

ECS763U/P Natural Language Processing

Julian Hough

**Week 5: Generative
and Logical Grammars**

(with slides by
Mehrnoosh Sadrzadeh)

Why are sequence models not enough for natural language?

- We have looked at ways to model the probability of word/category sequences and how those models work in sequence to sequence classification.
- All these models have used notions of linear, contiguous dependency from one word to the next:
 - HMMs and n-gram Language Models used the **Markov assumption**: current word/category label is only dependent on k previous words/categories.
 - CRFs use a fixed contiguous **window** of words of width k to get a category.
- How would they capture the following sentence even if $k=10$?:
 - “the computer I just put into the machine room on the fifth floor crashed”

Why are sequence models not enough for natural language?

KEY POINT:

Language has
hierarchical
structure

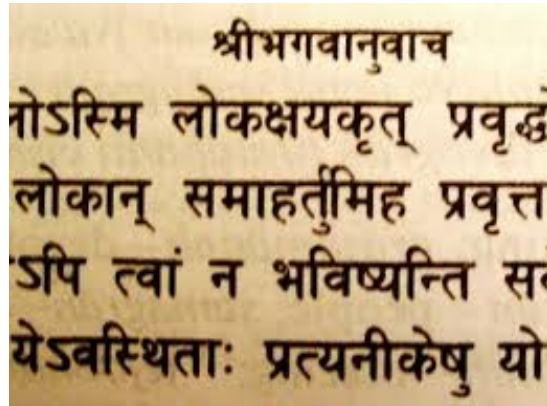
Why are sequence models not enough for natural language?

KEY POINT:

Formal grammars
were designed to
capture the
structure of NL

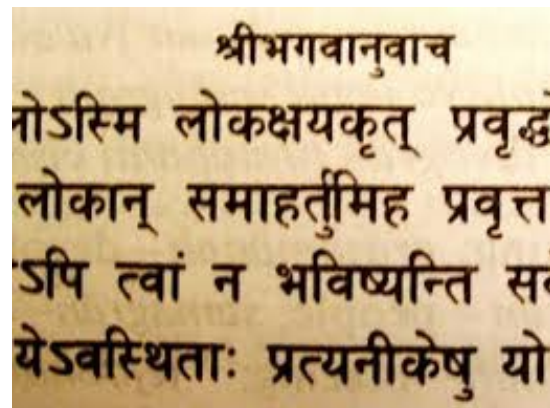
A bit of history of Formal Grammar

- The first formal grammar was written over 2000 years ago for Sanskrit by Panini. But it is still referenced today when teaching Sanskrit and studying its grammar.



A bit of history of Formal Grammar

- Panini's book had just under 4000 commented algebraic statements, an example of which is the following one:



Sanskrit has ten classes of verbs divided into two broad groups: athematic and thematic. The thematic verbs are so called because an 'a', called the theme vowel, is inserted between the stem and the ending. This serves to make the thematic verbs generally more regular.



A bit of history of Formal Grammar

- Formal grammar is sometimes referred to as the study of “syntax” (vs *semantics* or pragmatics).
- The word “syntax” originates from the Greek word “**SYNTAXIX**”, which means “setting out together” or “arrangement”

e.g. the syntax of chairs around a table,
the syntax of plates on a table,
the syntax of flowers in a pot,

A bit of history of Formal Grammar

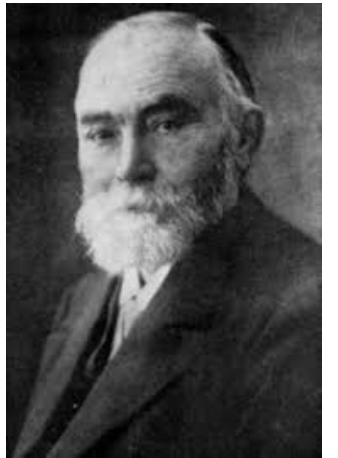
- Formal grammar is sometimes referred to as the study of “syntax” (vs *semantics* or pragmatics).
- The word “syntax” originates from the Greek word “**SYNTAXIS**”, which means “setting out together” or “arrangement”

In a linguistic context, this word is used to refer to
“the ways words are arranged together”
e.g. in the sentences and other constructions of
language.

A bit of history of Formal Grammar

- Modern founders of formal approaches to language were **G. Frege** (1848-1925) and **F. de Saussure** (1857-1913).
- Frege introduced predicate logic and used it to do a functional analysis of language using diagrams.
- He was also the first person who introduced a, now widely used, **principle of compositionality**:

The meaning of a complex term is a function of meanings of its parts.



A bit of history of Formal Grammar



- de Saussure was the first person who provided a **formalisation of communication using language:**

A speaker expresses a psychological idea using a physiological act, which produces a signal that is transferred to a hearer. Upon hearing this signal, the hearer uses the physiological impressions received and recovers the idea. For the communication to be a success, both have to have shared associations between form and meaning.

Logical Grammars

- The first modern formal grammars were introduced by mathematicians interested in language - Ajdukiewicz in 1935.
- Naturally, they used concepts from mathematics, namely that of division and multiplication to write down a formalisation of rules of grammar.
- The grammars resulting from this line of research are known under the term “**logical**” or “**categorical**” grammars.

Generative Grammars

The most widely used formal grammars were motivated by the work of one of the fathers of psychology:

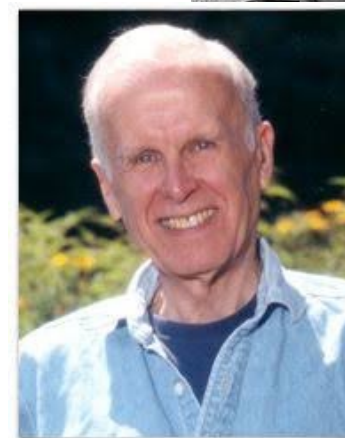
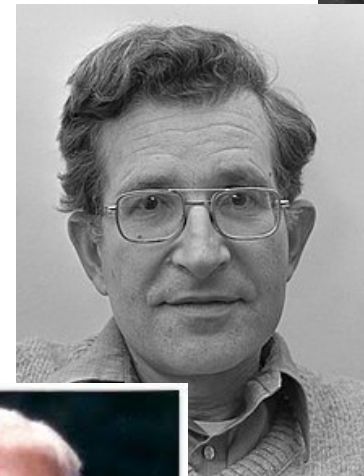
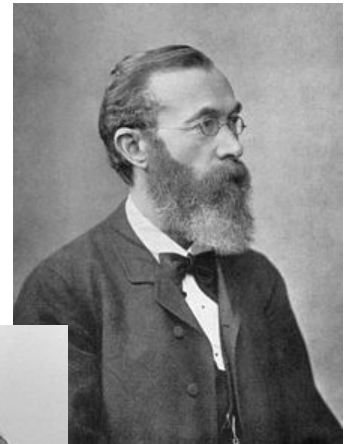
Wilhem Wundt in 1900.

A linguist formalised these ideas using a set of generative rules, starting modern linguistics:

Noam Chomsky in 1955-57

They were later also expressed by a computer scientist:

Backus in 1959.



Generative Grammars

KEY POINT:

The main concept
underlying generative
grammars is
constituency.

Formal Grammar

- Groups of words that behave as a single unit are called a **constituent**.
- A significant part of developing a grammar involves discovering the constituents present in a particular corpus, or in a language more generally.

Formal Grammar

Examples of Constituents in English:

Noun Phrases

Harry the Horse
the Broadway coppers
they

a high-class spot such as Mindy's
the reason he comes into the Hot Box
three parties from Brooklyn

How do we know which words group together?

- They appear in similar syntactic positions, e.g. before a verb:

three parties from Brooklyn *arrive...*
a high-class spot such as Mindy's *attracts...*
the Broadway coppers *love...*
they *sit*

Formal Grammar

Examples of Constituents in English:

Noun Phrases

How do we know which words group together?

- They appear in similar syntactic positions, e.g. before a verb:
- But not true for the individual words in the constituent. It doesn't make sense to say:

*from *arrive*... *as *attracts*...
*the *is*... *spot *sat*...

