Advances in Abstract Categorial Grammars

Language Theory and Linguistic Modeling

Lecture 5

Application of magic-sets rewriting
Prefix-correct Earley recognizer from Datalog program representing MCFG

Early parsing

- From CFGs to MCFGs
- Indirectly applies to string-generating second-order ACGs

Polynomial-time algorithm

```
Seminaive(P, D)
1   agenda[0] ← D
2   i ← 0
3   chart ← ∅
4   while agenda[i] ≠ ∅
5         do
6            chart ← chart ∪ agenda[i]
7            agenda[i + 1] ← Conseq(P, agenda[i], chart) − chart
8            i ← i + 1
9   return chart
```

holds facts with derivation tree of minimal height *i*

facts that immediately follow from one fact in *agenda*[*i*] plus some facts in *chart*

≈ well-formed substring table

Seminaive is a purely bottom-up tabular algorithm.
This algorithm works for Datalog programs in general.
The size of chart is $O(n^r)$ (r = maximal arity)
Running time $O(n^k)$ depends on the number of variables in rules.

S → NP VP
S → Aux NP VP
S → VP
NP → Det N1
NP → Name
NP → Pronoun
VP → V
VP → V NP
VP → V NP PP
VP → VP PP
N1 → N
N1 → Name N1
N1 → N1 PP
PP → P NP

Aux → does
Det → that | this | a
Name → Houston | TWA
Pronoun → I | she | me
V → book | include | prefer
N → book | flight | meal | money
P → from | to | on

An example of a context-free grammar.
Adapted from Jurafsky and Martin.

S(*i*, *k*) :- NP(*i*, *j*), VP(*j*, *k*).
S(*i*, *l*) :- Aux(*i*, *j*), NP(*j*, *k*), VP(*k*, *l*).
S(*i*, *j*) :- VP(*i*, *j*).
NP(*i*, *k*) :- Det(*i*, *j*), N1(*j*, *k*).
NP(*i*, *j*) :- Name(*i*, *j*).
NP(*i*, *j*) :- Pronoun(*i*, *j*).
VP(*i*, *j*) :- V(*i*, *j*).
VP(*i*, *k*) :- V(*i*, *j*), NP(*j*, *k*).
VP(*i*, *l*) :- V(*i*, *j*), NP(*j*, *k*), PP(*k*, *l*).
VP(*i*, *k*) :- VP(*i*, *j*), PP(*j*, *k*).
N1(*i*, *j*) :- N(*i*, *j*).
N1(*i*, *k*) :- N1(*i*, *j*), N(*j*, *k*).
N1(*i*, *k*) :- N1(*i*, *j*), PP(*j*, *k*).
PP(*i*, *k*) :- P(*i*, *j*), NP(*j*, *k*).

Aux(*i*, *j*) :- does(*i*, *j*).
Det(*i*, *j*) :- that(*i*, *j*).
Det(*i*, *j*) :- this(*i*, *j*).
Det(*i*, *j*) :- a(*i*, *j*).
Name(*i*, *j*) :- Houston(*i*, *j*).
Name(*i*, *j*) :- TWA(*i*, *j*).
Pronoun(*i*, *j*) :- I(*i*, *j*).
Pronoun(*i*, *j*) :- she(*i*, *j*).
Pronoun(*i*, *j*) :- me(*i*, *j*).
V(*i*, *j*) :- book(*i*, *j*).
V(*i*, *j*) :- include(*i*, *j*).
V(*i*, *j*) :- prefer(*i*, *j*).
N(*i*, *j*) :- book(*i*, *j*).
N(*i*, *j*) :- flight(*i*, *j*).
N(*i*, *j*) :- meal(*i*, *j*).
N(*i*, *j*) :- money(*i*, *j*).
P(*i*, *j*) :- from(*i*, *j*).
P(*i*, *j*) :- to(*i*, *j*).
P(*i*, *j*) :- on(*i*, *j*).

Datalog program representing CFG.
Running time of Seminaive is $O(N^4)$

D = { book(0, 1), the(1, 2), flight(2, 3), from(3, 4), Houston(4, 5) }.

agenda[0] = { book(0, 1), the(1, 2), flight(2, 3), from(3, 4), Houston(4, 5) }.
agenda[1] = { N(0, 1), V(0, 1), Det(1, 2), N(2, 3), P(3, 4), Name(4, 5) }
agenda[2] = { N1(0, 1), VP(0, 1), N1(2, 3), NP(4, 5) }
agenda[3] = { NP(1, 3), PP(3, 5) }
agenda[4] = { VP(0, 3), VP(0, 5), N1(2, 5) }
agenda[5] = { S(0, 3), S(0, 5), NP(1, 5) }
agenda[6] = ∅

chart = *agenda*[0] ∪ ··· ∪ *agenda*[6]

The run of Seminaive on “book the flight from Houston”

Outputing shared forest

Seminaive_parse(**P**, *D*)

```
 1   agenda[0] ← D
 2   i ← 0
 3   chart ← ∅
 4   parse ← ∅
 5   while agenda[i] ≠ ∅
 6         do
 7               chart ← chart ∪ agenda[i]
 8               agenda[i + 1] ← Conseq(P, agenda[i], chart) − chart
 9               parse ← parse ∪ Inst(P, agenda[i], chart)
10               i ← i + 1
11   return parse
```

holds instances of rules that can be used in derivation trees

facts that immediately follow from one fact in *agenda*[*i*] plus some facts in *chart*

