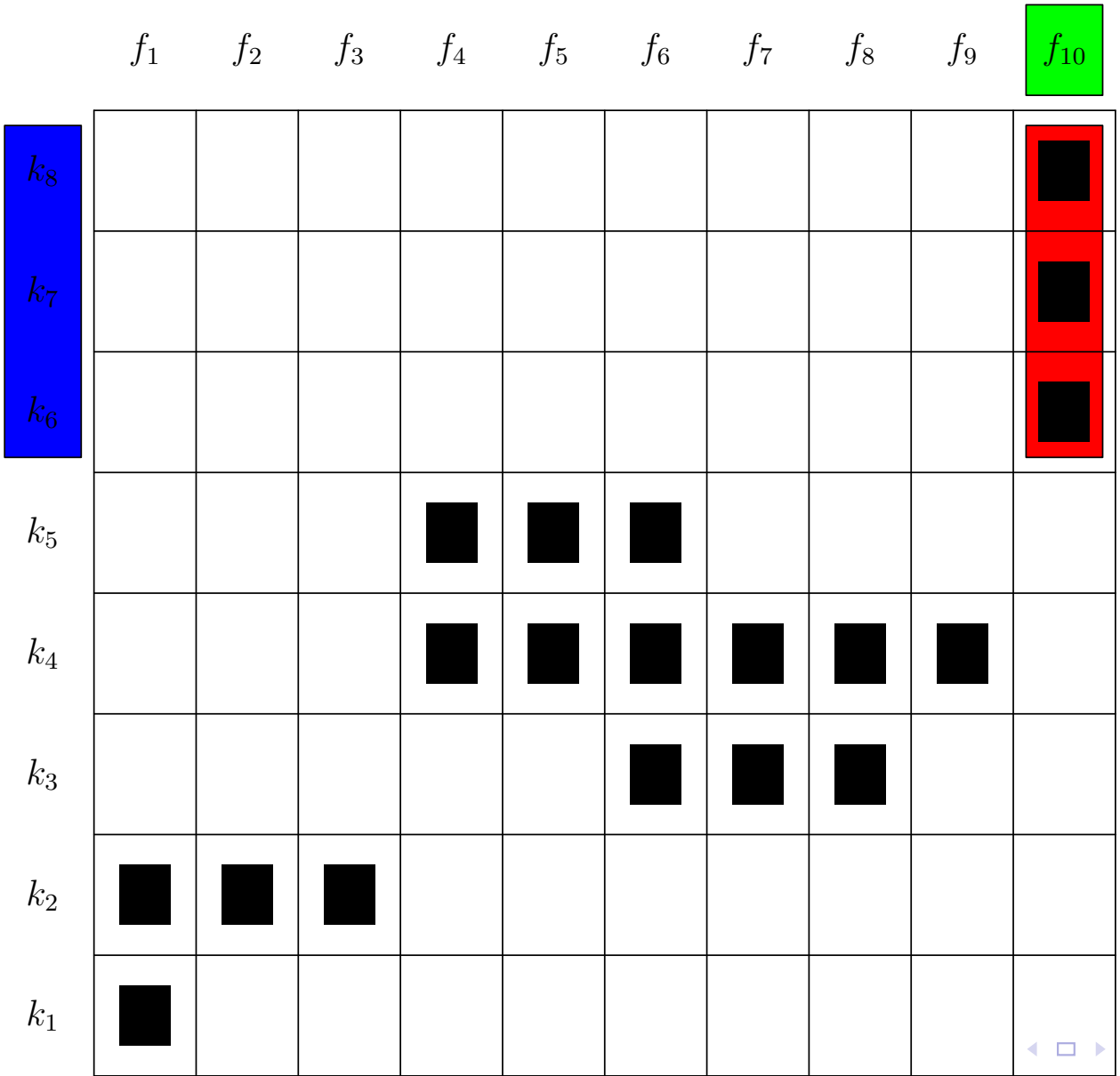
Concepts
Maximal rectangles



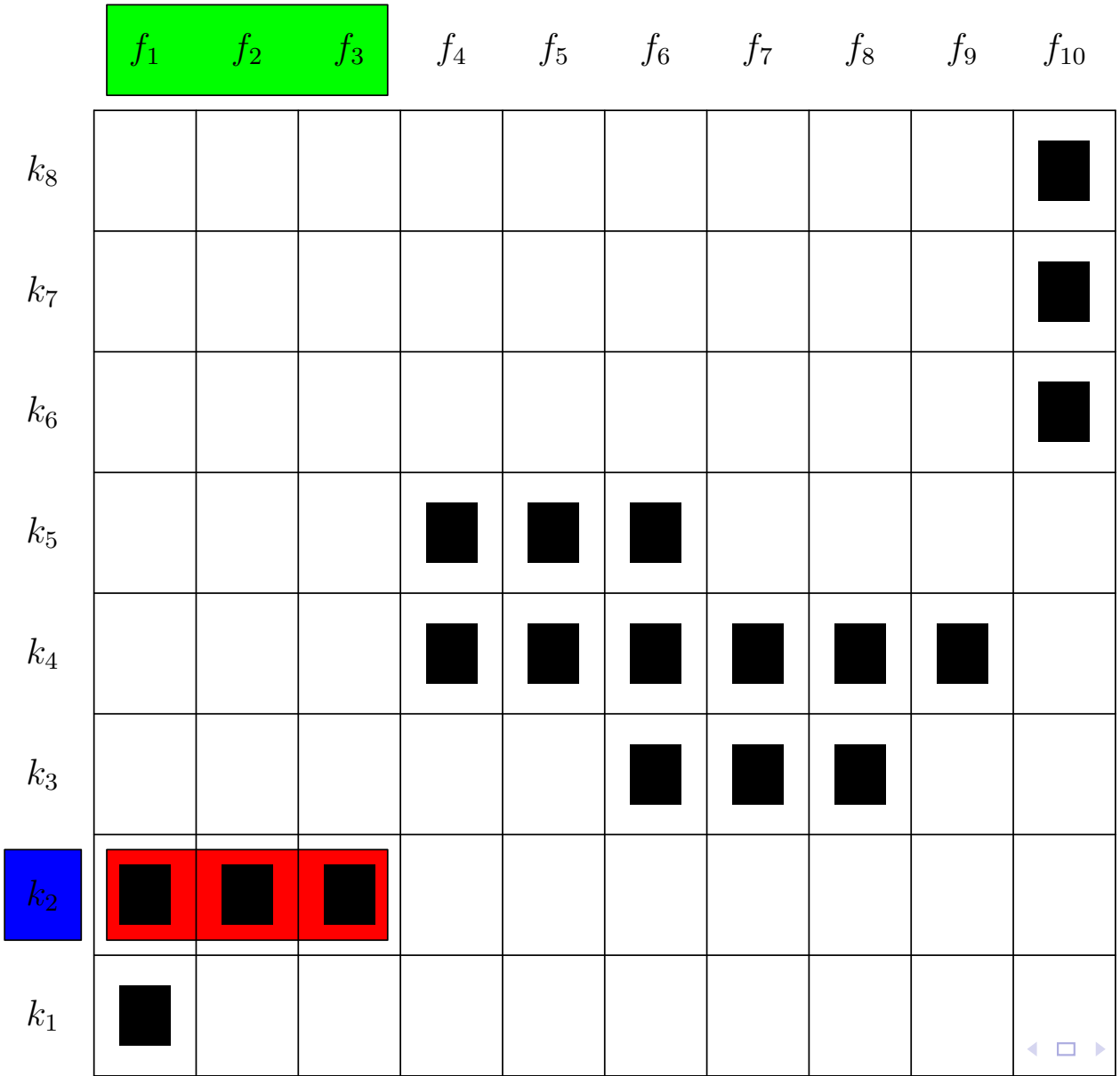
Concepts
Maximal rectangles



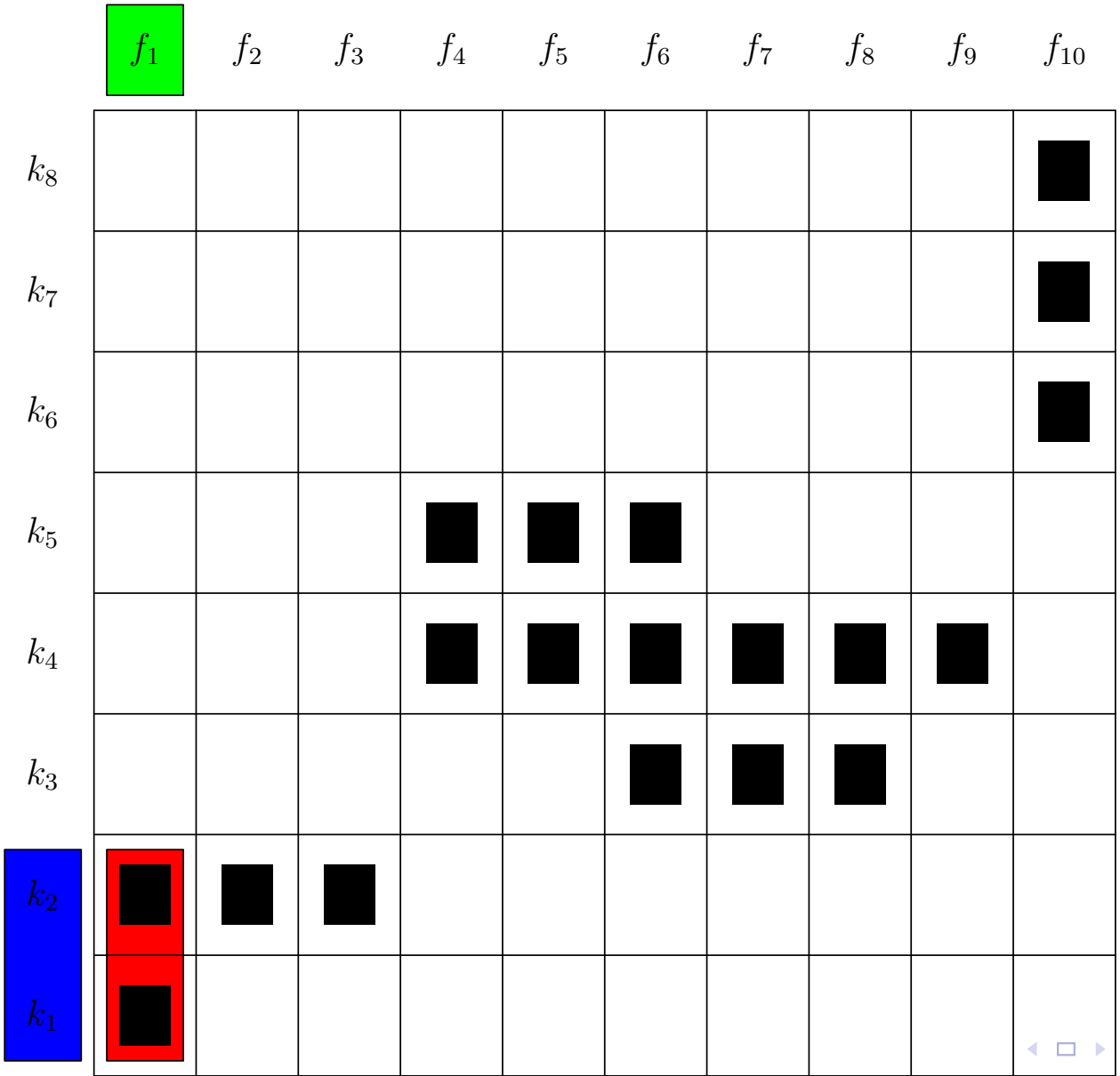
Concepts
Maximal rectangles



Concepts
Maximal rectangles

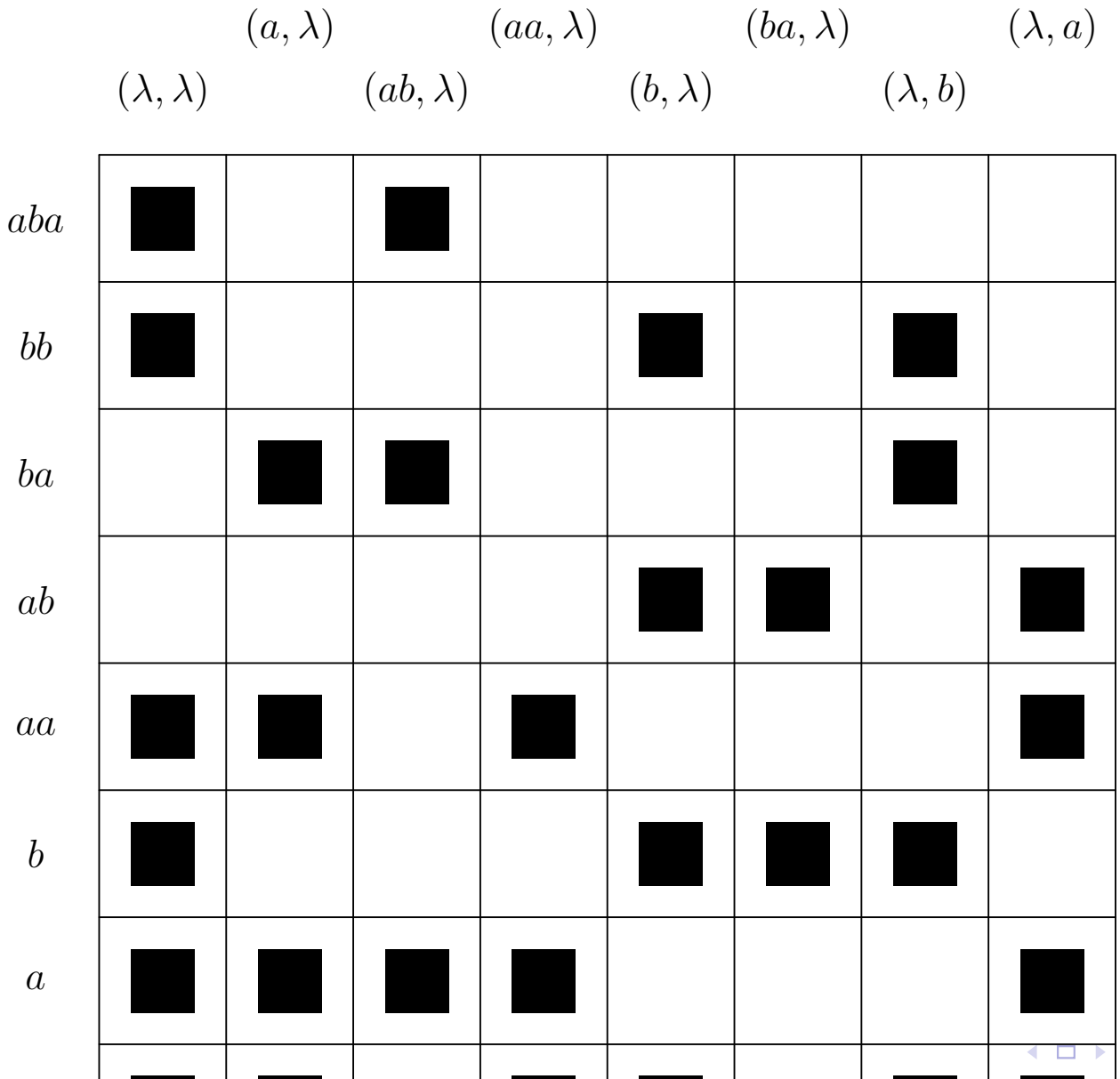


Concepts
Maximal rectangles

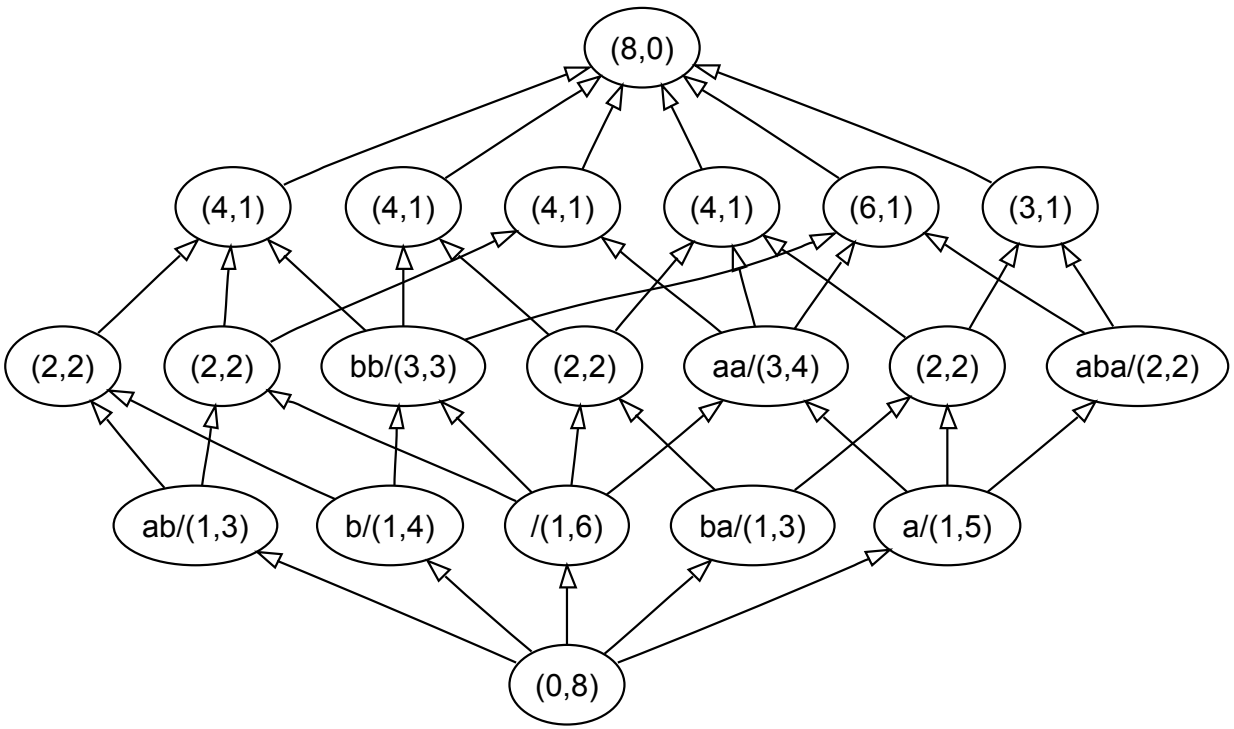


Lattice

Palindrome language over *a*, *b*



Lattice



Technical detail

- These rectangles are “concepts” which form a complete lattice $\mathfrak{B}(K, L, F)$