Formal Grammar

Examples of Constituents in English:

Prepositional Phrases

On September seventeenth,

- They appear in different syntactic positions in the sentence:

On September seventeenth, I'd like to fly from Atlanta to Denver

I'd like to fly *on September seventeenth* from Atlanta to Denver

I'd like to fly from Atlanta to Denver *on September seventeenth*

Formal Grammar

Examples of Constituents in English:

Prepositional Phrases

- They appear in different syntactic positions in the sentence.
- But again, not true for the individual words in the constituent.
It doesn't make sense to say:

*On September, I'd like to fly seventeenth from Atlanta to Denver

*On I'd like to fly September seventeenth from Atlanta to Denver

*I'd like to fly on September from Atlanta to Denver seventeenth

Formal Grammar

Grammatical Relations

These are relationships between the constituents.
Examples are **Subject** and **Object**.

For example in the sentence:

“She adores the deep blue sky”,

“she” and “the deep blue sky” are noun phrase constituents that are the **subject** and the **object** of verb “adores”.

Formal Grammar

Dependency Relations

Special types of **relations** between **words and phrases**.

e.g. the verb “want” can be followed by an **infinitive**:

“I want to sleep.”

It can also be followed by a noun phrase:

“I want a sleeping bag.”

This is not the case for all verbs, for example the verb “find”, cannot be followed by an infinitive. One cannot say:

*“I find to fly to Edinburgh.”

Context Free Grammars

Context free grammars (from the **Chomsky hierarchy** of formal languages) are the backbone of many formal models of syntax.

Reasons:

1- **Powerful** enough to formalise (most) relationships between words in a sentence.

2- **Computationally tractable enough** for implementation, so there exists lots of parsing algorithms and tools for them.

Context Free Grammars

Context free grammars or CFG's are also called

Phrase Structure Grammars.

Their formalisation is similar to

Backus-Naur Form (BNF)

Context Free Grammars

A CFG has:

a set of **production rules**: how constituents of language are grouped and ordered together

a lexicon: what constituents words of language are part of.

Context Free Grammars

Example:
production rules:

NP -> ProperNoun
NP -> Det Nominal

A noun phrase NP can be generated from either a ProperNoun, or a determiner followed by a Nominal.

Context Free Grammars

Example:

lexicon:



Det -> a
Det -> the

Words “a” and “the” are determiners.

Context Free Grammars

We can combine the production rules with the lexicon and other production rules to form phrases and sentences.

“a” can be a determiner,
“flight” can be a Noun

“a flight” can be a noun phrase

Det -> a
Noun -> flight
NP -> Det Nominal

Context Free Grammars

The symbols on the left hand side of lexical rules are the **lexical categories** of words.

```
Det -> a
Def -> the
Det -> this
Det -> that
Noun -> flight
Noun -> morning
Noun -> star
```

```
Det -> a | the | this | that
Noun -> flight | morning | star
```

Rules with the same left hand side can also be denoted using the delimiter | to save space. This form is often used for lexical rules.

Context Free Grammars

A CFG can be thought of as:

Generating sentences of language in the lexicon.

In generation, the rules are treated as **rewrite rules**.

NP → Det Nom	rewrites NP to Det Nom
Nominal → Noun	rewrites Det Nom to Det Noun
Noun → flight	rewrites Det Noun to Det flight
Det → a	rewrites Det flight to a flight

Context Free Grammars

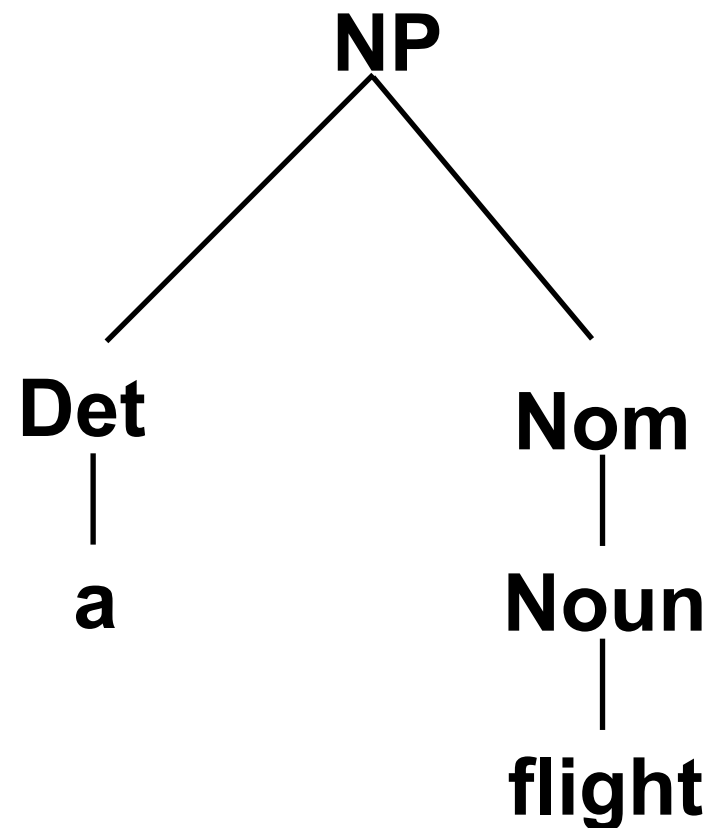
Here, the terminology we use is:

- The string “a flight” can be **derived from** non-terminal NP.
- The sequence of rules is called a **derivation**.
- The **language** of a CFG is the set of strings that are derivable from the designated start symbol S.
- The set of strings derived from S is the set of sentences of a simplified version of English.

Context Free Grammars

A set of possible derivations is commonly represented by
a **parse tree**.

For example the parse tree of “a flight” is:



Context Free Grammars

A few more rules for the grammar of English

e.g.

S → NP VP
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
PP → Preposition NP
Pronoun → me | I | you | it

I prefer a morning flight
prefer a morning flight
leave London at noon
Leave on Sunday

From London
on Wednesday
On July 16th

Context Free Grammars

PP's are often used with times, dates, and locations.

PP -> Preposition NP

From London
on Wednesday
On July 16th

When used with other nouns, they get quite complex:

to Seattle
in Minneapolis
on Wednesday
in the evening
on the ninth of July

on these flights
about the ground transportation in Chicago
of the round trip flight on United Airlines
of the AP fifty seven flight
with a stopover in Nashville

Context Free Grammars

A sample lexicon:

Noun → *flights* | *breeze* | *trip* | *morning*
Verb → *is* | *prefer* | *like* | *need* | *want* | *fly*
Adjective → *cheapest* | *non-stop* | *first* | *latest*
 | *other* | *direct*
Pronoun → *me* | *I* | *you* | *it*
Proper-Noun → *Alaska* | *Baltimore* | *Los Angeles*
 | *Chicago* | *United* | *American*
Determiner → *the* | *a* | *an* | *this* | *these* | *that*
Preposition → *from* | *to* | *on* | *near*
Conjunction → *and* | *or* | *but*