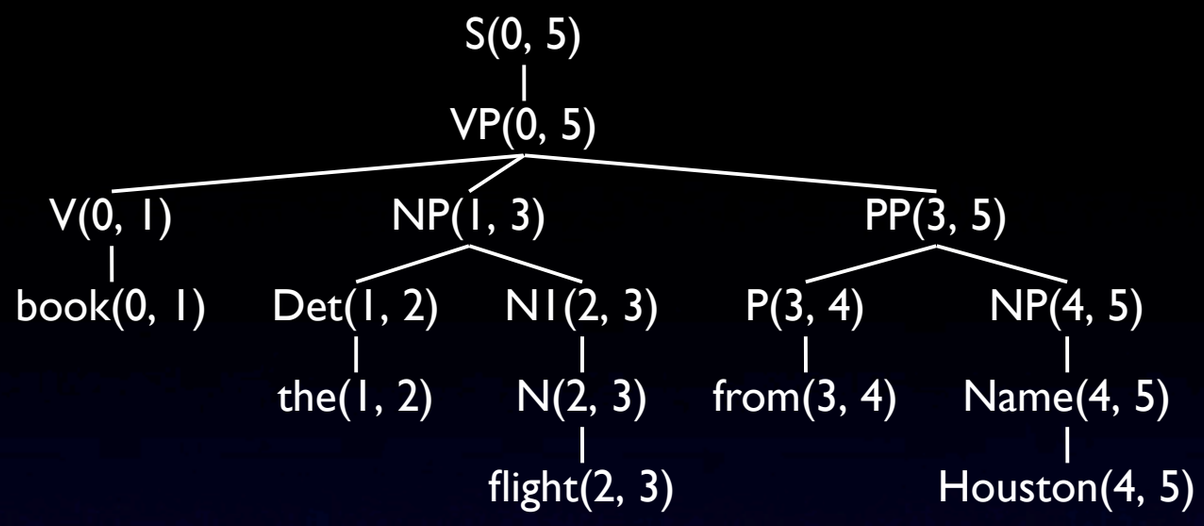
shared parse forest

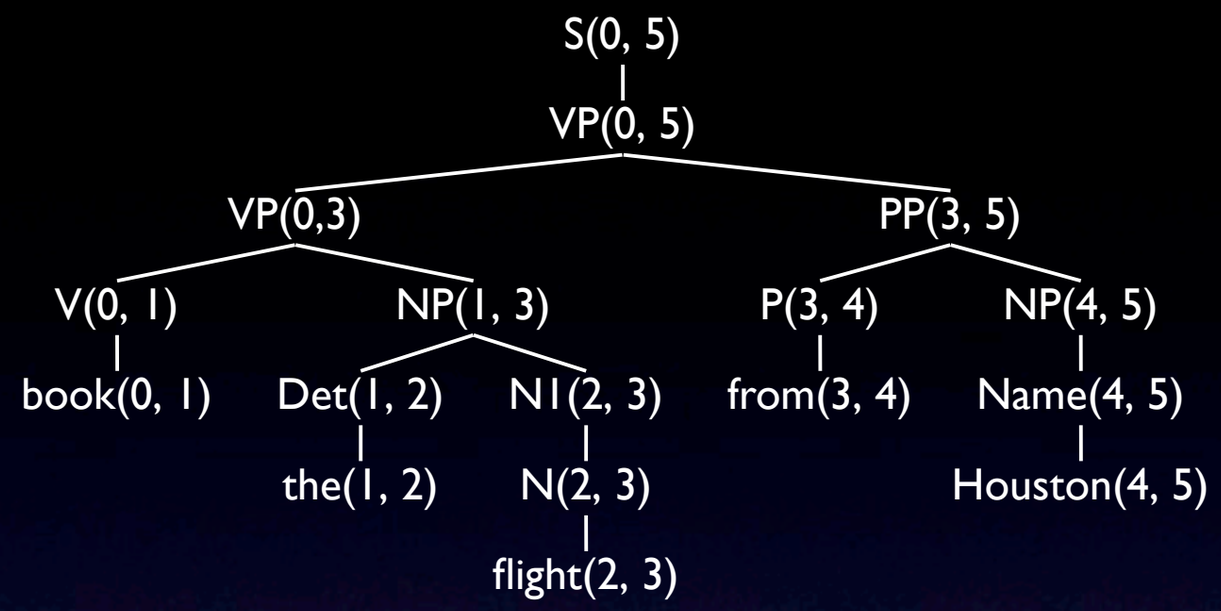
instances of rules with right-hand side consisting of one fact in *agenda*[i] and some facts in *chart*

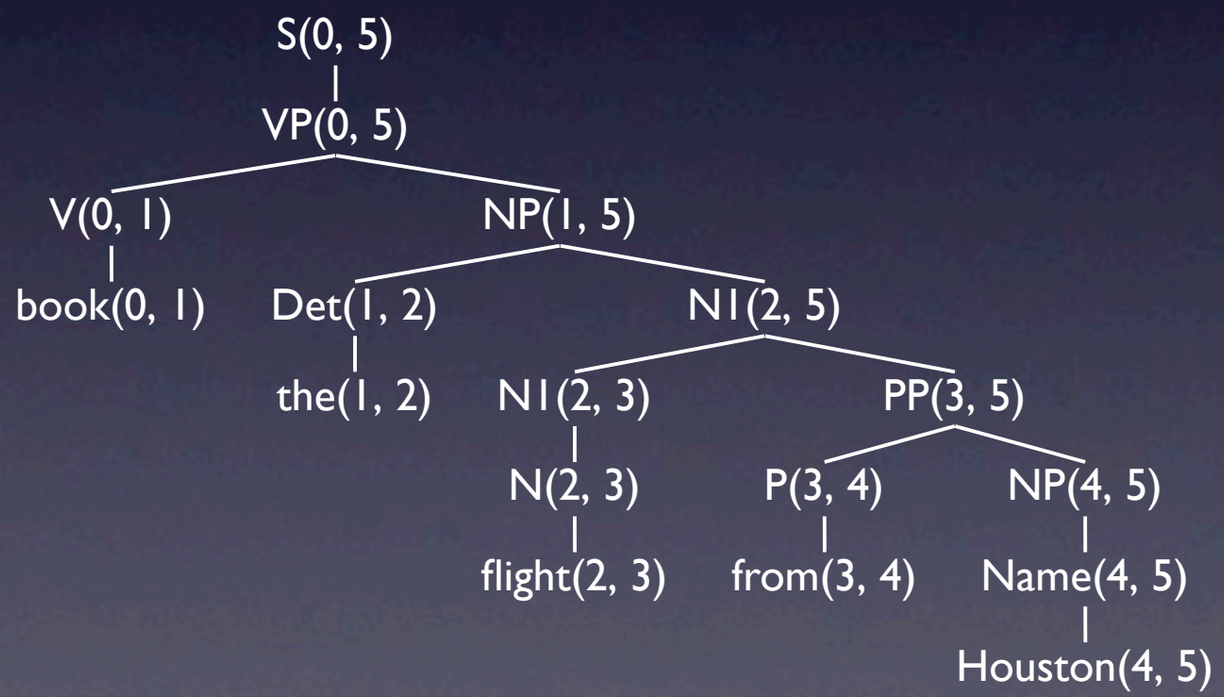
Parsing algorithms are often not explicitly stated in textbooks.
Lines 8 and 9 should be performed simultaneously.

