

TAGs for iXML

dr. Hans-Dieter A. Hiep (hdh@drheap.nl)

1st International Symposium on Invisible XML

27 February 2026

Talk overview

1. Context-Free Grammars (CFGs)
2. Tree-Adjoining Grammars (TAGs)
3. TAGs for iXML

Context-Free Grammars (CFGs)

- ▶ Alphabet, strings
- ▶ Terminals (lowercase), non-terminals (uppercase)
- ▶ Start non-terminal, rules, derivations

Example (Rules)

$S \rightarrow P \mid Q \mid A$

$P \rightarrow aS$

$Q \rightarrow Sb$

$A \rightarrow \epsilon$

Example (String derivation)

$S \rightarrow P \rightarrow aS \rightarrow aQ \rightarrow aSb \rightarrow aAb \rightarrow ab$

- ▶ Basic algorithm: find and replace, string rewriting
- ▶ Type-2 in Chomsky hierarchy

Context-Free Grammars (CFGs)

- ▶ Alphabet, strings
- ▶ Terminals (lowercase), non-terminals (uppercase)
- ▶ Start non-terminal, rules, derivations

Example (Rules)

$S \rightarrow P \mid Q \mid A$

$P \rightarrow aS$

$Q \rightarrow Sb$

$A \rightarrow \epsilon$

Example (String derivation)

$S \rightarrow P \rightarrow aS \rightarrow aQ \rightarrow aSb \rightarrow aAb \rightarrow ab$

- ▶ Basic algorithm: find and replace, string rewriting
- ▶ Type-2 in Chomsky hierarchy

Context-Free Grammars (CFGs)

- ▶ Alphabet, strings
- ▶ Terminals (lowercase), non-terminals (uppercase)
- ▶ Start non-terminal, rules, derivations

Example (Rules)

$S \rightarrow P \mid Q \mid A$

$P \rightarrow aS$

$Q \rightarrow Sb$

$A \rightarrow \epsilon$

Example (String derivation)

$S \rightarrow P \rightarrow aS \rightarrow aQ \rightarrow aSb \rightarrow aAb \rightarrow ab$

- ▶ Basic algorithm: find and replace, string rewriting
- ▶ Type-2 in Chomsky hierarchy

Context-Free Grammars (CFGs)

- ▶ Alphabet, strings
- ▶ Terminals (lowercase), non-terminals (uppercase)
- ▶ Start non-terminal, rules, derivations

Example (Rules)

$S \rightarrow P \mid Q \mid A$

$P \rightarrow aS$

$Q \rightarrow Sb$

$A \rightarrow \epsilon$

Example (String derivation)

$S \rightarrow P \rightarrow aS \rightarrow aQ \rightarrow aSb \rightarrow aAb \rightarrow ab$

- ▶ Basic algorithm: find and replace, string rewriting
- ▶ Type-2 in Chomsky hierarchy

Context-Free Grammars (CFGs)

- ▶ Alphabet, strings
- ▶ Terminals (lowercase), non-terminals (uppercase)
- ▶ Start non-terminal, rules, derivations

Example (Rules)

$S \rightarrow P \mid Q \mid A$

$P \rightarrow aS$

$Q \rightarrow Sb$

$A \rightarrow \epsilon$

Example (String derivation)

$S \rightarrow P \rightarrow aS \rightarrow aQ \rightarrow aSb \rightarrow aAb \rightarrow ab$

- ▶ Basic algorithm: find and replace, string rewriting
- ▶ Type-2 in Chomsky hierarchy

Context-Free Grammars (CFGs)

- ▶ Alphabet, strings
- ▶ Terminals (lowercase), non-terminals (uppercase)
- ▶ Start non-terminal, rules, derivations

Example (Rules)

$S \rightarrow P \mid Q \mid A$

$P \rightarrow aS$

$Q \rightarrow Sb$

$A \rightarrow \epsilon$

Example (String derivation)

$S \rightarrow P \rightarrow aS \rightarrow aQ \rightarrow aSb \rightarrow aAb \rightarrow ab$

- ▶ Basic algorithm: find and replace, string rewriting
- ▶ Type-2 in Chomsky hierarchy

Context-Free Grammars (CFGs)

- ▶ Alphabet, strings
- ▶ Terminals (lowercase), non-terminals (uppercase)
- ▶ Start non-terminal, rules, derivations

Example (Rules)

$S \rightarrow P \mid Q \mid A$

$P \rightarrow aS$

$Q \rightarrow Sb$

$A \rightarrow \epsilon$

Example (String derivation)

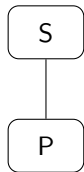
$S \rightarrow P \rightarrow aS \rightarrow aQ \rightarrow aSb \rightarrow aAb \rightarrow ab$