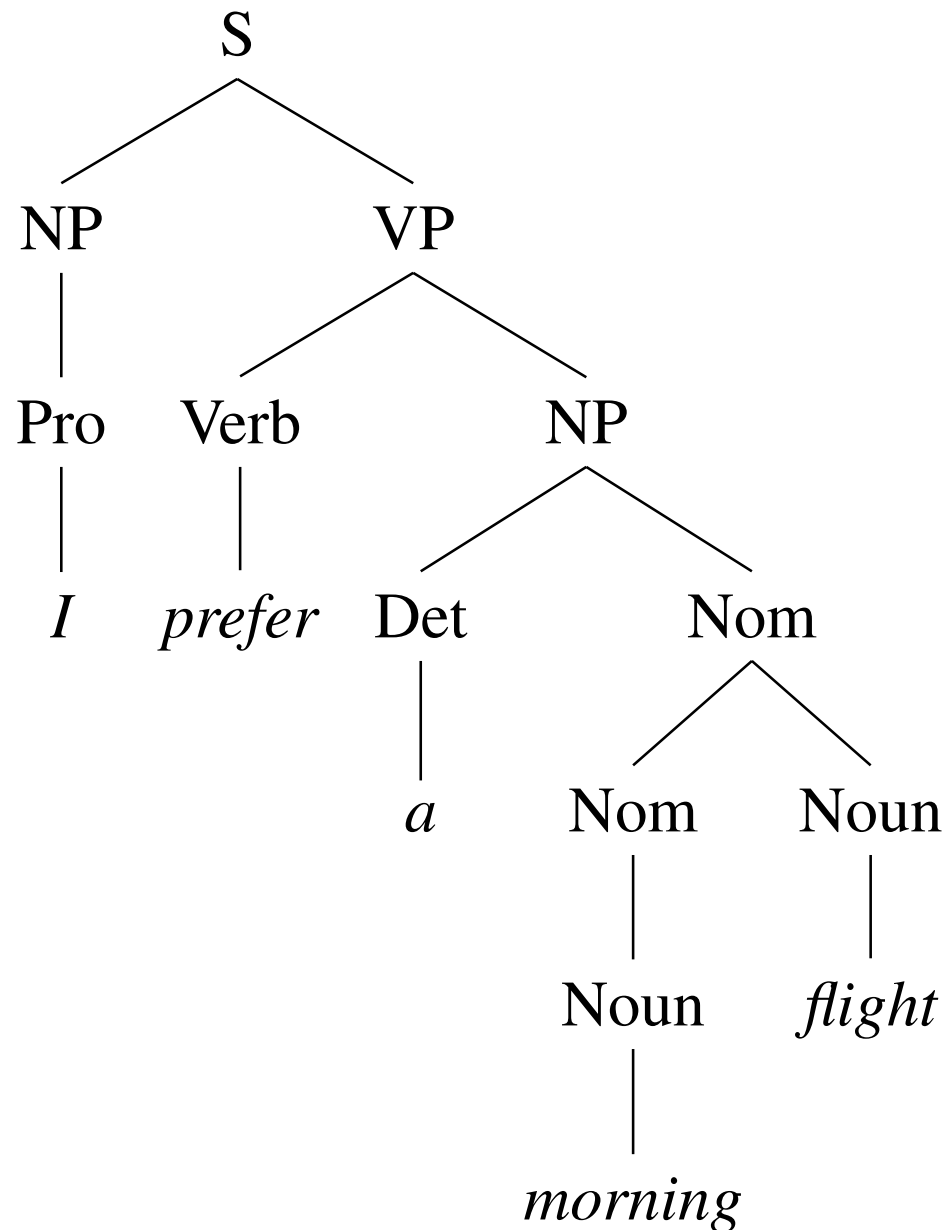
Context Free Grammars

Some grammatical rules:

Grammar Rules	Examples
$S \rightarrow NP VP$	I + want a morning flight
$NP \rightarrow$ <i>Pronoun</i> <i>Proper-Noun</i> <i>Det Nominal</i>	I Los Angeles a + flight
$Nominal \rightarrow$ <i>Nominal Noun</i> <i>Noun</i>	morning + flight flights
$VP \rightarrow$ <i>Verb</i> <i>Verb NP</i> <i>Verb NP PP</i> <i>Verb PP</i>	do want + a flight leave + Boston + in the morning leaving + on Thursday
$PP \rightarrow$ <i>Preposition NP</i>	from + Los Angeles

Context Free Grammars

The **parse tree** for “I prefer a morning flight.”

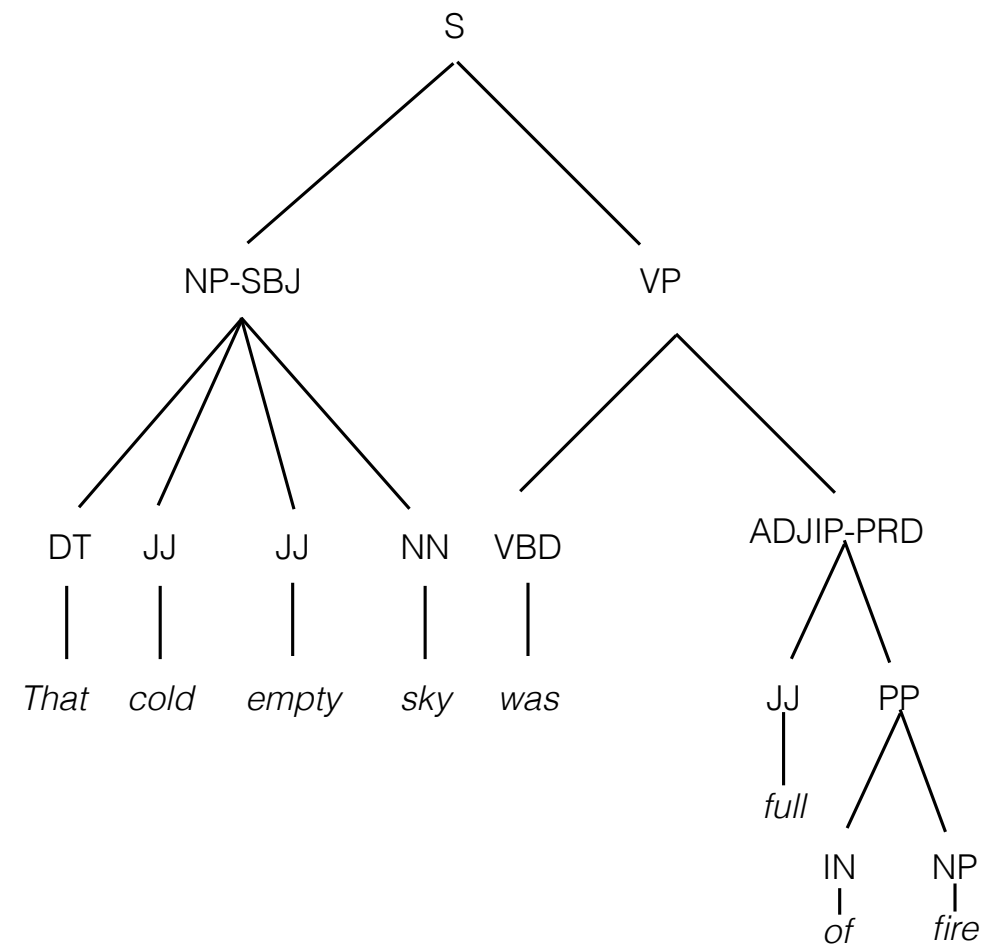


Context Free Grammars

The **bracketed notation** for the parse tree:

$[_S [_{NP} [_{Pro} I]] [_{VP} [_{V} prefer] [_{NP} [_{Det} a] [_{Nom} [_{N} morning] [_{Nom} [_{N} flight]]]]]]]$

A more complicated example:



“The cold, empty sky was full of fire”

A more complicated example:

((S
 (NP-SBJ (DT The)
 (JJ long) (, ,)
 (JJ lonely) (NN night))
 (VP (VBD is)
 (ADJP-PRD (JJ full)
 (PP (IN of)
 (NP (NN stars)
 (CC and)
 (NN moonlight)))))
 (. .)))

The long, lonely night is full of stars and moonlight.

Formal Definition of a CFG

Formal definition of a CFG:

$$(N, \Sigma, R, S)$$

N **a set of non-terminal symbols**

Σ **a set of terminal symbols (disjoint from N)**

S **a designated start symbol**

R **a set of production rules of the form** $\alpha \rightarrow \beta$

α **a non-terminal**

β **a string of symbols from the strings** $(\Sigma \cup N)^*$

Formal Definition of a CFG

Formal definition of a CFG:

$$(N, \Sigma, R, S)$$

Notational conventions:

Capital letters like A , B , and S

S

Lower-case Greek letters like α , β , and γ

Lower-case Roman letters like u , v , and w

Non-terminals

The start symbol

Strings drawn from $(\Sigma \cup N)^*$

Strings of terminals

Formal Definition of a CFG

The symbols of a CFG are classified into two groups:

1- Terminals:

These correspond to the words of language.
The words are introduced via these rules in the lexicon.
e.g. flight, morning, star, a, the, this, that

2- Non-Terminals:

Symbols that express generalisations of these.
a.g. S, NP, VP, Noun, Det

3- and of course we have the -> symbol

Formal Definition of a CFG

In a CFG:

The items to the right of \rightarrow are:
an ordered list of one or more T's or NT's

The item to the left of \rightarrow is a
a single NT

This is what makes the language generated by these rules context free: there is no context for the application of the rule: on the left we only have a single NT.