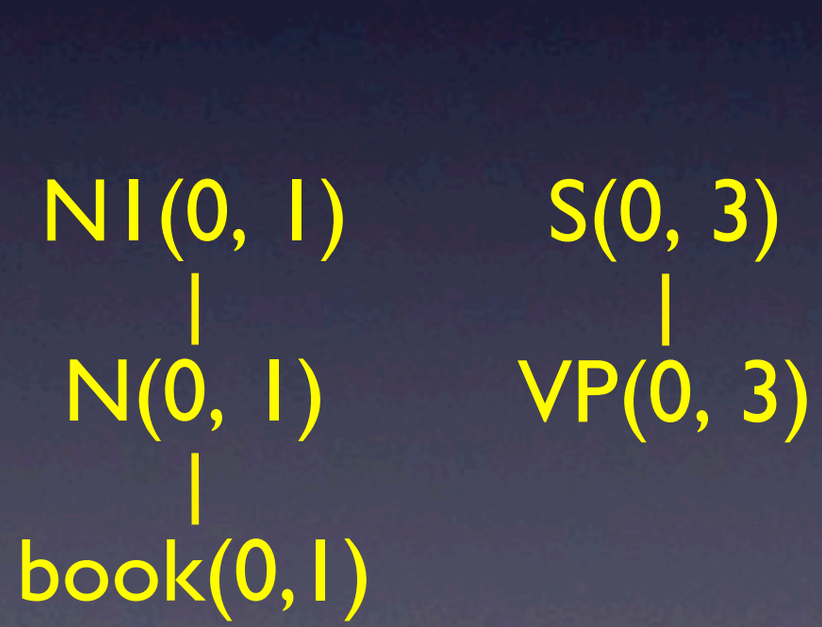
```
parse = { book(0, 1).  the(1, 2).  flight(2, 3).  from(3, 4).  Houston(4, 5).
          N(0, 1) :- book(0, 1).
          V(0, 1) :- book(0, 1).
          Det(1, 2) :- the(1, 2).
          N(2, 3) :- flight(2, 3).
          P(3, 4) :- from(3, 4).
          Name(4, 5) :- Houston(4, 5).
          N1(0, 1) :- N(0, 1).
          VP(0, 1) :- V(0, 1).
          N1(2, 3) :- N(2, 3).
          NP(4, 5) :- Name(4, 5).
          NP(1, 3) :- Det(1, 2), N1(2, 3).
          PP(3, 5) :- P(3, 4), NP(4, 5).
          VP(0, 3) :- V(0, 1), NP(1, 3).
          VP(0, 5) :- V(0, 1), NP(1, 3), PP(3, 5).
          N1(2, 5) :- N1(2, 3), PP(3, 5).
          S(0, 3) :- VP(0, 3).
          S(0, 5) :-VP(0, 5).
          VP(0, 5) :-VP(0, 3), PP(3, 5).
          NP(1, 5) :- Det(1, 2), N1(2, 5).
          VP(0, 5) :- V(0, 1), NP(1, 5).
        }
```

Output parse forest.
Can be thought of as a CFG for a singleton language.





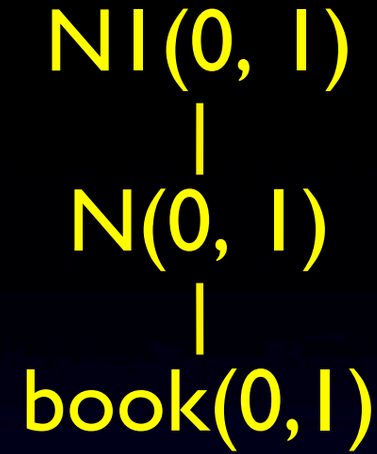


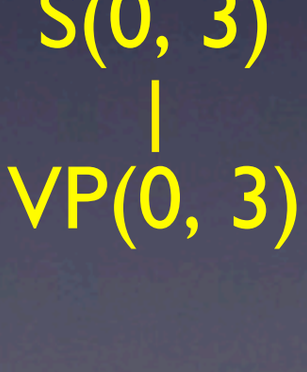


Some rule instances found by the algorithm cannot be used in any derivation trees.

parse = { book(0, 1). the(1, 2). flight(2, 3). from(3, 4). Houston(4, 5).

N(0, 1) :- book(0, 1).
V(0, 1) :- book(0, 1).
Det(1, 2) :- the(1, 2).
N(2, 3) :- flight(2, 3).
P(3, 4) :- from(3, 4).
Name(4, 5) :- Houston(4, 5).
N1(0, 1) :- N(0, 1).
VP(0, 1) :- V(0, 1).
N1(2, 3) :- N(2, 3).
NP(4, 5) :- Name(4, 5).
NP(1, 3) :- Det(1, 2), N1(2, 3).
PP(3, 5) :- P(3, 4), NP(4, 5).
VP(0, 3) :- V(0, 1), NP(1, 3).
VP(0, 5) :- V(0, 1), NP(1, 3), PP(3, 5).
N1(2, 5) :- N1(2, 3), PP(3, 5).
S(0, 3) :- VP(0, 3).
S(0, 5) :-VP(0, 5).
VP(0, 5) :-VP(0, 3), PP(3, 5).
NP(1, 5) :- Det(1, 2), N1(2, 5).
VP(0, 5) :- V(0, 1), NP(1, 5).
}





This parse forest is not “reduced”.

D = { book(0, 1), the(1, 2), flight(2, 3), from(3, 4), Houston(4, 5) }.

agenda[0] = { book(0, 1), the(1, 2), flight(2, 3), from(3, 4), Houston(4, 5) }.
agenda[1] = { N(0, 1), V(0, 1), Det(1, 2), N(2, 3), P(3, 4), Name(4, 5) }
agenda[2] = { N1(0, 1), VP(0, 1), N1(2, 3), NP(4, 5) }
agenda[3] = { NP(1, 3), PP(3, 5) }
agenda[4] = { VP(0, 3), VP(0, 5), N1(2, 5) }
agenda[5] = { S(0, 3), S(0, 5), VP(0, 5), NP(1, 5) }
agenda[6] = ∅

chart = *agenda*[0] ∪ ··· ∪ *agenda*[6]

"Weak parse" as opposed to "parse" (Ruzzo).

Earley parsing

- $O(n^3)$ running time for arbitrary CFG
- Partial reduction of parse forest by top-down filtering
- Correct prefix property

The first two goals independent of each other in principle.
Top-down filtering plus appropriate control achieves partial reduction.

Correct prefix property

- Process input from left to right
- Reject the input as soon as the portion of the input that has been processed so far is not a correct prefix (i.e., cannot be extended to an element of the language)