- ▶ Basic algorithm: find and replace, string rewriting
- ▶ Type-2 in Chomsky hierarchy

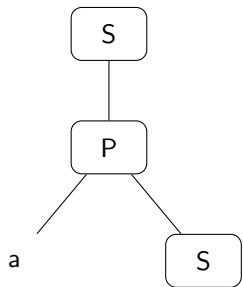
Parse trees



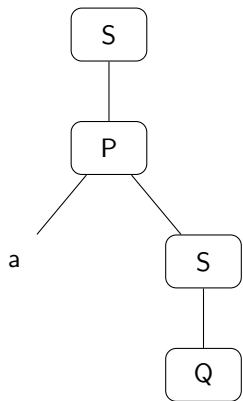
Parse trees



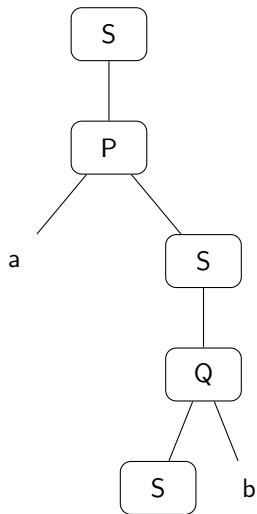
Parse trees



Parse trees



Parse trees



Marks


Marks are special instructions for terminals or non-terminals.

Parse tree operations (in front of terminal or non-terminal):



- ▶ + inserts the terminal but without matching any input
- ▶ – deletes non-terminal moving its children in its place
- ▶ @ flattens the tree below and makes the non-terminal an attribute of the node above

Tree-Adjoining Grammars

Import from computational linguistics


- ▶  **Tree-adjoining grammars**, Joshi & Schabes in *Handbook of formal languages: Volume 3 beyond words* (1997)

Many parsers available including CYK, Earley



- ▶  **Parsing beyond context-free grammars**, Kallmeyer (2010)
- ▶  **The equivalence of four extensions of context-free grammars**, Vijay-Shanker & Weir (1994)

Tree-Adjoining Grammars

Import from computational linguistics


- ▶  **Tree-adjoining grammars**, Joshi & Schabes in *Handbook of formal languages: Volume 3 beyond words* (1997)

Many parsers available including CYK, Earley



- ▶  **Parsing beyond context-free grammars**, Kallmeyer (2010)
- ▶  **The equivalence of four extensions of context-free grammars**, Vijay-Shanker & Weir (1994)

Tree-Adjoining Grammars

Import from computational linguistics

- ▶  **Tree-adjoining grammars**, Joshi & Schabes in *Handbook of formal languages: Volume 3 beyond words* (1997)

Many parsers available including CYK, Earley

- ▶  **Parsing beyond context-free grammars**, Kallmeyer (2010)
- ▶  **The equivalence of four extensions of context-free grammars**, Vijay-Shanker & Weir (1994)

Tree-Adjoining Grammars

- ▶ Terminals (lowercase), non-terminals (uppercase)
- ▶ Start non-terminal
- ▶ Set of **initial trees**:
 - ▶ All interior nodes are non-terminal
 - ▶ All leaves are either terminals, or marked for substitution (\downarrow)
- ▶ Set of **auxiliary trees**:
 - ▶ All interior nodes are non-terminal
 - ▶ All leaves are either terminals, or marked for substitution (\downarrow)
except one marked as foot ()*

Tree-Adjoining Grammars

- ▶ Terminals (lowercase), non-terminals (uppercase)
- ▶ Start non-terminal
- ▶ Set of **initial trees**:
 - ▶ All interior nodes are non-terminal
 - ▶ All leaves are either terminals, or marked for substitution (\downarrow)
- ▶ Set of **auxiliary trees**:
 - ▶ All interior nodes are non-terminal
 - ▶ All leaves are either terminals, or marked for substitution (\downarrow)
except one marked as foot ()*

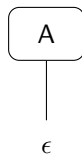
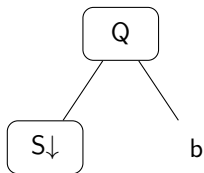
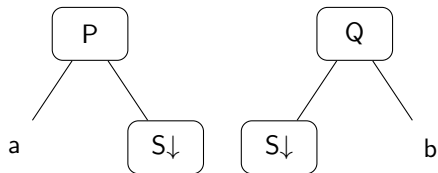
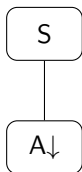
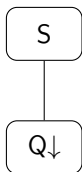
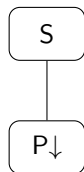
Tree-Adjoining Grammars