- We can define a concatenation operation $X \circ Y$ and a lower bound $X \wedge Y$.
- We can recursively define a function from strings to concepts

Distributional lattice grammars

Definition

$\phi : \Sigma^* \to \mathfrak{B}(K, D, F)$.

- for all $a \in \Sigma$
  $\phi(a) = \mathcal{C}(a)$
- for all w with $|w| > 1$,

$$\phi(w) = \bigwedge_{u,v \in \Sigma^+ : uv = w} \phi(u) \circ \phi(v)$$

A string w is in the language if $\phi(w)$ has the context (λ, λ).

Derivation example

Dyck language

$\{\lambda, ab, aabb, abab, aaababbb \dots\}$
$F = \{(\lambda, \lambda), (\lambda, b), (a, \lambda)\}$
$K = \{\lambda, a, b, ab\}$

- $\top = \langle K, \emptyset \rangle$
- $\bot = \langle \emptyset, F \rangle$
- $\mathbf{L} = \langle \{\lambda, ab\}, \{(\lambda, \lambda)\} \rangle$
- $A = \langle \{a\}, \{(\lambda, b)\} \rangle$
- $B = \langle \{b\}, \{(a, \lambda)\} \rangle$

Derivation example

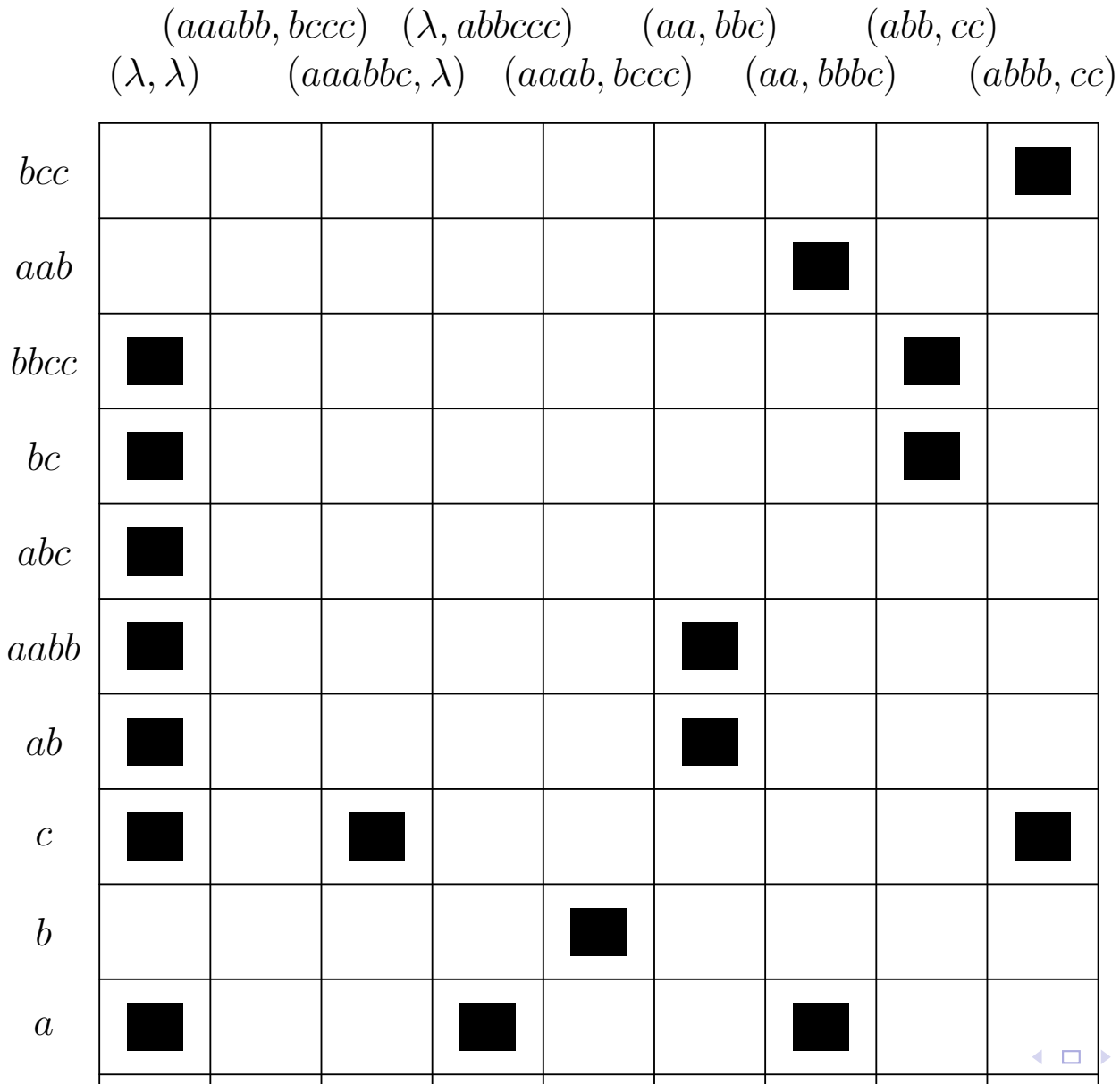
Dyck language

$\{\lambda, ab, aabb, abab, aaababbb \dots\}$
$F = \{(\lambda, \lambda), (\lambda, b), (a, \lambda)\}$
$K = \{\lambda, a, b, ab\}$