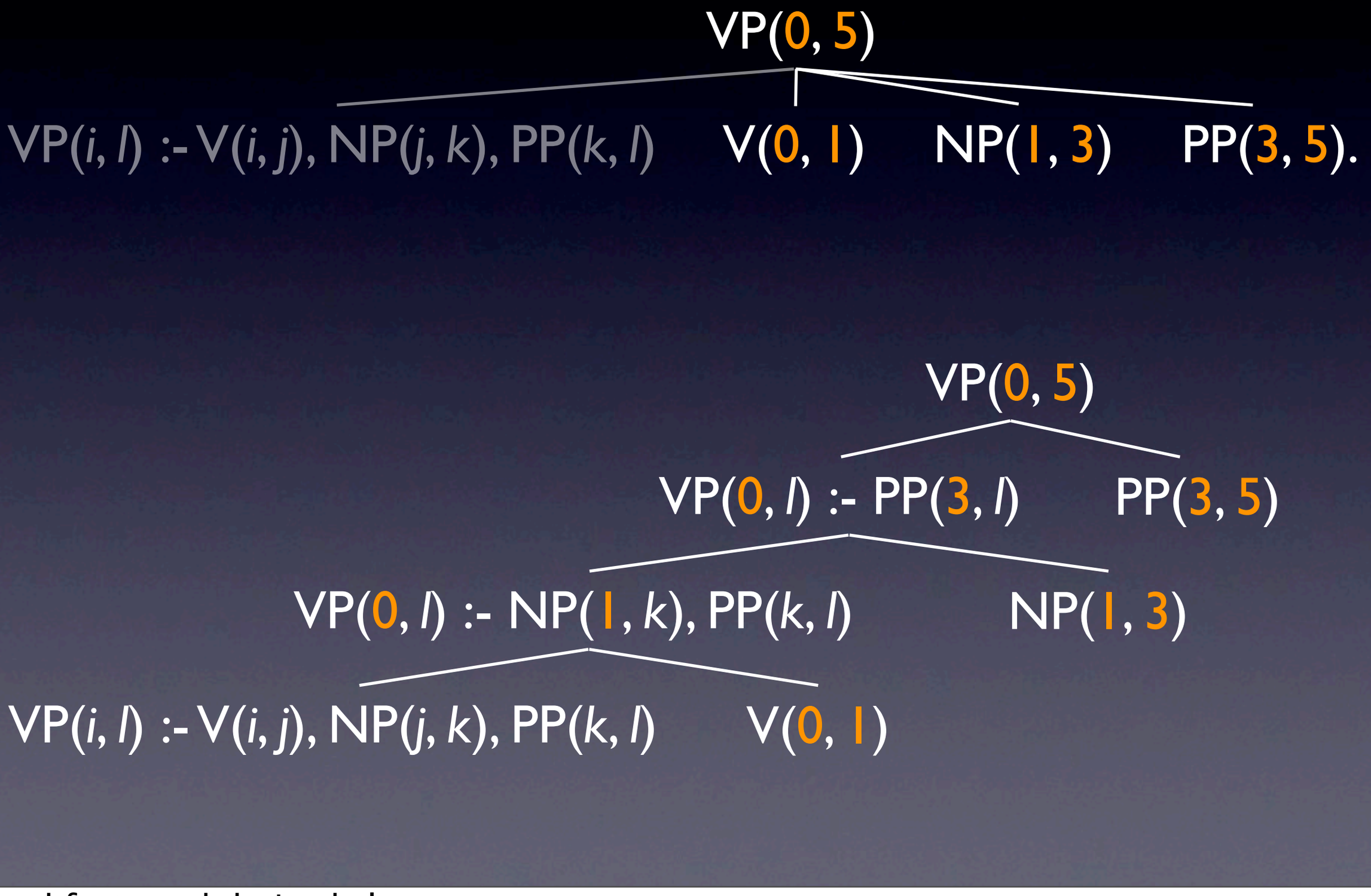
Term comes from stack-based parsing.

Earley parsing

- Binarization of rules
- Additional subgoals

Binarization improves running time.
Additional subgoals achieve top-down filtering.
Binarization helps for top-down filtering as well.
The way the top-down filtering is done achieves prefix-correctness.

Binarization of rules

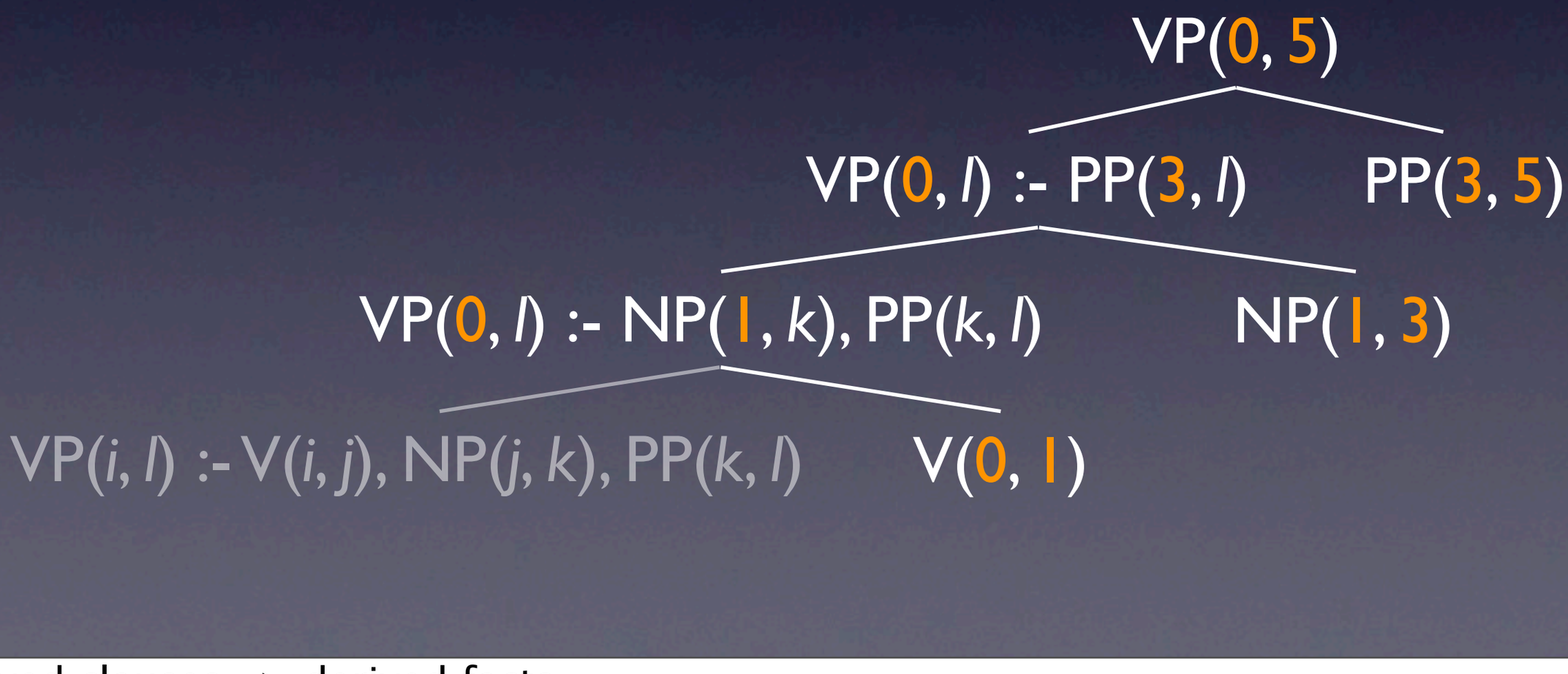


Derived facts and derived clauses.