- ▶ Terminals (lowercase), non-terminals (uppercase)
- ▶ Start non-terminal
- ▶ Set of **initial trees**:
 - ▶ All interior nodes are non-terminal
 - ▶ All leaves are either terminals, or marked for substitution (\downarrow)
- ▶ Set of **auxiliary trees**:
 - ▶ All interior nodes are non-terminal
 - ▶ All leaves are either terminals, or marked for substitution (\downarrow)
except one marked as foot ()*

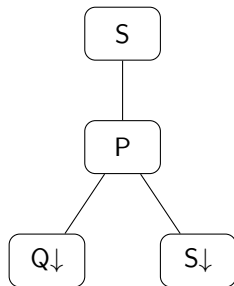
Tree-Adjoining Grammars

- ▶ Terminals (lowercase), non-terminals (uppercase)
- ▶ Start non-terminal
- ▶ Set of **initial trees**:
 - ▶ All interior nodes are non-terminal
 - ▶ All leaves are either terminals, or marked for substitution (\downarrow)
- ▶ Set of **auxiliary trees**:
 - ▶ All interior nodes are non-terminal
 - ▶ All leaves are either terminals, or marked for substitution (\downarrow)
except one marked as foot ()*

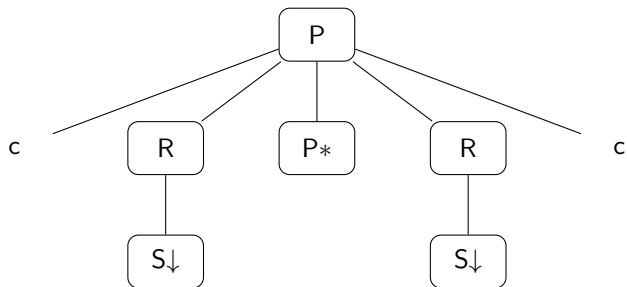
Expressivity: shallow initial trees



Example initial trees



Example auxiliary tree



Tree-Adjoining Grammars (cont.)

Rules: the initial trees and auxiliary trees

Start with singleton tree $S\downarrow$ of the start non-terminal S .

Tree derivation steps:

▶ **Substitution:**

replace non-terminal $X\downarrow$ marked for substitution by *initial tree* with same root X

▶ **Adjoining:**

1. take non-terminal X not marked for substitution
2. remember the subtree rooted in this X
3. replace this X with an auxiliary tree with root X
4. replace foot node X^* with the remembered tree

Tree-Adjoining Grammars (cont.)

Rules: the initial trees and auxiliary trees

Start with singleton tree $S\downarrow$ of the start non-terminal S .

Tree derivation steps:

▶ **Substitution:**

replace non-terminal $X\downarrow$ marked for substitution
by *initial tree* with same root X

▶ **Adjoining:**

1. take non-terminal X not marked for substitution
2. remember the subtree rooted in this X
3. replace this X with an auxiliary tree with root X
4. replace foot node X^* with the remembered tree

Tree-Adjoining Grammars (cont.)

Rules: the initial trees and auxiliary trees

Start with singleton tree $S\downarrow$ of the start non-terminal S .

Tree derivation steps:

▶ **Substitution:**

replace non-terminal $X\downarrow$ marked for substitution by *initial tree* with same root X

▶ **Adjoining:**

1. take non-terminal X not marked for substitution
2. remember the subtree rooted in this X
3. replace this X with an auxiliary tree with root X
4. replace foot node X^* with the remembered tree

Tree-Adjoining Grammars (cont.)

Rules: the initial trees and auxiliary trees

Start with singleton tree $S\downarrow$ of the start non-terminal S .

Tree derivation steps:

▶ **Substitution:**

replace non-terminal $X\downarrow$ marked for substitution
by *initial tree* with same root X

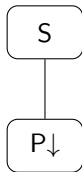
▶ **Adjoining:**

1. take non-terminal X not marked for substitution
2. remember the subtree rooted in this X
3. replace this X with an auxiliary tree with root X
4. replace foot node X^* with the remembered tree