Formal Definition of a CFG

A language is defined through the concept of **derivation**.

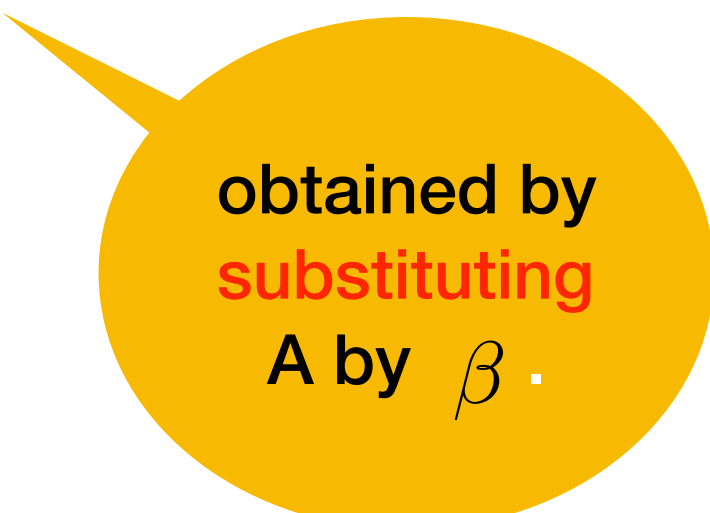
A string **derives** another if it can be **rewritten** as the second one by a series of rule applications.

If $A \rightarrow \beta$ is a production rule and α and γ are any two strings in $(\Sigma \cup N)^*$, then we say:

$\alpha A \gamma$ **directly derives** $\alpha \beta \gamma$

This is more formally denoted by:

$$\alpha A \gamma \Longrightarrow \alpha \beta \gamma$$



obtained by
substituting
A by β .

Formal Definition of a CFG

A derivation is a generalisation of a direct derivation. If we have $\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \dots, \alpha_{n-1} \Rightarrow \alpha_n$

then, we say:

α_1 **derives** α_n

and formally write:

$$\alpha_1 \xRightarrow{*} \alpha_n$$

Parsing is the problem of mapping a string of words to its derivation.
(More next week)

Formal Definition of a CFG

- The **language** generated by a CFG is the set of strings composed of terminals that can be derived from the designated start symbol. A **context-free language** is therefore:

$$\mathcal{L}_{CFG} = \left\{ w \mid w \in \Sigma^* \quad \text{and} \quad S \xRightarrow{*} w \right\}$$

Formal Definition of a CFG

- Sentences (strings of words) that can be derived by a grammar are in the formal language defined by that grammar, and are called **grammatical sentences**.
- Sentences that cannot be derived by a given formal grammar are not in the language defined by that grammar and are referred to as **ungrammatical**.
- In linguistics, this is called a **generative grammar** since the language is defined the set of possible sentences “generated” by the grammar.
- In the next lecture we go beyond ‘in’ and ‘out’ binary notions of grammaticality and move towards **probabilistic notions of grammaticality**.

More sentence types

Imperative:
S → VP

Show the lowest fare
Give me the morning flights

Yes-No Question:
S → Aux NP VP

Does American fly to Boston?

Wh Question:
S → Wh-NP VP

What airlines fly to Boston?

Logical Grammars

- The first logical grammar was formalised by Ajdukiewicz in 1935 and only had one rule:

$$B|A \quad A \Rightarrow B$$

- This rule says that when an expression of grammatical type A is preceded by an expression of type $B|A$, then we obtain an expression of type B .

Logical Grammars

Ajdukiewicz's rule can be thought of like **multiplying a fraction**:

$$B|A \times A = B$$

$B|A$ can be thought of as the fraction: B over A

The rule can then be thought of as multiplication: when B over A is multiplied by A, we get B.

Logical Grammars

Ajdukiewicz called this rule a **cancelation scheme**.

He defined a notion of **grammaticality** as follows:

A string of words has a satisfying syntactic connection, iff some ordering of its word types reduced to the distinguished type S (of sentence) via successive uses of the cancelation scheme.

Logical Grammars

Some basic **logical types**:

NP: noun phrase

S: sentence

Some **types assignments**:

flight: NP

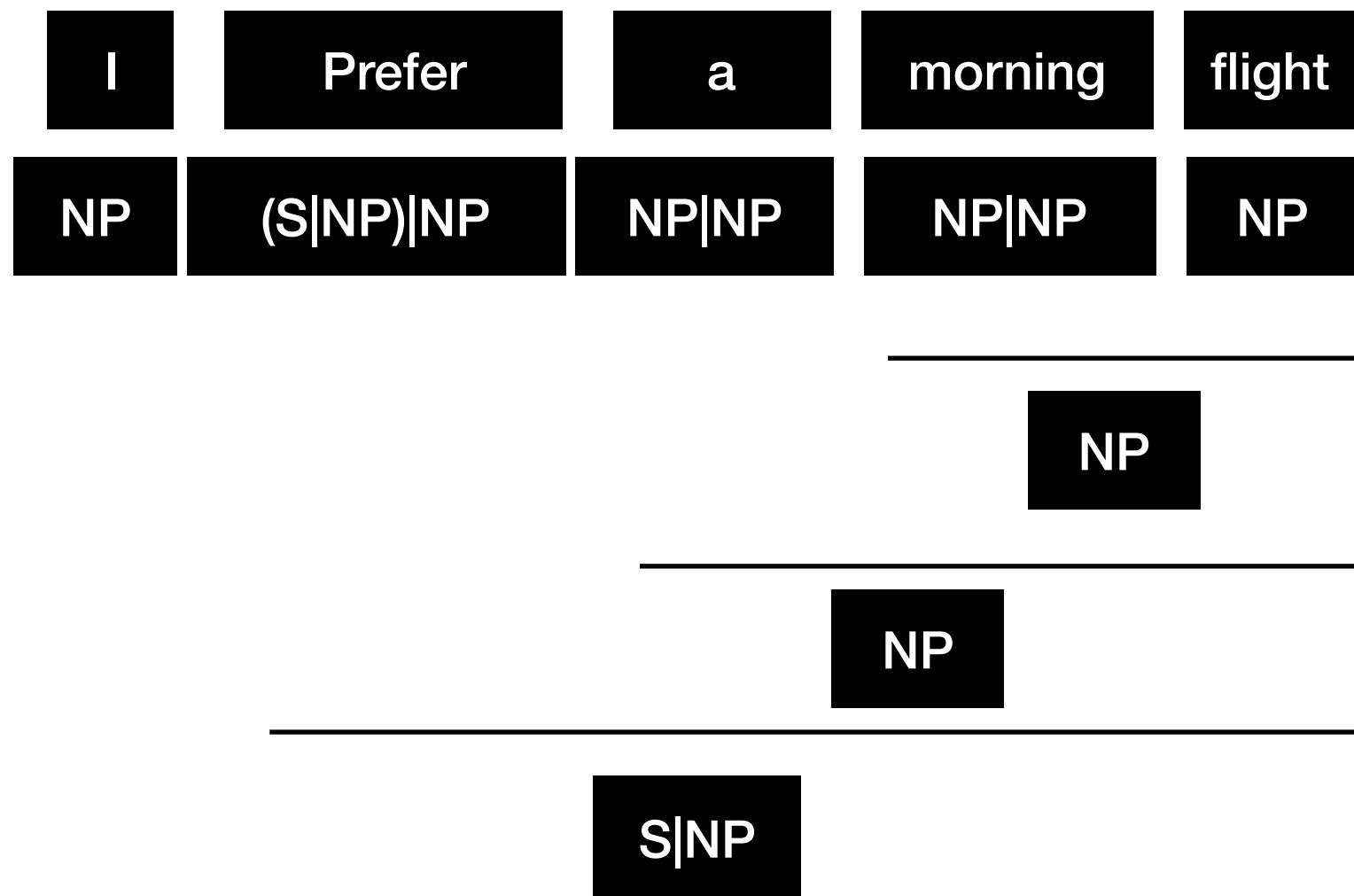
morning: NP|NP

a: NP|NP

prefer: (S|NP)|NP

Logical Grammars

An example of a string of words with a satisfying syntactic connection:



Logical Grammars

An example of a string of words with a satisfying syntactic connection:

I	Prefer	a	morning	flight
NP	(S NP) NP	NP NP	NP NP	NP

and now?