Dotted rules

VP → V NP PP

VP → V • NP PP　　VP → V NP • PP



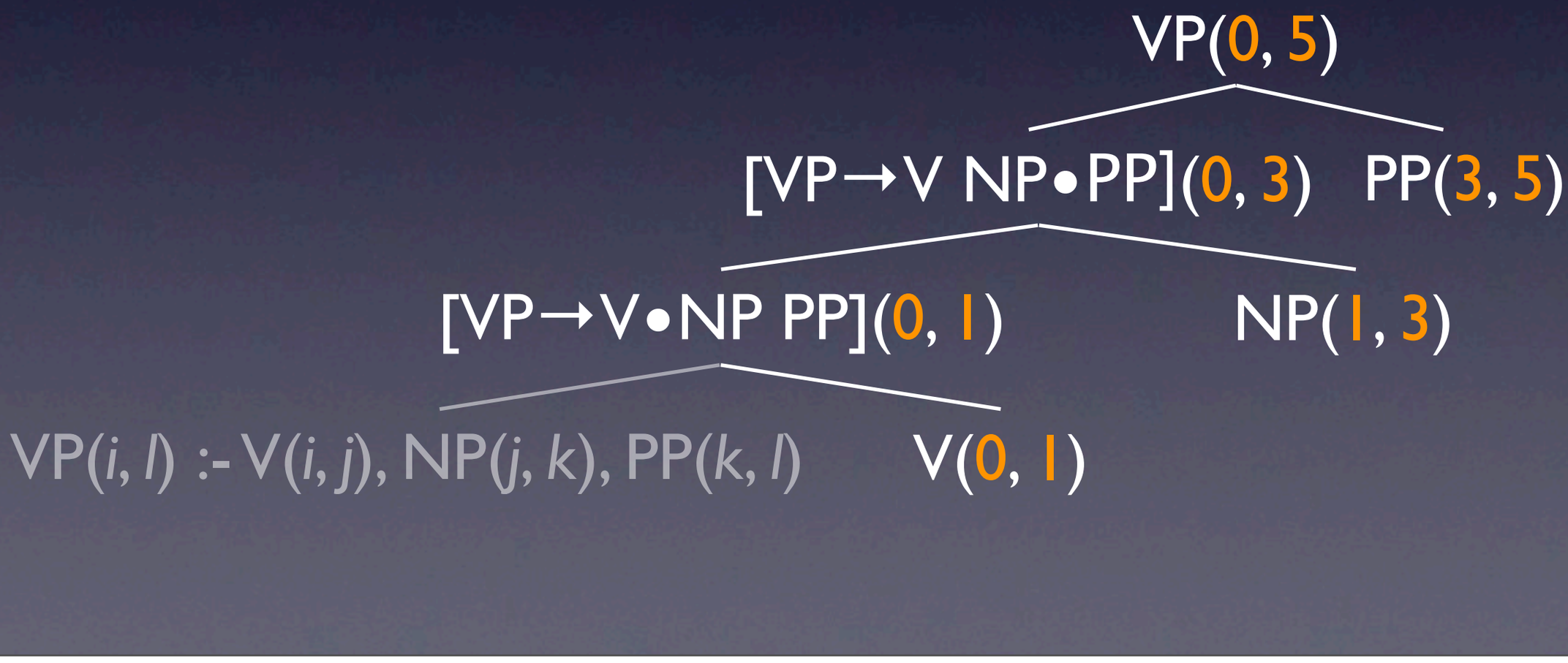
Derived clauses -> derived facts
Dotted rules as new predicate names.

Dotted rules

[VP→V•NP PP](*i*, *j*) :- V(*i*,*j*).

[VP→V NP•PP](*i*, *k*) :- [VP→V•NP PP](*i*, *j*), NP(*j*, *k*).

VP(*i*, *l*) :- [VP→V NP•PP](*i*, *k*), PP(*k*, *l*).