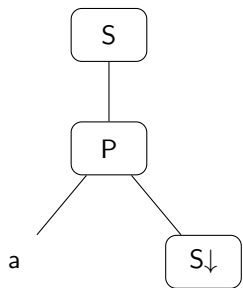
Tree derivation

S↓

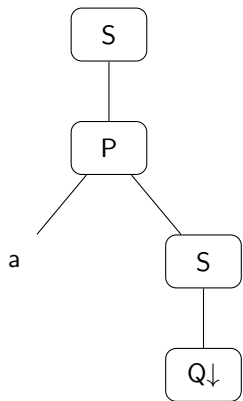
Tree derivation



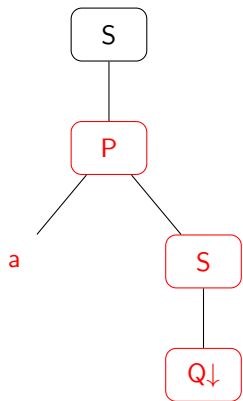
Tree derivation



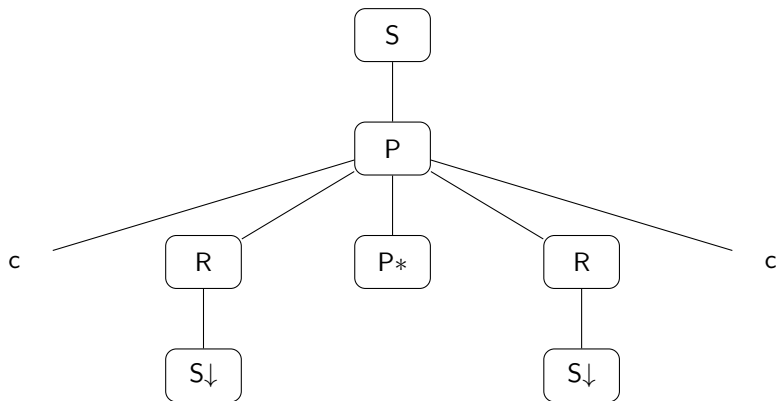
Tree derivation



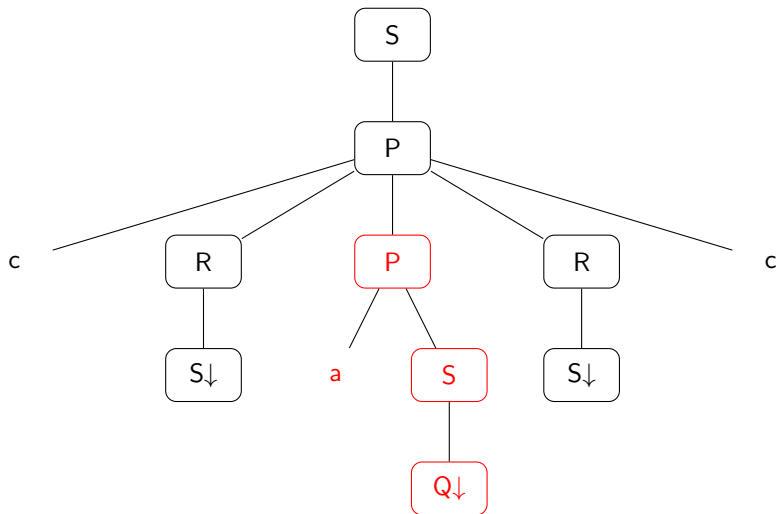
Tree derivation



Tree derivation



Tree derivation



More advanced topics

Seminal paper:

 **Tree adjunct grammars**, Joshi & Levy & Takahashi (1975)

Many more extensions, equivalent formalisms, parsers...

Lexicalized TAGs (for efficient parsing)

Adjoining constraints (attributes per non-terminal):

- ▶ Selective adjunction
- ▶ Null adjunction
- ▶ Obligatory adjunction

More advanced topics

Seminal paper:

 **Tree adjunct grammars**, Joshi & Levy & Takahashi (1975)

Many more extensions, equivalent formalisms, parsers...

Lexicalized TAGs (for efficient parsing)

Adjoining constraints (attributes per non-terminal):

- ▶ Selective adjunction
- ▶ Null adjunction
- ▶ Obligatory adjunction

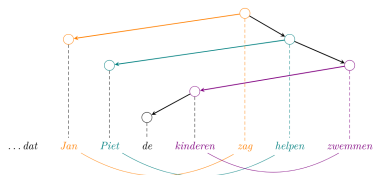
TAGs for iXML

Basic idea:

- ▶ Rules (initial trees, auxiliary trees) are defined in XML.
- ▶ Tree state is described by an XML tree.
- ▶ Perform substitution and adjoining operations on XML.

Why bother?

- ▶ Cross serial dependencies, e.g. structure of $A^n B C^n$



Downsides:

- ▶ Valid prefix property, i.e. to detect errors early, may be costly
- ▶ Not implemented

Rough example

```
<initial>
  <S> <P mark="sub" /> </S>
  <S> <Q mark="sub" /> </S>
  <S> <A mark="sub" /> </S>
  ...
</initial>

<auxiliary>
  <P>
    c
    <R><S mark="sub" /></R>
    <P mark="foot" />
    <R><S mark="sub" /></R>
    c
  </P>
</auxiliary>
```