1. $\phi(a) = A$
2. $\phi(ab) = \phi(a) \circ \phi(b) = \mathbf{L}$, $\phi(aa) = \top \dots$
3. $\phi(aab) = \phi(a) \circ \phi(ab) \wedge \phi(aa) \circ \phi(b) = A$
4. $\dots$
5. $\phi(aaababbb) = \phi(a) \circ \phi(aababbb) \wedge \dots = \mathbf{L}$

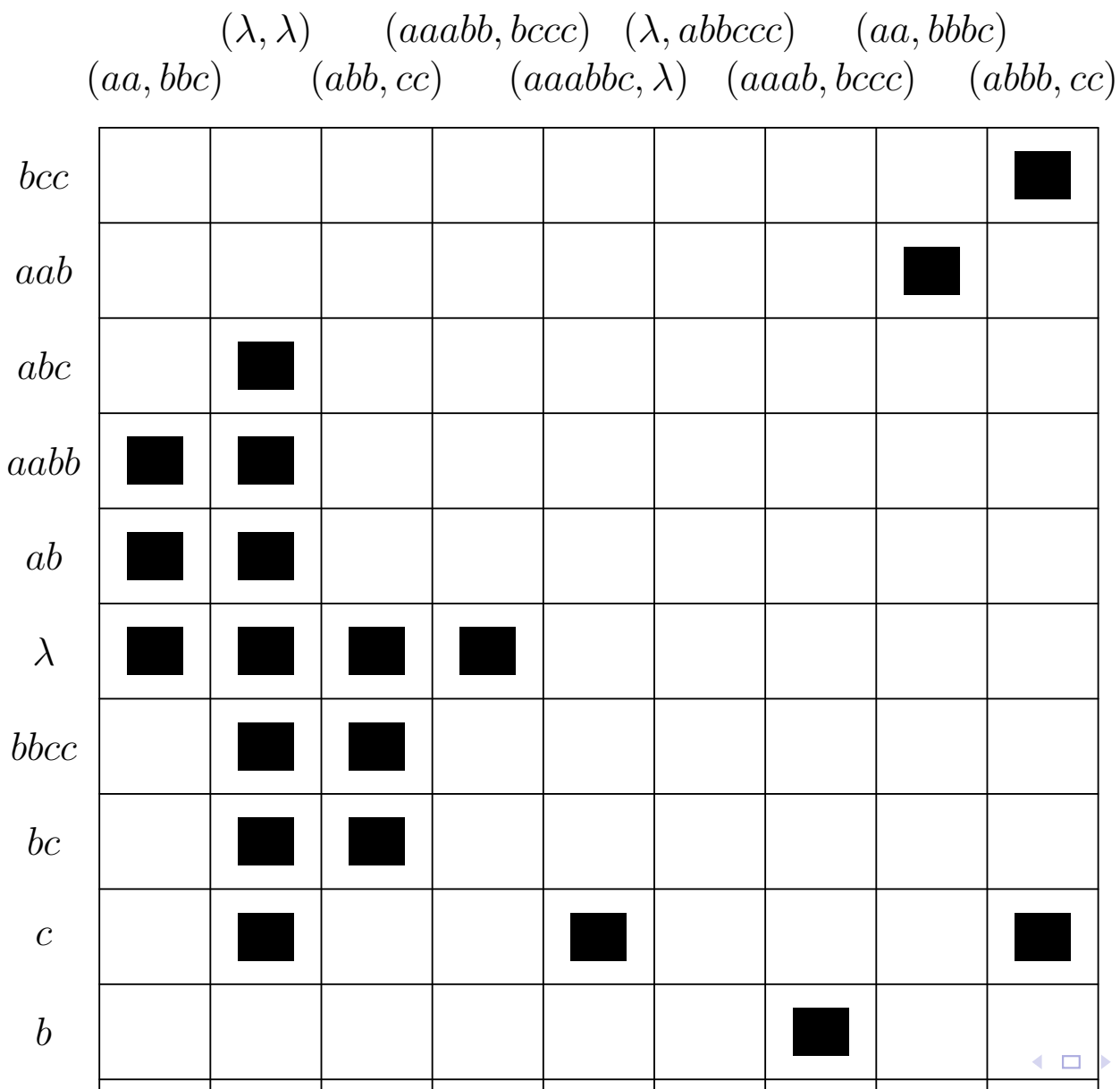
Example

$L = \{a^n b^n c^m | n, m \geq 0\} \cup \{a^m b^n c^n | n, m \geq 0\}$



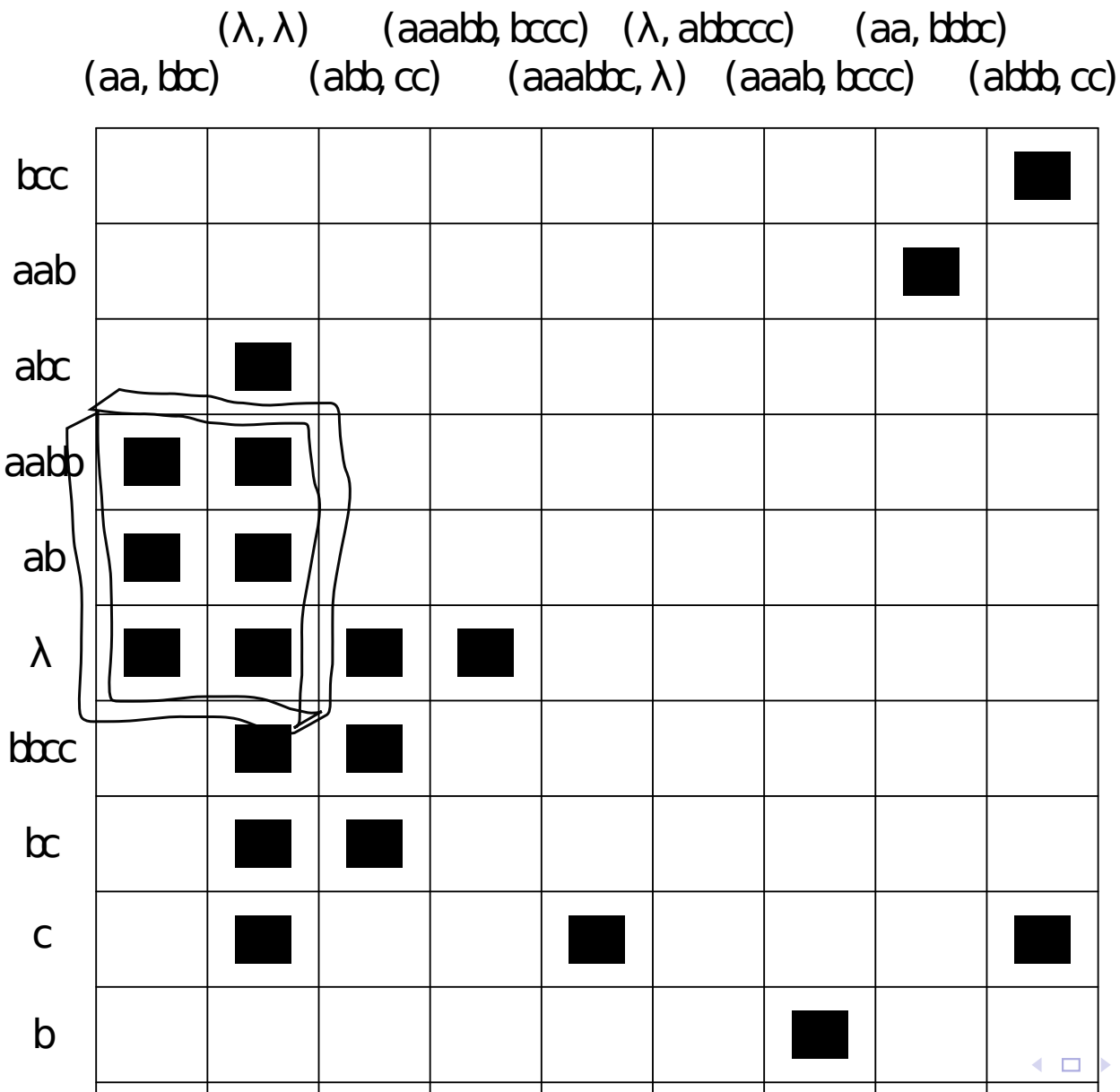
Example

$L = \{a^n b^n c^m | n, m \geq 0\} \cup \{a^m b^n c^n | n, m \geq 0\}$



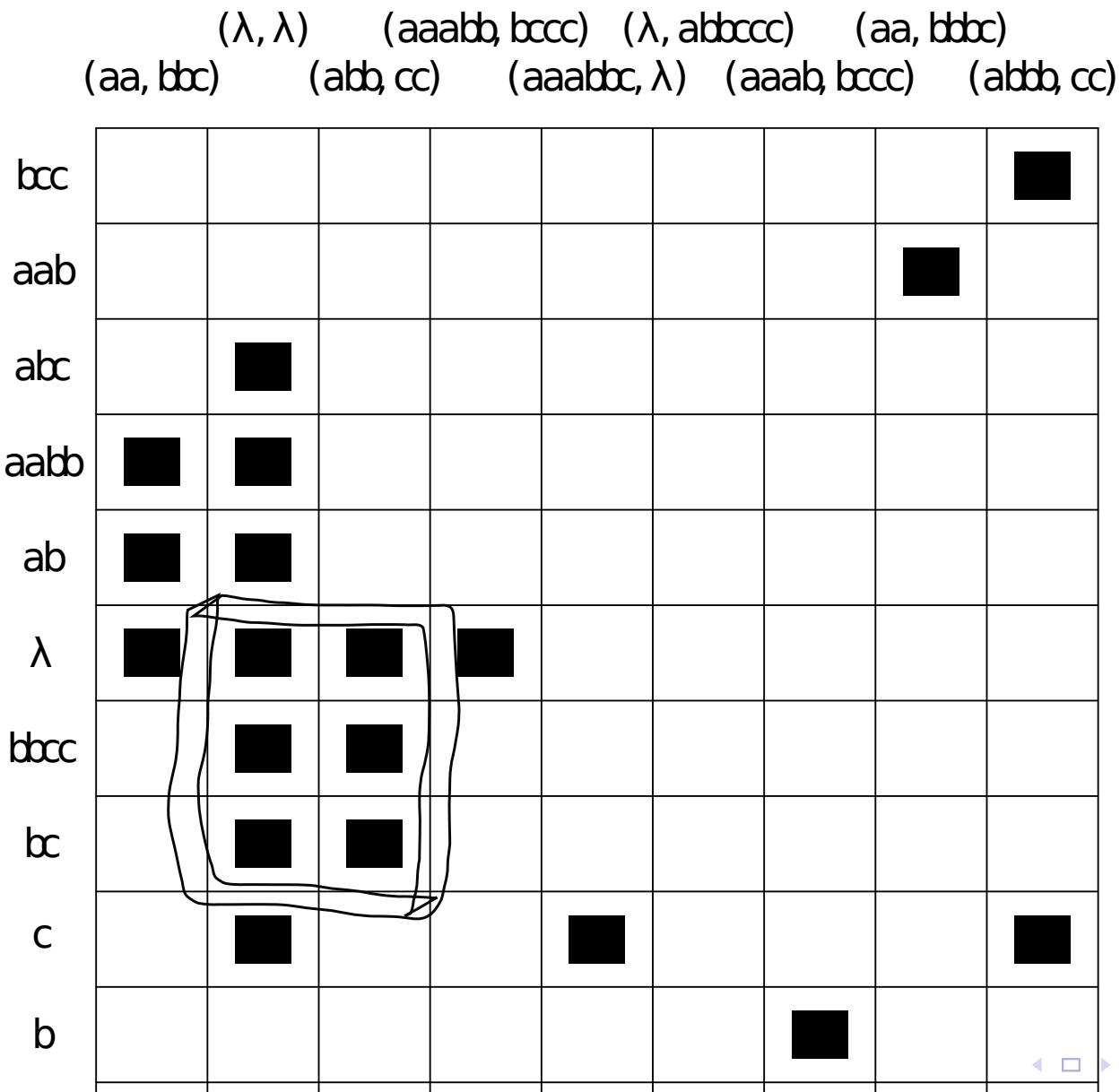
Example

$L = \{a^n b^n c^m | n, m \geq 0\} \cup \{a^m b^n c^n | n, m \geq 0\}$



Example

$L = \{a^n b^n c^m | n, m \geq 0\} \cup \{a^m b^n c^n | n, m \geq 0\}$



Learnability I

Search

Language is defined by choice of K and F
How can we find suitable K and F?

Lemma 1

as we increase K the language defined by $\langle K, L, F \rangle$ decreases monotonically
It will always converge to a subset of L in a finite time

Lemma 2

As we increase the set of contexts F the language monotonically increases.
Any sufficiently large set of contexts will do.

Search problem is trivial

Naive Algorithm

Start with $F = \{(\lambda, \lambda)\}, K = \Sigma \cup \{\lambda\}$

- If we see a string that is not in our hypothesis, the hypothesis is too small, and we add contexts to F
- Add strings to K if it will change the lattice at all.

Clark, (CoNLL, 2010)

DLGs can be learnt from positive data and MQs
Polynomial update time

Power of Representation

Language class

Let $\mathcal{L}$ be the set of all languages L such that there is a *finite* set of contexts F s.t. $L = L(\mathfrak{B}(\Sigma^*, L, F))$

Learnable class includes

1. All regular languages
2. Some but not all CFLs
3. Some non context free languages

Not in DLG?

$L = \{a^n b | n > 0\} \cup \{a^n c^m | m > n > 0\}$

Context sensitivity

- The lattice may have exponentially many elements in $|K| + |F|$
- (Exponentially many overlapping generalisations at different levels of generality)
- We cannot construct a CFG from all of them
- We can approximate parsing by taking the $\wedge$ of all of the elements in one slot of the chart
- This means we have a CS representation but we maintain a cubic time parsing algorithm

We are forced to move to a CS representation to *solve* a computational problem.

Why the delay?

The perceived problem

Not enough to learn a grammar, you have to learn the right grammar.
You have to learn constituent structure
Distributional structure is a product of hidden constituent structure.

Why the delay?

The perceived problem

Not enough to learn a grammar, you have to learn the right grammar.
You have to learn constituent structure
Distributional structure is a product of hidden constituent structure.