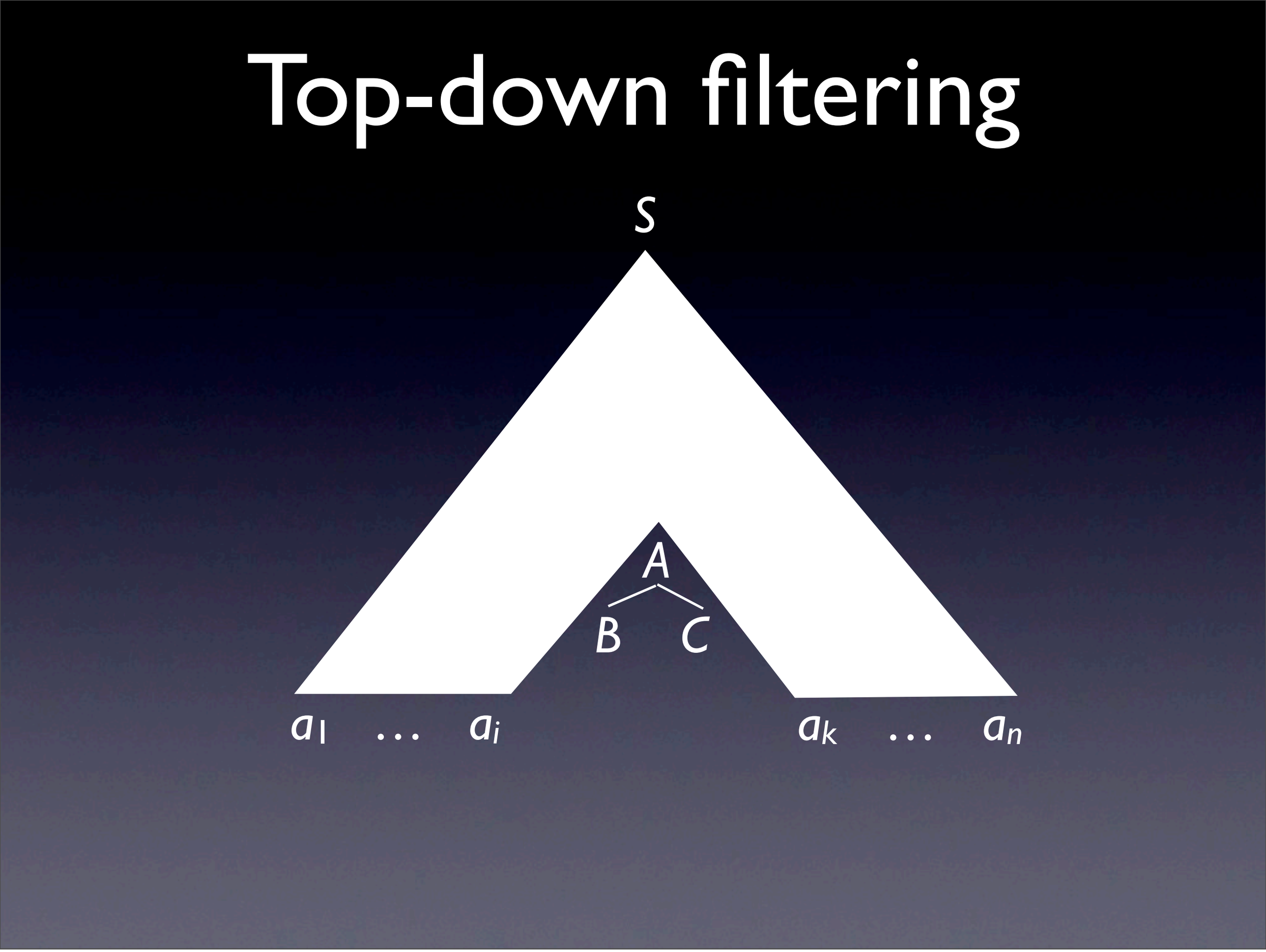


[S→NP•VP](*i*, *j*) :- NP(*i*, *j*).
S(*i*, *k*) :- [S→NP•VP](*i*, *j*), VP(*j*, *k*).
[S→Aux•NP VP](*i*, *j*) :- Aux(*i*, *j*).
[S→Aux NP•VP](*i*, *k*) :- [S→Aux•NP VP](*i*, *j*), Aux(*j*, *k*).
S(*i*, *l*) :- [S→Aux NP•VP](*i*, *k*), VP(*k*, *l*).
S(*i*, *j*) :- VP(*i*, *j*).
[NP→Det•N1](*i*, *j*) :- Det(*i*, *j*).
NP(*i*, *k*) :- [NP→Det•N1](*i*, *j*), N1(j, k).
NP(*i*, *j*) :- Name(*i*, *j*).
NP(*i*, *j*) :- Pronoun(*i*, *j*).
VP(*i*, *j*) :- V(*i*, *j*).
[VP→V•NP](*i*, *j*) :- V(*i*, *j*).
VP(*i*, *k*) :- [VP→V•NP](*i*, *j*), NP(*j*, *k*).
[VP→V•NP PP](*i*, *j*) :- V(*i*, *j*).
[VP→V NP•PP](*i*, *k*) :- [VP→V•NP PP](*i*, *j*), NP(*j*, *k*).
VP(*i*, *l*) :- [VP→V NP•PP](*i*, *k*), PP(*k*, *l*).

⋮

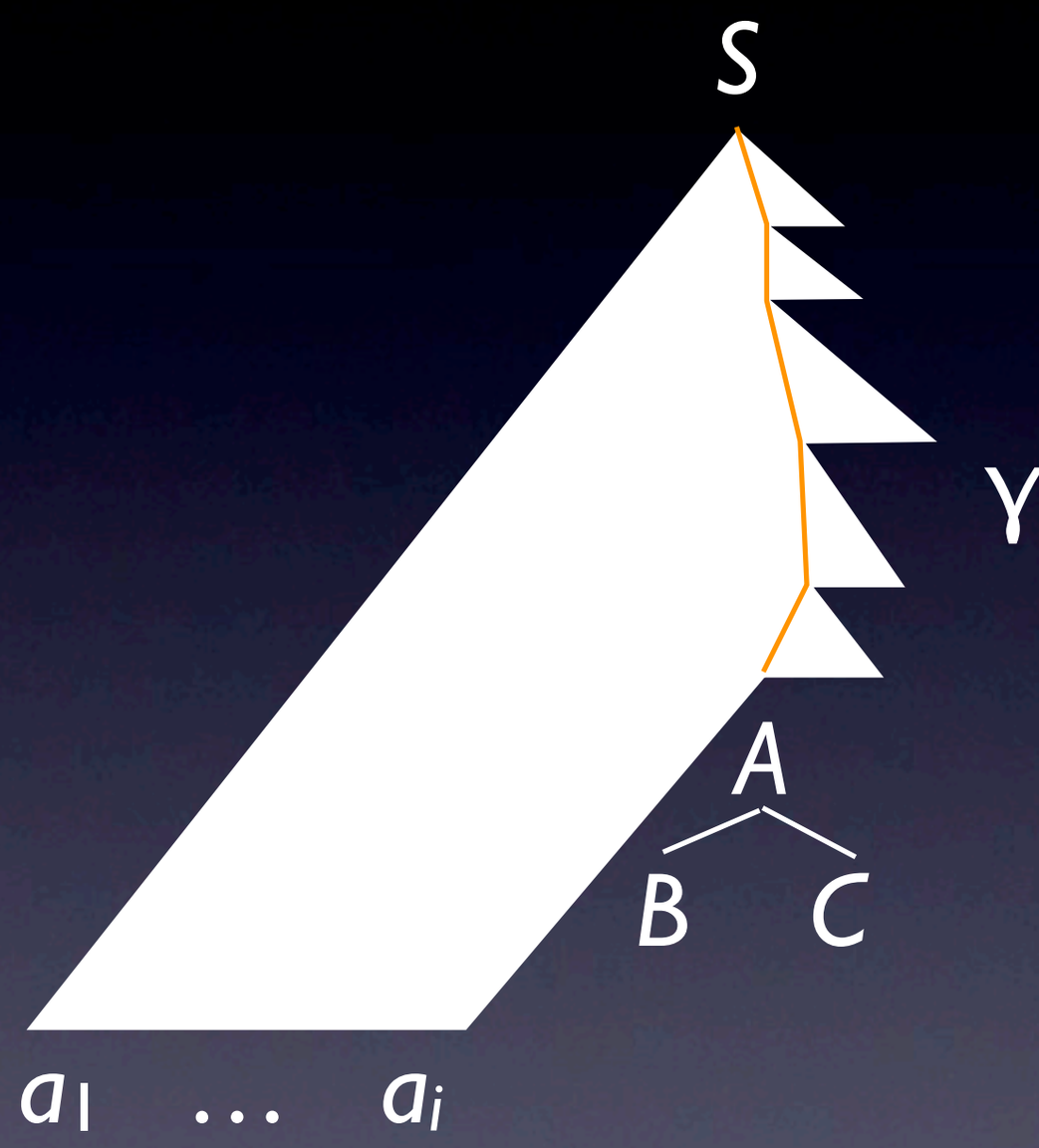
Binarized program: each rule contains at most 3 variables.
Running time is $O(N^3)$
“Bottom-up chart” recognizer

Top-down filtering



The existence of the white region needs to be checked in order to produce a complete parse/ reduced parse forest.
But this is too much and inconsistent with correct prefix property.

Top-down filtering



$S \Rightarrow^* a_1 \ldots a_i \, A \, \gamma$ leftmost derivation

Require there to be a path from the root and the part to the left of the path to be realized. This has the dual effect of partial reduction of parse forest and correct prefix property.

Top-down prediction

$m\_VP(i) \Leftrightarrow S \Rightarrow^* a_1 \ldots a_i \; VP \; \gamma$

"VP is predicted to start at *i*"

[VP→V•NP PP](*i*, *j*) :- *m*_VP(*i*), V(*i*,*j*).

[VP→V NP•PP](*i*, *k*) :- [VP→V•NP PP](*i*, *j*), NP(*j*, *k*).

VP(*i*, *l*) :- [VP→V NP•PP](*i*, *k*), PP(*k*, *l*).

*m*_S(0). seed

S(*i*, *j*) :- VP(*i*, *j*). *m*_VP(*i*) :- m_S(*i*).

*m*_V(*i*) :- *m*_VP(*i*).

*m*_NP(*j*) :- [VP→V•NP PP](*i*, *j*).

*m*_PP(*k*) :- [VP→V NP•PP](*i*, *k*).

Earley deduction system

$m\_S(0)$. INITIALIZE

$m\_B(j)$:- $[A \rightarrow \alpha \bullet B \beta](i, j)$.