NP

NP

S|NP

Logical Grammars

An example of a string of words with a satisfying syntactic connection:

I	Prefer	a	morning	flight
NP	(S NP) NP	NP NP	NP NP	NP

well, the rule allows for
reordering of the string, so

...

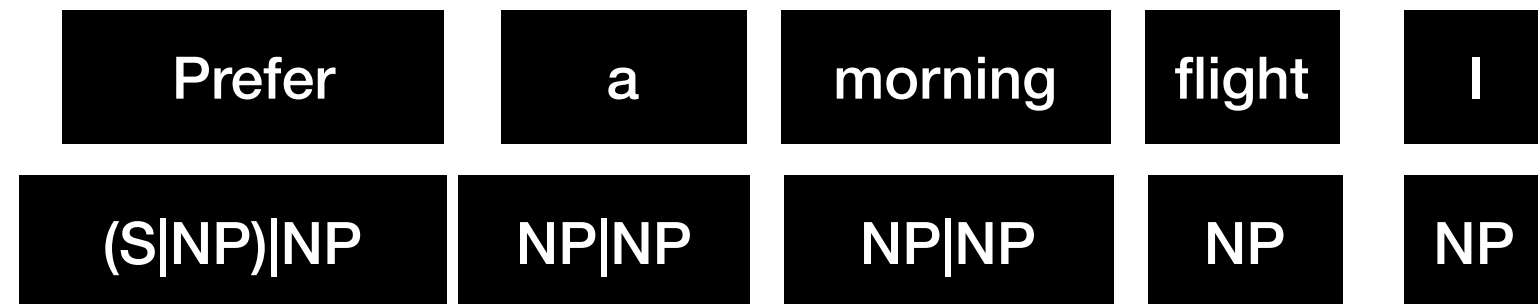
NP

NP

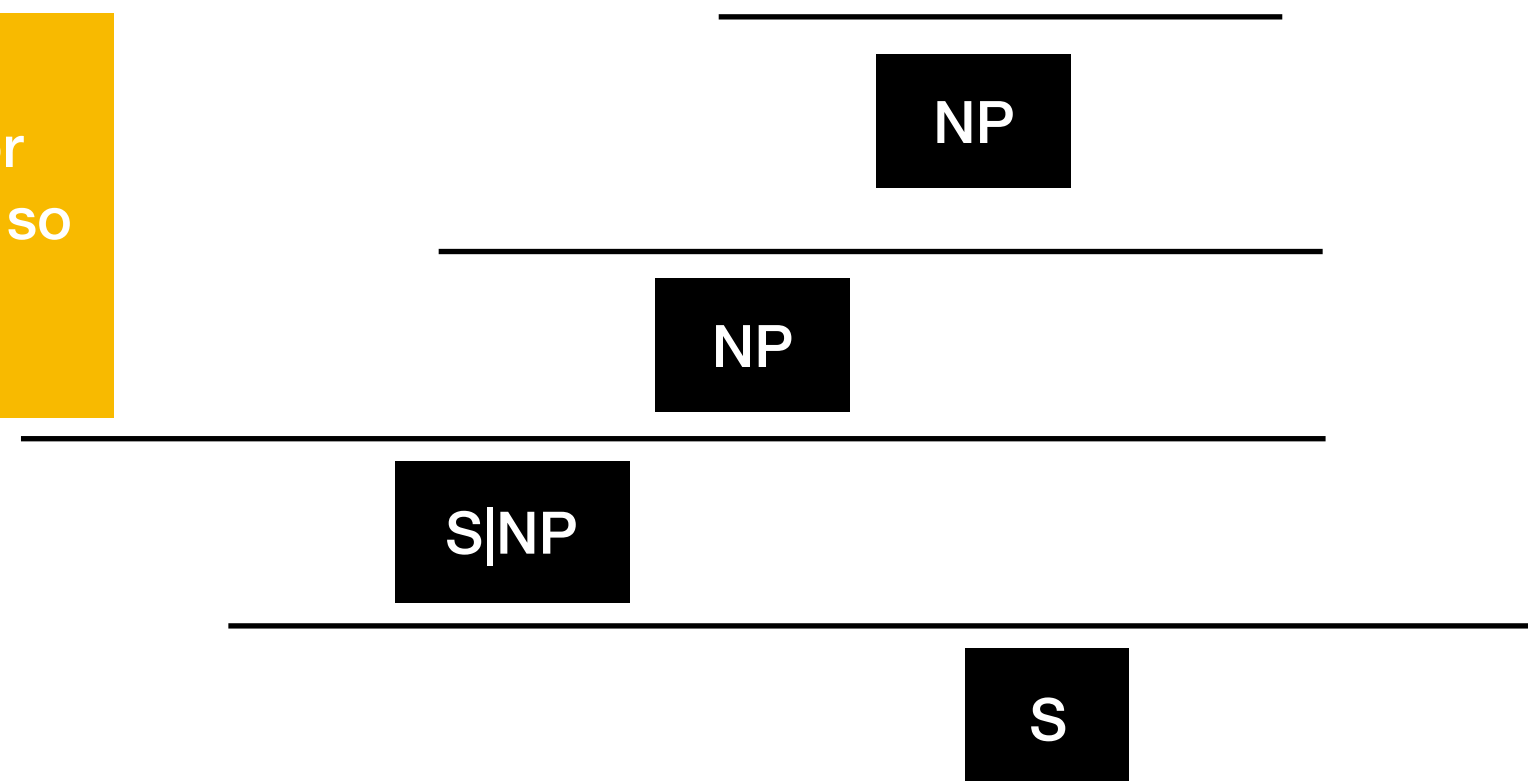
S|NP

Logical Grammars

An example of a string of words with a satisfying syntactic connection:



well, the rule allows for reordering of the string, so



Logical Grammars

In order to make the calculus more refined, Bar-Hillel (1953) introduced directional division types: $A \backslash B$ and B / A and introduced a **directional** version of the cancelation schema:

$$A \quad A \backslash B \quad \Rightarrow \quad B$$

$$B / A \quad A \quad \Rightarrow \quad B$$

The resulting system is called the **AB calculus**.

Logical Grammars

Some basic **logical types**:

CN: common noun

N: nominal

S: sentence

Some **types assignments** using **directional types**:

flight: NP

morning: NP/NP

a: NP/NP

prefer: (NP\S)/NP

Logical Grammars

Deriving the satisfying syntactic connection using the directional types:

I	Prefer	a	morning	flight
NP	(NP\S)/NP	NP/NP	NP/NP	NP

and now no reordering is needed

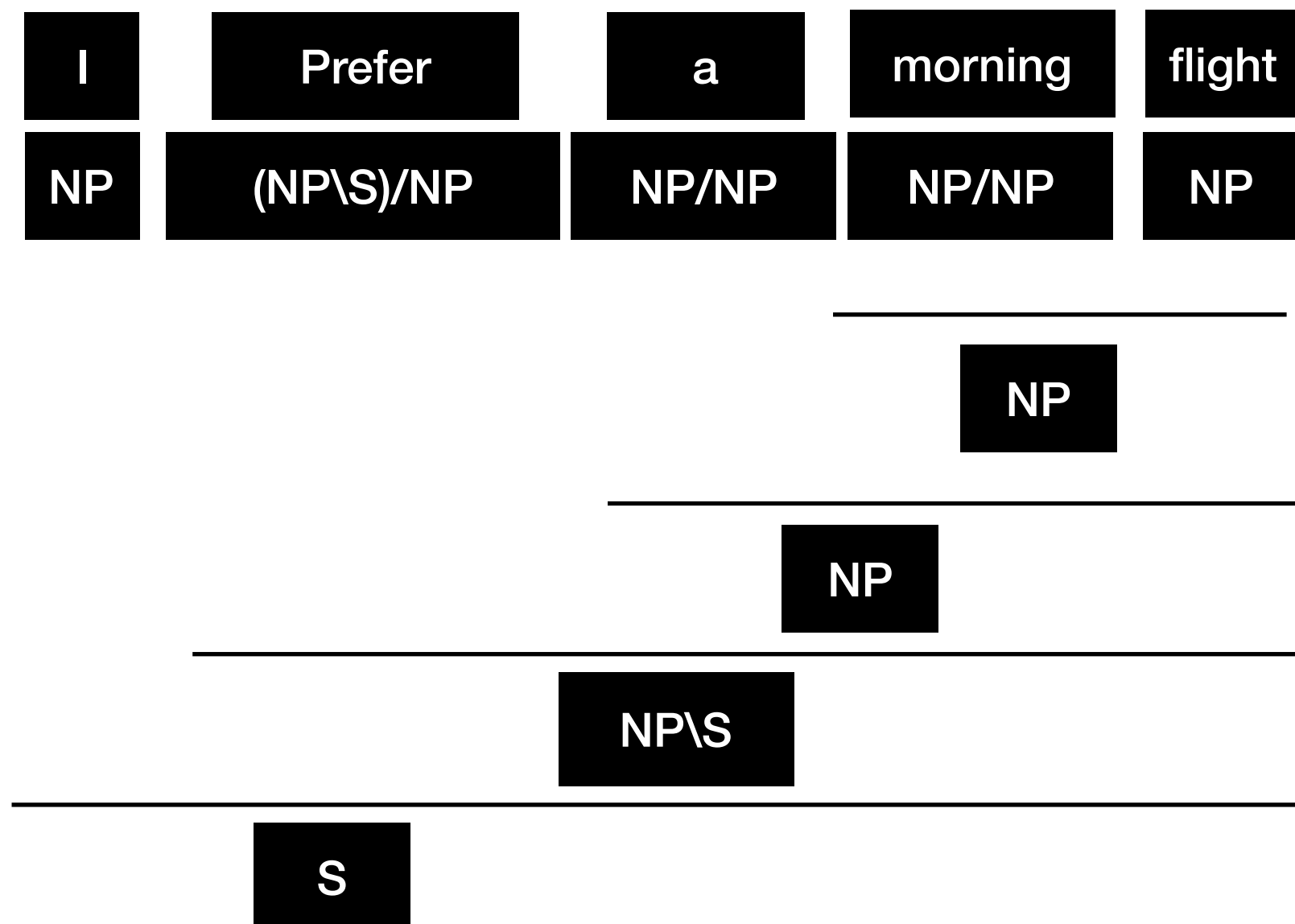
NP

NP

NP\S

Logical Grammars

An example of a string of words with a satisfying syntactic connection:



Logical Grammars

An extended set of basic **logical types**:

NP: noun phrase

S: sentence

PP: prepositional phrase

More advanced **type assignment**:

we, flight, Geneva, Chamonix: NP

to: PP/NP

and: $X \backslash (X/X)$

flew,drove: $(NP \backslash S)/PP$

X can be any type, basic or complex:

$X = NP$, $X = S$, $X = PP$

$X = PP/NP$, $X = NP \backslash S$, etc...

Logical Grammars

A more elaborate example:

We	flew	to	Geneva	and	drove	to	Chamonix
NP	(NP\S)/PP	PP/NP	NP	X\X/X	(NP\S)/NP	PP/NP	NP

Logical Grammars

A more elaborate example:

We	flew	to	Geneva	and	drove	to	Chamonix
NP	(NP\S)/PP	PP/NP	NP	X\X/X	(NP\S)/NP	PP/NP	NP

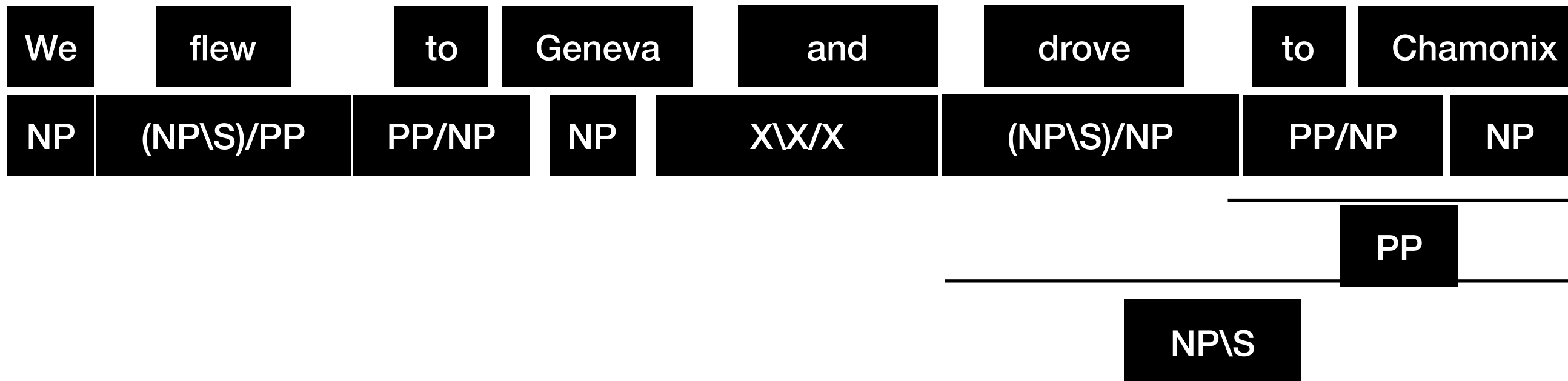
Logical Grammars

A more elaborate example:

We	flew	to	Geneva	and	drove	to	Chamonix
NP	(NP\S)/PP	PP/NP	NP	X\X/X	(NP\S)/NP	PP/NP	NP
						PP	

Logical Grammars

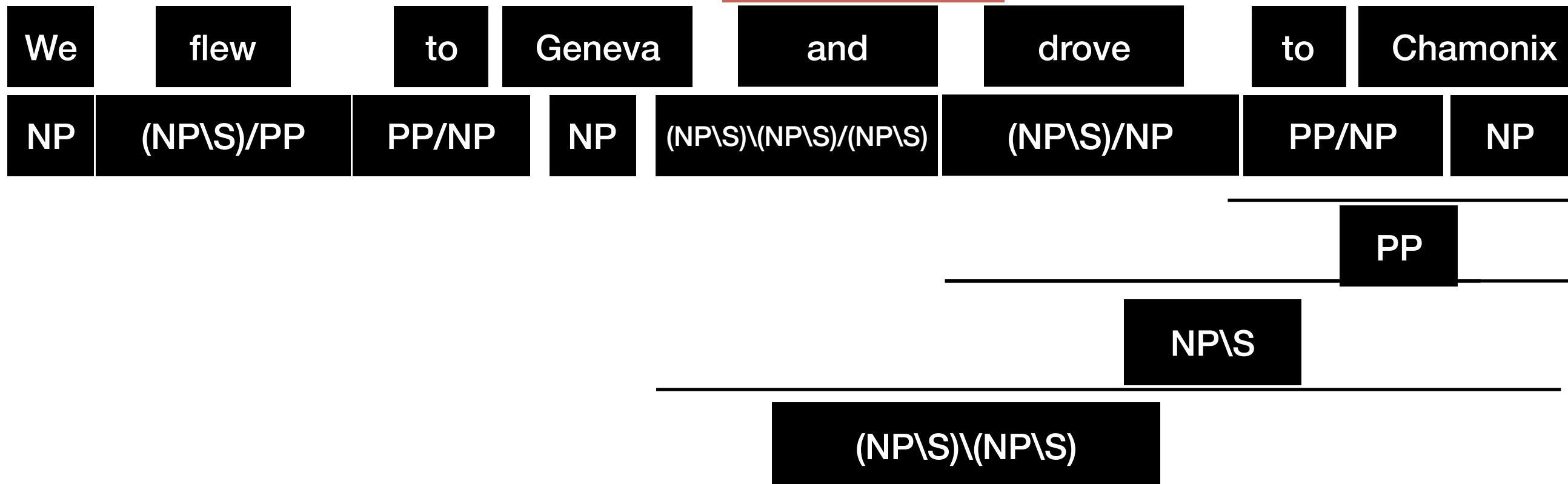
A more elaborate example:



Logical Grammars

A more elaborate example:

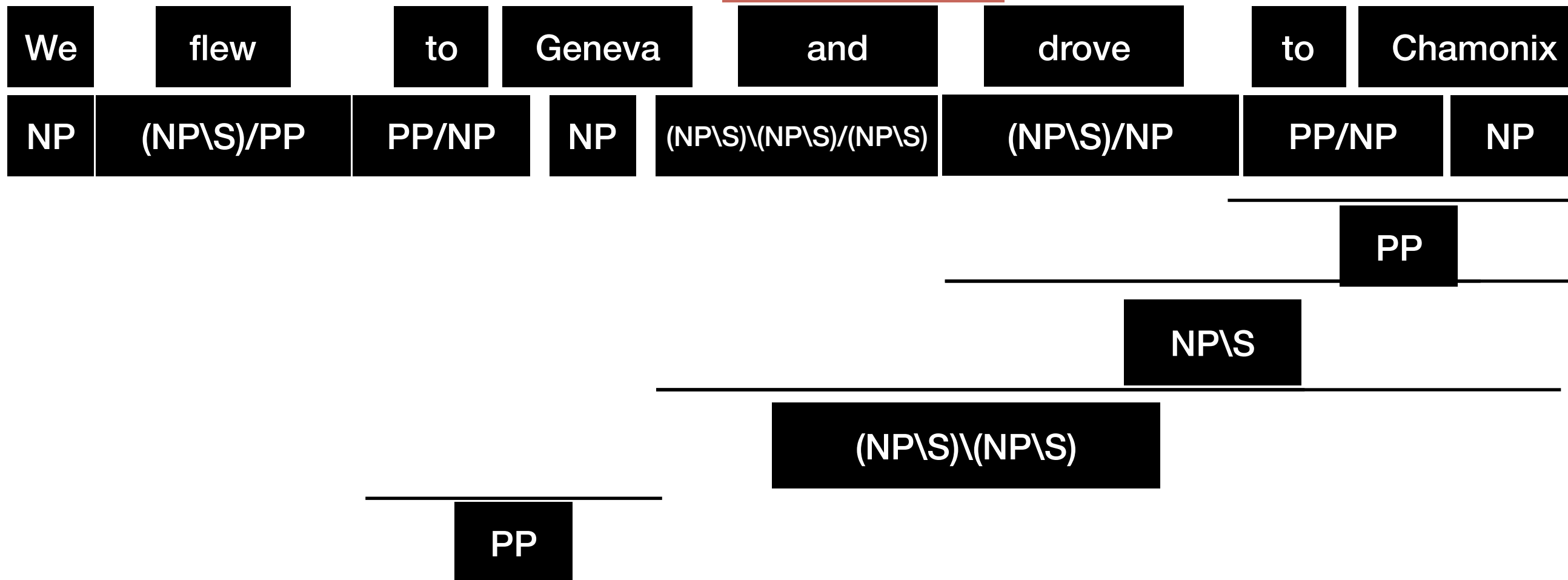
$X = NP \backslash S$



Logical Grammars

A more elaborate example:

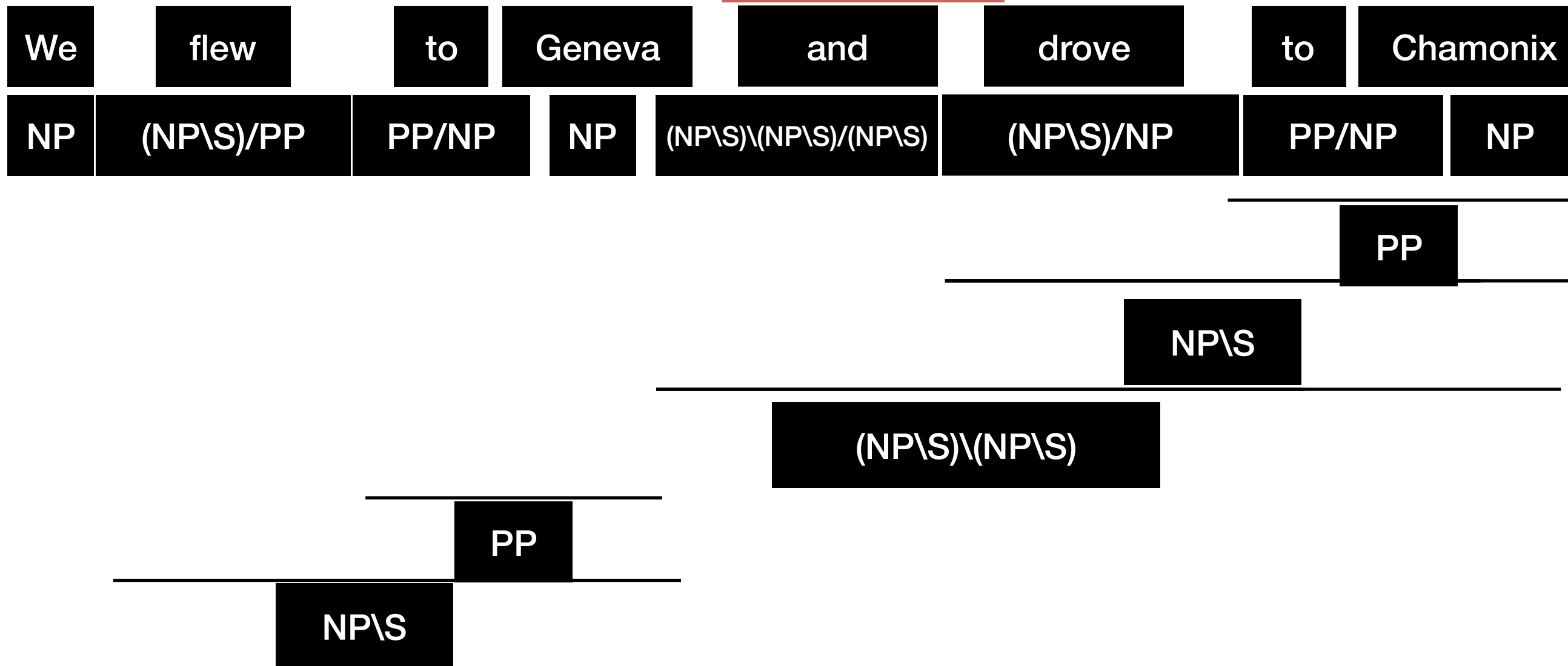
$X = NP \backslash S$



Logical Grammars

A more elaborate example:

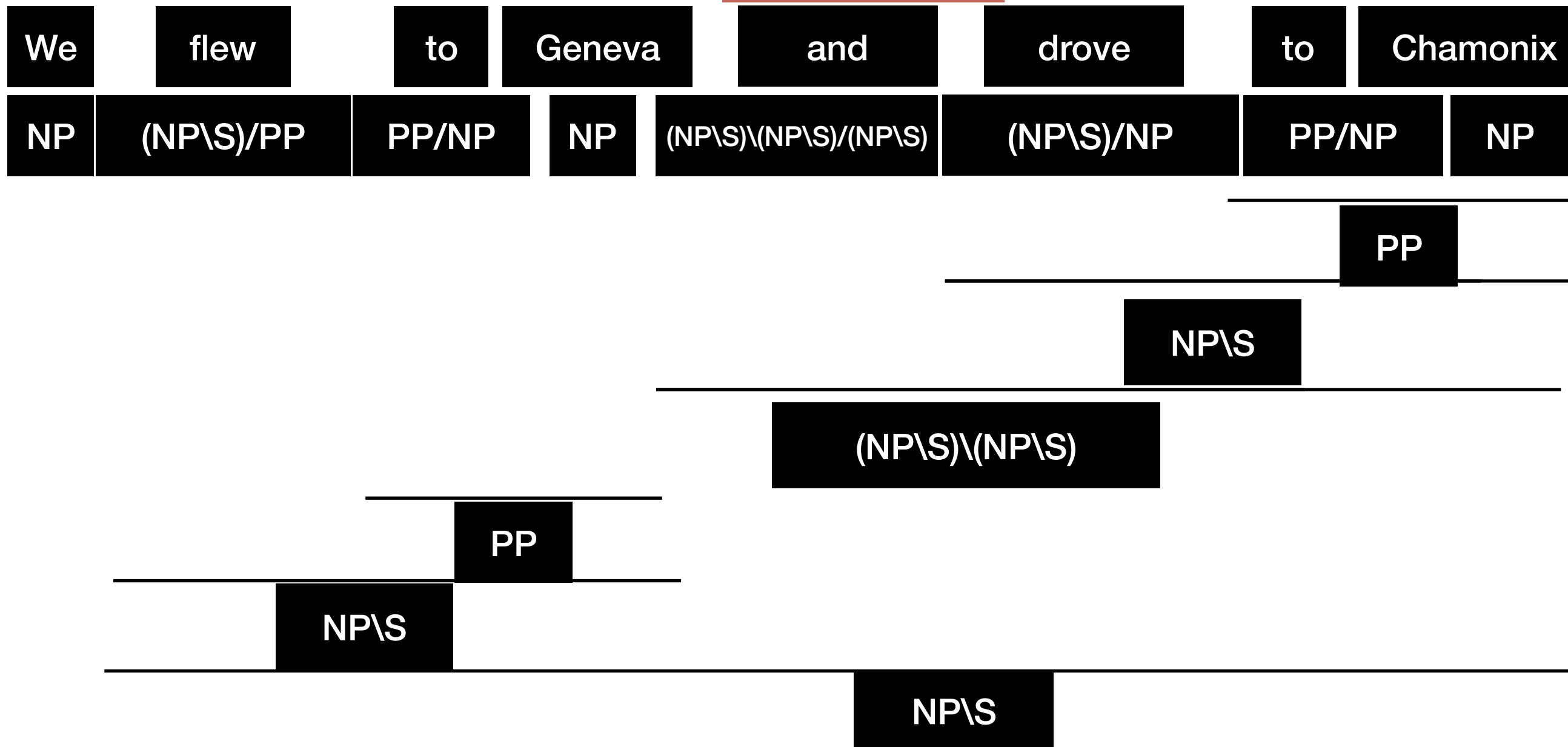
$X = NP \backslash S$



Logical Grammars

A more elaborate example:

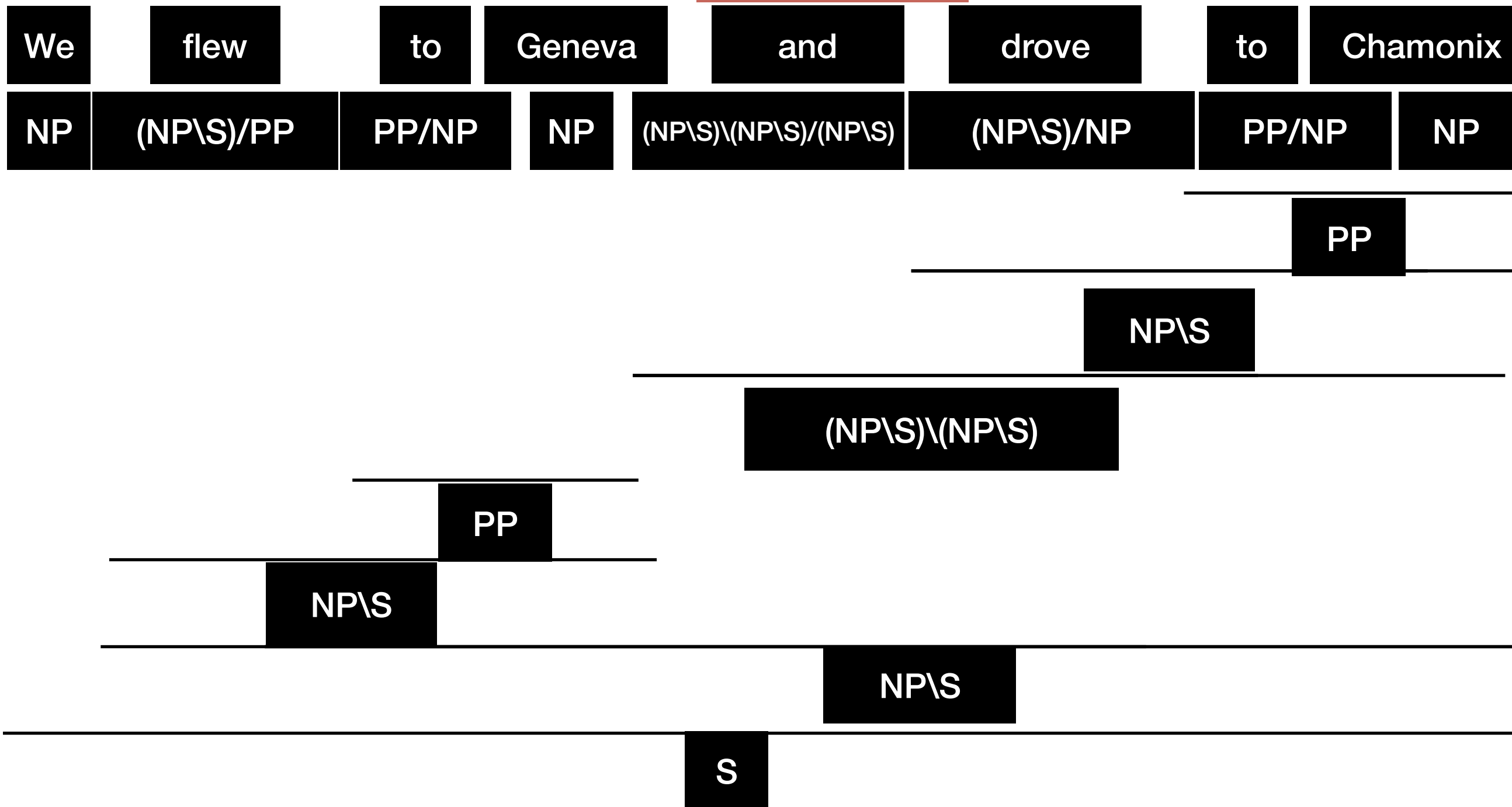
$X = NP \backslash S$



Logical Grammars

A more elaborate example:

$X = NP \backslash S$



Logical Grammars

- In 1958, the logician J. Lambek provided a logical reading of Bar-Hillel's directional types using the logical law of **Modus Ponens**:

$$A \rightarrow B, A \Rightarrow B$$

- He formalised Bar-Hillel's rules using proof rules and since then many people (Pollard and Sag 1987, Morrill 1994, Moortgat 1997, Steedman 2000) have used his setting to augment the AB calculus.
- Particularly well used is **Combinatorial Categorical Grammar (CCG)** (Steedman, 2000)

Logical Grammars

- In 1992 Pentus showed that the expressive power of Lambek's calculus is the same as Chomsky's context free grammars.
- He also presented a translation between the formalism of generative grammars and that of logical grammars.
- The two grammatical paradigms were finally united!

Summary

- Formal grammars define a mechanism for generating all strings of a formal language.
- The notion of constituency is key to understanding them.
- CFGs are common grammars which define context-free languages, but these don't cover all of a natural language like English.
- We need a way of parsing automatically (next week).
- For the next assessed **Lab 3 (released on Monday)**, you won't be programming, you'll be drawing (Syntax trees)!

Reading

- Jurafsky and Martin (3rd Ed) Chapter 12