Maybe distributional structure is all there is?
If we can

- represent (weak) syntax
- learn
- support semantic interpretation?

Associative structures

[Bouma, 1989] These and other such arguments suggest that there is no such thing as a fixed constituent structure, but that the order in which elements combine with each other is rather free.

Associative structures

[Bouma, 1989] These and other such arguments suggest that there is no such thing as a fixed constituent structure, but that the order in which elements combine with each other is rather free.

- Many theories allow flexible constituency: multiple trees for the same unambiguous sentence
- Structural completeness is an advantage for processing.
- CCG (Steedman, 1997)
- Dependency grammar
- Associative Lambek calculus

Structural descriptions from the lattice

Residuated lattice

If we let K and F be all strings and contexts we get a residuated lattice with operators $Y \backslash X$ and X / Y

Admissible structures for a string w

Each span has a concept $\psi[i, j]$

- $\psi[i, j] \geq \mathcal{C}(w[i : j])$
- $\psi[i, j] \geq \bigwedge_k \psi[i, k] \circ \psi[k, j]$
- $\psi[0, l] \leq \mathcal{C}(L)$

Maximal structures

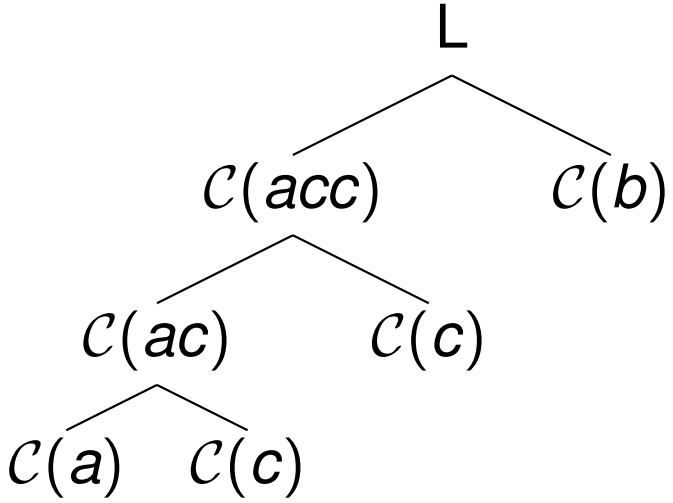
The set of maximal structures under the natural partial order can be viewed as the set of structural descriptions.
Discard $\top$ symbols and construct a graph or DAG.

Simple ambiguous language

Example

Dyck language with ambiguous symbol:
$\{ab, aabb, abab, aaababbb \ldots\}$
Add a symbol c which can be and a or a b
$\{ab, ac, cb, aabb, cccc, abab, aaacabbb \ldots\}$

Consider $accb$ – this could be $(ab)(ab)$ or $(a(ab)b)$.

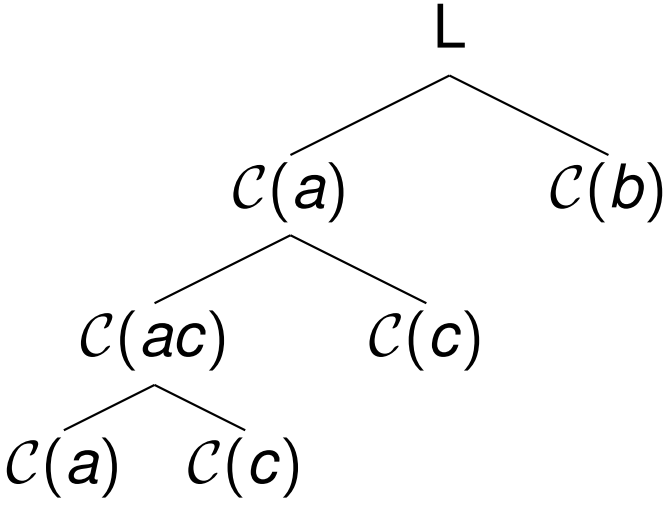


Simple ambiguous language

Example

Dyck language with ambiguous symbol:
$\{ab, aabb, abab, aaababbb \ldots\}$
Add a symbol c which can be and a or a b
$\{ab, ac, cb, aabb, cccc, abab, aaacabbb \ldots\}$

Consider $accb$ – this could be $(ab)(ab)$ or $(a(ab)b)$.

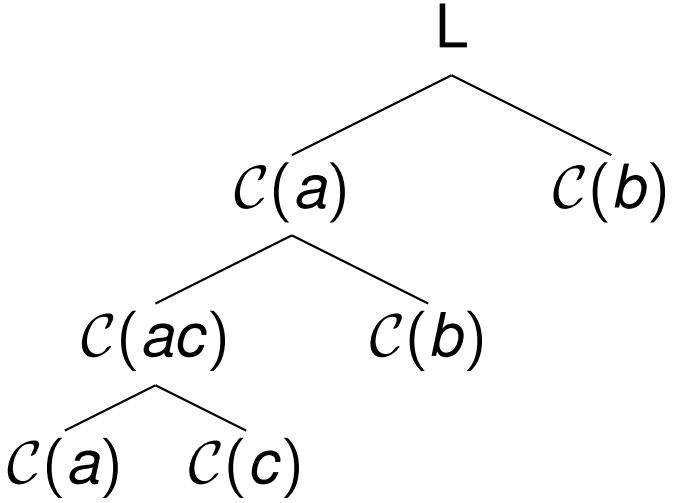


Simple ambiguous language

Example

Dyck language with ambiguous symbol:
$\{ab, aabb, abab, aaababbb \ldots\}$
Add a symbol c which can be and a or a b
$\{ab, ac, cb, aabb, cccc, abab, aaacabbb \ldots\}$

Consider $accb$ – this could be $(ab)(ab)$ or $(a(ab)b)$.

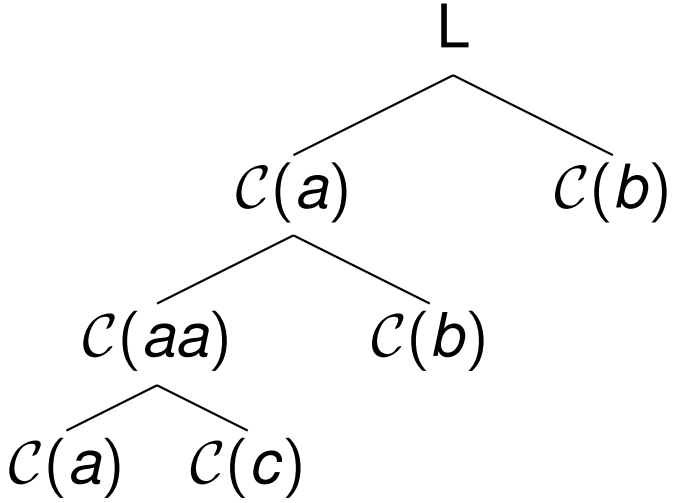


Simple ambiguous language

Example

Dyck language with ambiguous symbol:
$\{ab, aabb, abab, aaababbb \ldots\}$
Add a symbol c which can be and a or a b
$\{ab, ac, cb, aabb, cccc, abab, aaacabbb \ldots\}$

Consider $accb$ – this could be $(ab)(ab)$ or $(a(ab)b)$.