$m\_B(i)$:- $m\_A(i)$. $(A \rightarrow B\beta)$ PREDICT

$[A \rightarrow B \bullet \beta](i, j)$:- $m\_A(i)$, $B(i, j)$.

$[A \rightarrow \alpha B \bullet \beta](i, k)$:- $[A \rightarrow \alpha \bullet B\beta](i, j)$, $B(j, k)$.

$A(i, k)$:- $[A \rightarrow \alpha \bullet B](i, j)$, $B(j, k)$. COMPLETE

$[A \rightarrow b \bullet \beta](i, j)$:- $m\_A(i)$, $b(i, j)$.

$[A \rightarrow \alpha b \bullet \beta](i, k)$:- $[A \rightarrow \alpha \bullet b\beta](i, j)$, $b(j, k)$.

$A(i, k)$:- $[A \rightarrow \alpha \bullet b](i, j)$, $b(j, k)$. SCAN

A minor variation, expressed in the style of Datalog.
“m_” comes from magic-sets rewriting.

*m*_NP(*i*) :- *m*_S(*i*).
[S→NP•VP](*i*, *j*) :- *m*_S(*i*), NP(*i*, *j*).
*m*_VP(*j*) :- [S→NP•VP](*i*, *j*).
S(*i*, *k*) :- [S→NP•VP](*i*, *j*), VP(*j*, *k*).
*m*_Aux(*i*) :- *m*_S(*i*).
[S→Aux•NP VP](*i*, *j*) :- *m*_S(*i*), Aux(*i*, *j*).
*m*_NP(*j*) :- [S→Aux•NP VP](*i*, *j*).
[S→Aux NP•VP](*i*, *k*) :- [S→Aux•NP VP](*i*, *j*), Aux(*j*, *k*).
*m*_VP(*k*) :- [S→Aux NP•VP](*i*, *k*).
S(*i*, *l*) :- [S→Aux NP•VP](*i*, *k*), VP(*k*, *l*).
*m*_VP(*i*) :- *m*_S(*i*).
S(*i*, *j*) :- *m*_S(*i*), VP(*i*, *j*).
*m*_Det(*i*) :- *m*_NP(*i*).
[NP→Det•N1](*i*, *j*) :- *m*_NP(*i*), Det(*i*, *j*).
*m*_N1(*j*) :- [NP→Det•N1](*i*, *j*).
NP(*i*, *k*) :- [NP→Det•N1](*i*, *j*), N1(j, k).

⋮

Earley deduction system as Datalog program.

Prefix-correct control algorithm

Chart_recognize($\mathbf{P}$, D)

$agenda \leftarrow D$
$i \leftarrow 0$
$chart \leftarrow \varnothing$
while $agenda \neq \varnothing$
do $trigger \leftarrow \mathrm{Pop}(agenda)$
$chart \leftarrow chart \cup \{trigger\}$
$new \leftarrow \mathrm{Conseq}(\mathbf{P}, \{trigger\}, chart) - chart$
if $trigger = a_j(j-1, j)$ and $new = \varnothing$
then reject
foreach $item \in new$
do Push($agenda$, $item$)
return $chart$

Agenda is now first-in first-out.
Push and pop one item at a time.

Earley = magic-sets rewriting

$m\_S(0)$.

$m\_B(j)$:- $sup_{r,n}(i, j)$. $(r = A \rightarrow \alpha B \beta, n = |\alpha|)$

$m\_B(i)$:- $m\_A(i)$. $(A \rightarrow B\beta)$

$sup_{r,1}(i, j)$:- $m\_A(i), B(i, j)$. $(r = A \rightarrow B\beta)$

$sup_{r,n+1}(i, k)$:- $sup_{r,n}(i, j), B(j, k)$. $(r = A \rightarrow \alpha B \beta, n = |\alpha|)$

$A(i, k)$:- $sup_{r,n}(i, j), B(j, k)$. $(r = A \rightarrow \alpha \bullet B, n = |\alpha|)$

$sup_{r,1}(i, j)$:- $m\_A(i), b(i, j)$. $(r = A \rightarrow b\beta)$

$sup_{r,n+1}(i, k)$:- $sup_{r,n}(i, j), b(j, k)$. $(r = A \rightarrow \alpha b \beta, n \equiv |\alpha|)$

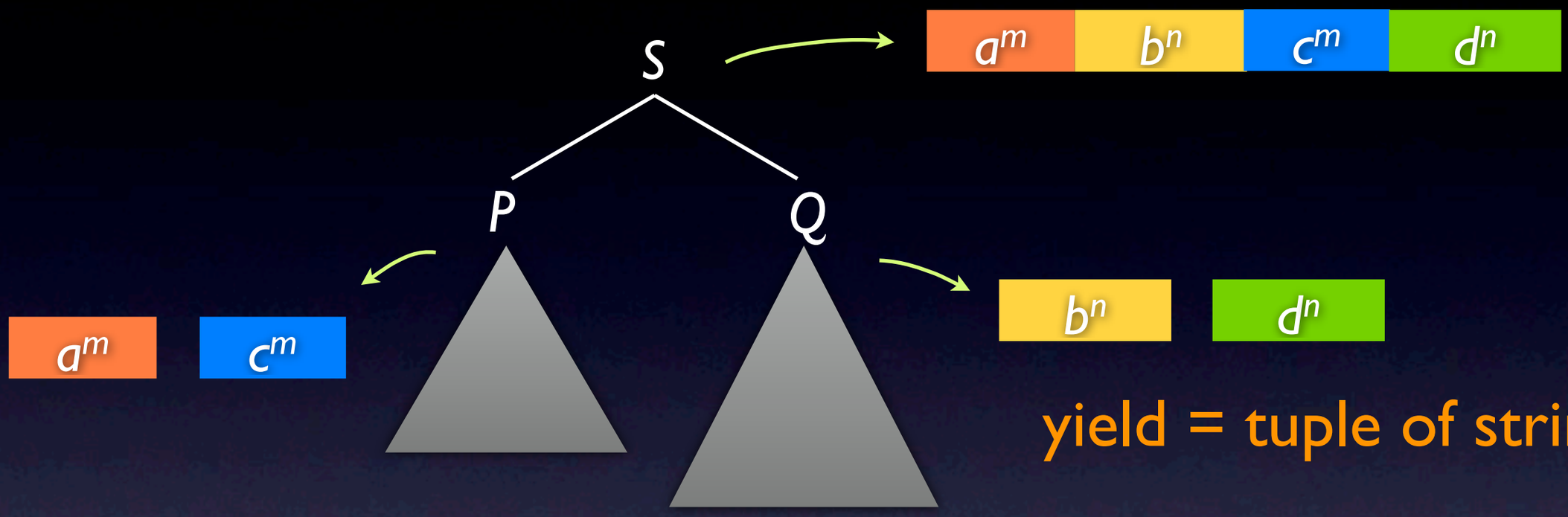
$A(i, k)$:- $sup_{r,n}(i, j), b(j, k)$. $(r = A \rightarrow \alpha \bullet b, n = |\alpha|)$

Notation for magic-sets rewriting.
Supplementary predicates correspond to dotted rules.