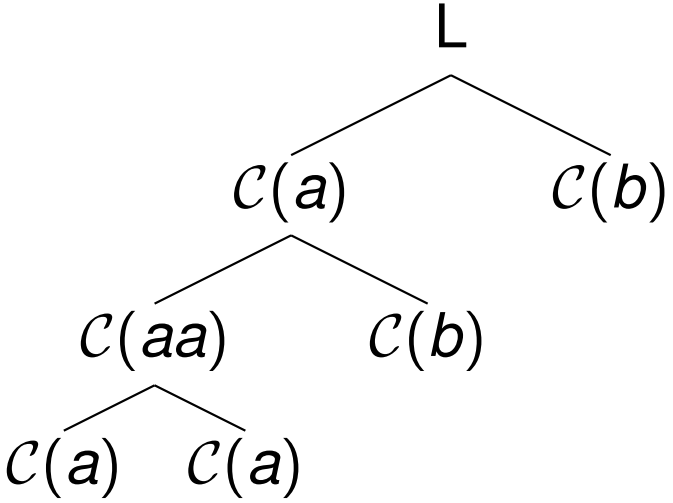


Simple ambiguous language

Example

Dyck language with ambiguous symbol:
$\{ab, aabb, abab, aaababbb \ldots\}$
Add a symbol c which can be and a or a b
$\{ab, ac, cb, aabb, cccc, abab, aaacabbb \ldots\}$

Consider $accb$ – this could be $(ab)(ab)$ or $(a(ab)b)$.

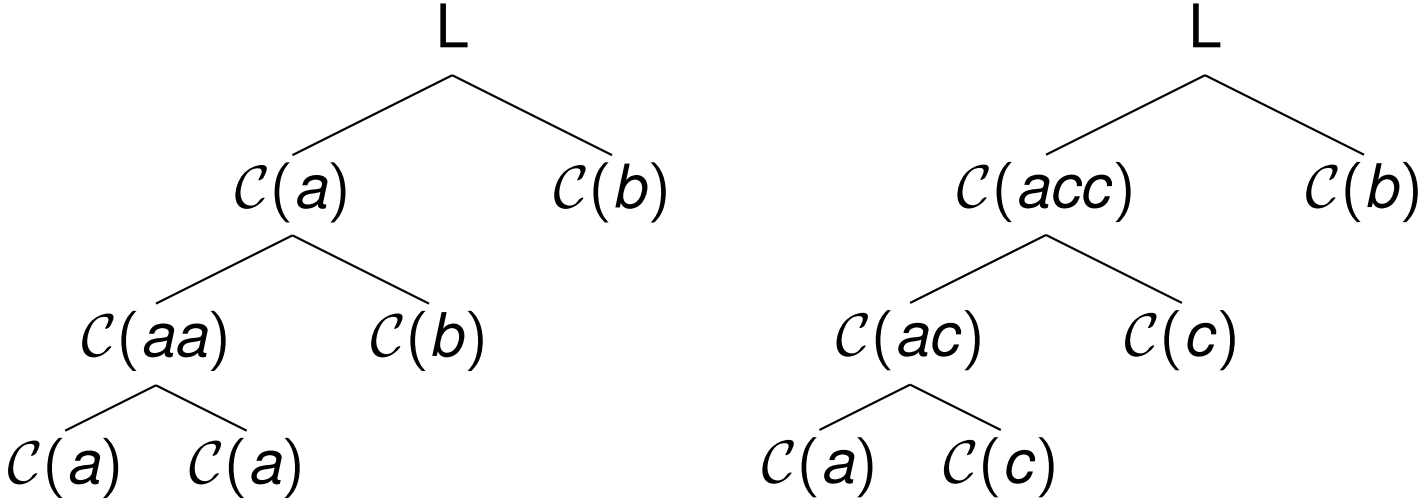


Simple ambiguous language

Example

Dyck language with ambiguous symbol:
$\{ab, aabb, abab, aaababbb \ldots\}$
Add a symbol c which can be and a or a b
$\{ab, ac, cb, aabb, cccc, abab, aaacabbb \ldots\}$

Consider $accb$ – this could be $(ab)(ab)$ or $(a(ab)b)$.

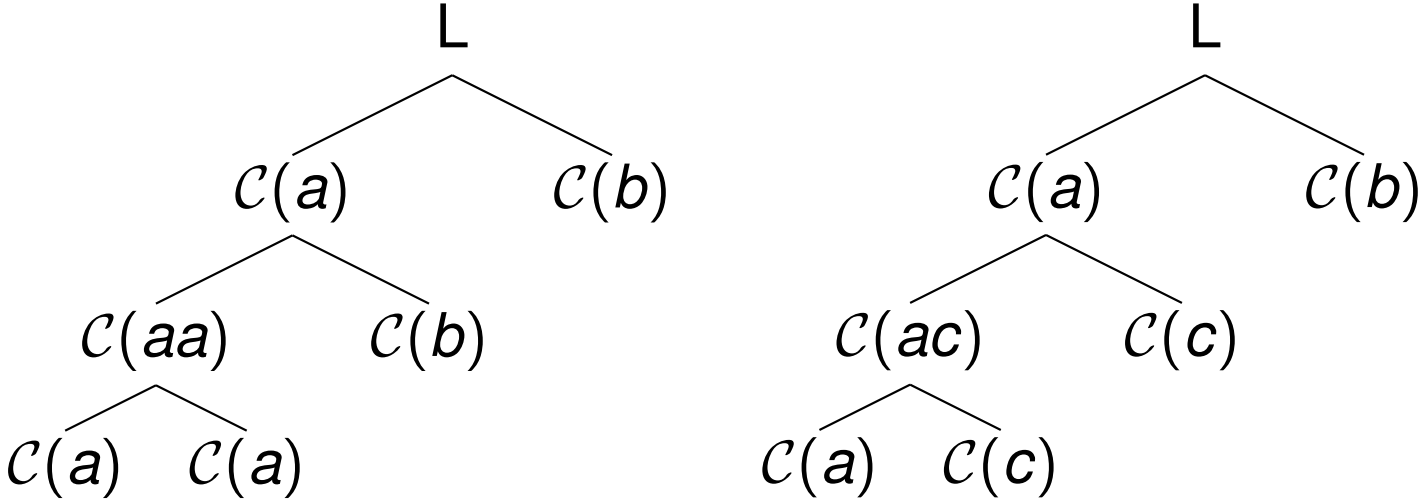


Simple ambiguous language

Example

Dyck language with ambiguous symbol:
$\{ab, aabb, abab, aaababbb \ldots\}$
Add a symbol c which can be and a or a b
$\{ab, ac, cb, aabb, cccc, abab, aaacabbb \ldots\}$

Consider $accb$ – this could be $(ab)(ab)$ or $(a(ab)b)$.

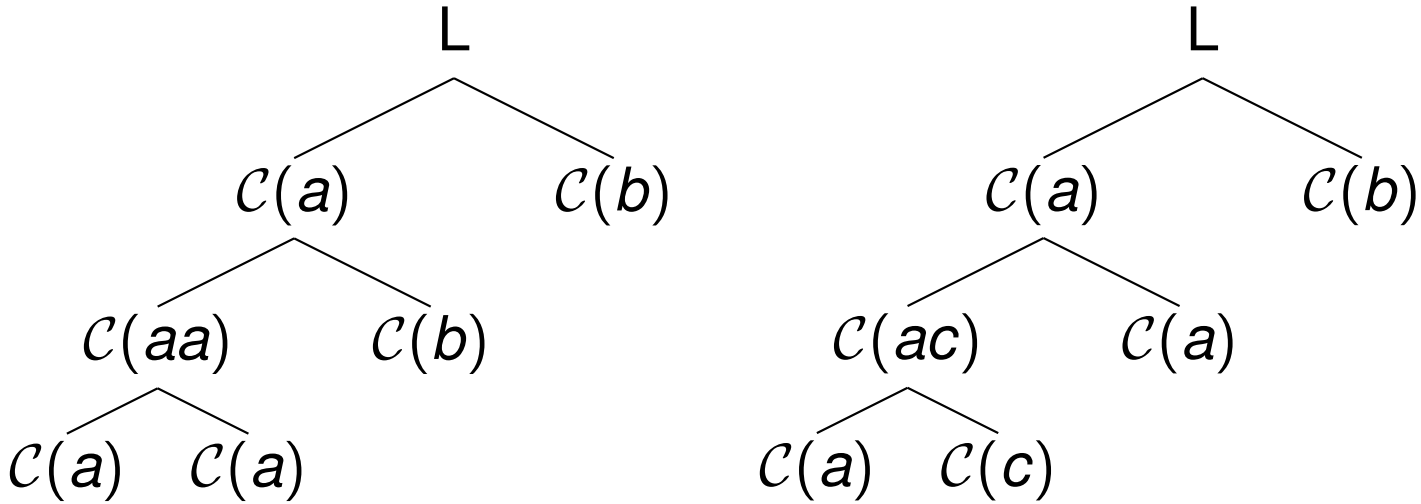


Simple ambiguous language

Example

Dyck language with ambiguous symbol:
$\{ab, aabb, abab, aaababbb \ldots\}$
Add a symbol c which can be and a or a b
$\{ab, ac, cb, aabb, cccc, abab, aaacabbb \ldots\}$

Consider $accb$ – this could be $(ab)(ab)$ or $(a(ab)b)$.

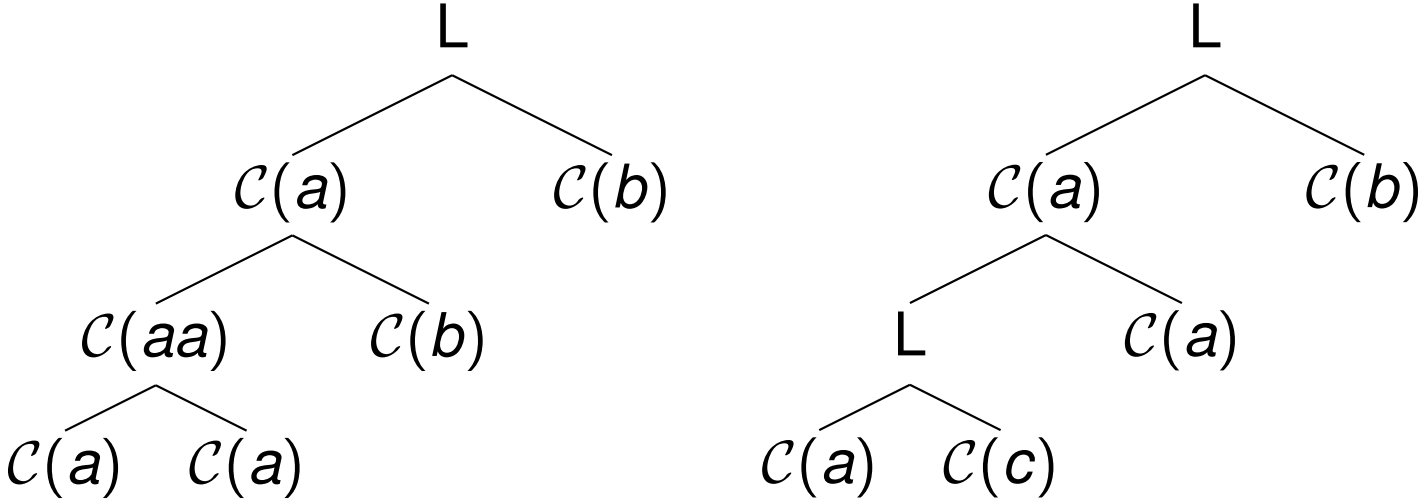


Simple ambiguous language

Example

Dyck language with ambiguous symbol:
$\{ab, aabb, abab, aaababbb \ldots\}$
Add a symbol c which can be and a or a b
$\{ab, ac, cb, aabb, cccc, abab, aaacabbb \ldots\}$

Consider $accb$ – this could be $(ab)(ab)$ or $(a(ab)b)$.

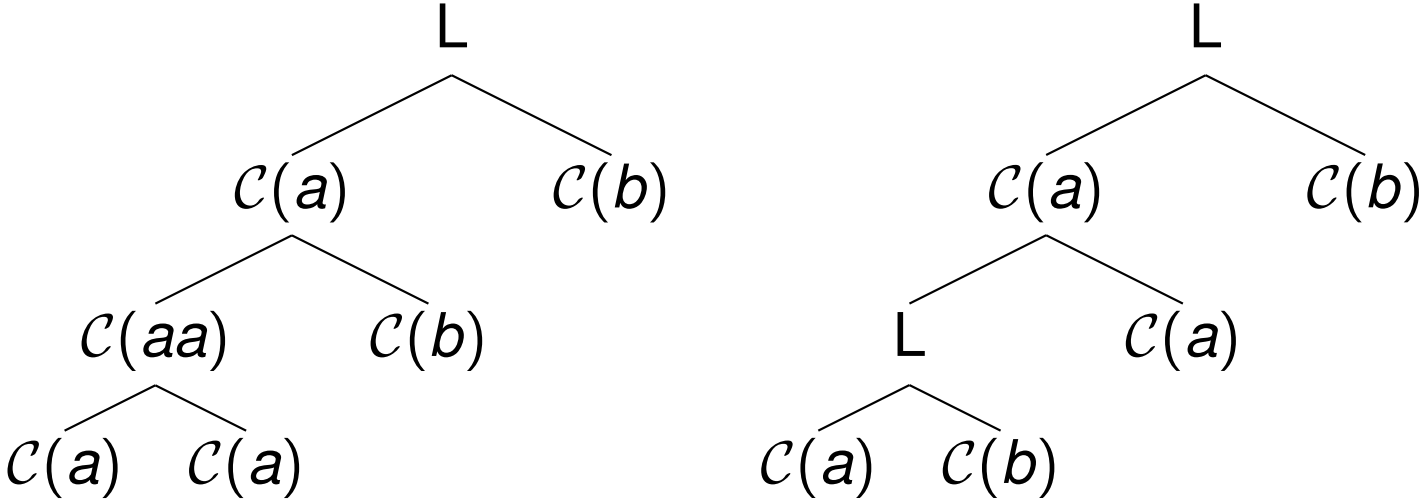


Simple ambiguous language

Example

Dyck language with ambiguous symbol:
$\{ab, aabb, abab, aaababbb \ldots\}$
Add a symbol c which can be and a or a b
$\{ab, ac, cb, aabb, cccc, abab, aaacabbb \ldots\}$

Consider $accb$ – this could be $(ab)(ab)$ or $(a(ab)b)$.



Tension between two notions of language
1950s

Abstraction - Chomsky

- Rich abstract structure that you need to model ambiguity etc.

Requires representations like CFGs, TAGs etc.

Learnability - Shannon

- Observable properties that mean you can learn

n-gram models

Tension between two notions of language
1950s

Abstraction - Chomsky

- Rich abstract structure that you need to model ambiguity etc.

Requires representations like CFGs, TAGs etc.

Learnability - Shannon

- Observable properties that mean you can learn

n-gram models

Not incompatible

There is a very rich abstract structure which is also observable and thus learnable.

Conclusion

Jackendoff (2008)

1. Descriptive constraint: the class of languages must be sufficiently rich to represent natural languages
2. Learnability constraint: there must be a way for the child to learn these representations from the data available
3. Evolutionary constraint: it must not posit a rich, evolutionarily implausible language faculty

- Distributional Lattice Grammars potentially satisfy all three criteria

Summary

- "Empiricism" – objective models
- Distributional learning provably works:
  1. The basic "one non-terminal per congruence class" model works
  2. A lattice based approach works even better but (and?) we need to move to a context sensitive representation

Acknowledgments

Shalom Lappin, Rémi Eyraud, Franck Thollard, Amaury Habrard