Magic-sets rewriting

- Logic program transformation particularly effective for Datalog
- Rougly equivalent to "memoized" top-down evaluation methods (including Earley deduction)
- Binarization of rules, top-down filtering

Multiple context-free grammars



$S(x_1 y_1 x_2 y_2) :- P(x_1, x_2), Q(y_1, y_2).$ bottom-up

$P(a, c).$ $Q(b, d).$

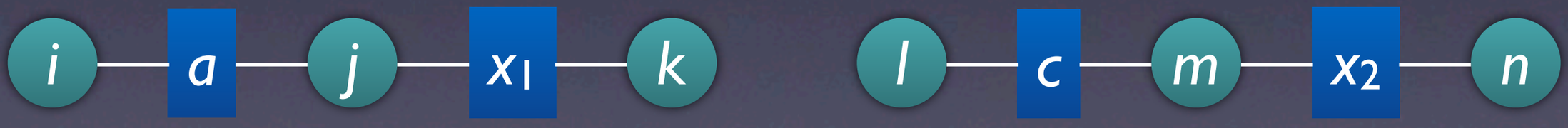
$P(ax_1, cx_2) :- P(x_1, x_2).$ $Q(by_1, dy_2) :- Q(y_1, y_2).$

A slight variation of a by-now-familiar 2-MCFG.
Generates { $a^m b^n c^m d^n$ | m,n ≥1 }
Directly convert to Datalog.

$$S(x_1 y_1 x_2 y_2) :- P(x_1, x_2), Q(y_1, y_2).$$

$$S(i, m) :- P(i, j, k, l), Q(j, k, l, m).$$

$$P(ax_1, cx_2) :- P(x_1, x_2).$$



$$P(i, k, l, n) :- P(j, k, m, n), a(i, j), a(l, m).$$

From MCFG to Datalog

$S(\ x_1\ y_1\ x_2\ y_2\ ) :- P(x_1, x_2), Q(y_1, y_2).$
$\quad i\ \ j\ \ k\ \ l\ \ m$

$P(\ a\ ,\ c\ ).$
$\quad i\ j\ k\ l$

$P(\ a\, x_1\ ,\ c\, x_2\ ) :- P(x_1, x_2).$
$\quad i\ j\ \ k\ lm\ \ n$

$Q(b, d).$

$Q(by_1, dy_2) :- Q(y_1, y_2).$

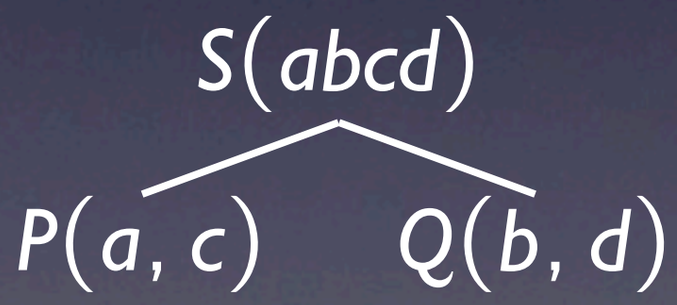
$S(i, m) :- P(i, j, k, l), Q(j, k, l, m).$

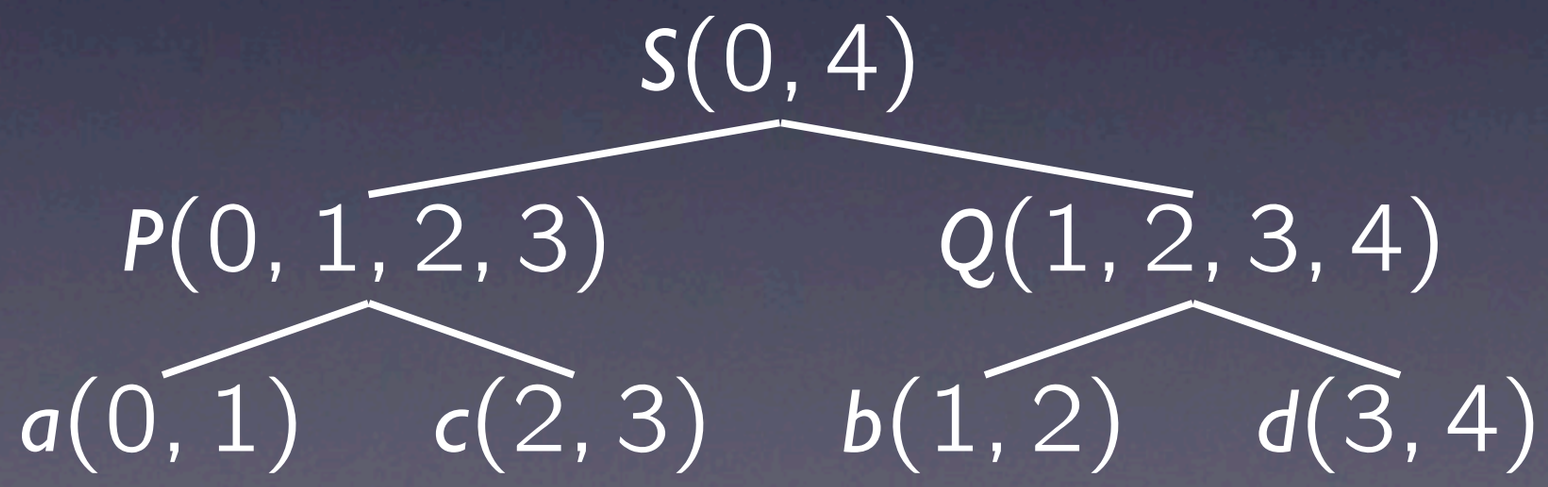
$P(i, j, k, l) :- a(i, j), c(k, l).$

$P(i, k, l, n) :- a(i, j), P(j, k, m, n), c(l, m).$

$Q(i, j, k, l) :- b(i, j), d(k, l).$

$Q(i, k, l, n) :- b(i, j), d(l, m), Q(j, k, m, n).$





Adornment

1: $S^{bf}(i, m) :- P^{bfff}(i, j, k, l), Q^{bbbf}(j, k, l, m).$

2: $P^{bfff}(i, j, k, l) :- a^{bf}(i, j), c^{ff}(k, l).$

3: $P^{bfff}(i, k, l, n) :- a^{bf}(i, j), P^{bfff}(j, k, m, n), c^{fb}(l, m).$

4: $Q^{bbbf}(i, j, k, l) :- b^{bb}(i, j), d^{bf}(k, l).$

5: $Q^{bbbf}(i, k, l, n) :- b^{bf}(i, j), d^{bf}(l, m), Q^{bbbf}(j, k, m, n).$

$?- S(0, x).$

SLD derivation

$\stackrel{1}{\Longrightarrow} ?- P(0, j_1, k_1, l_1), Q(j_1, k_1, l_1, x).$

$\stackrel{2}{\Longrightarrow} ?- a(0, j_1), c(k_1, l_1), Q(j_1, k_1, l_1, x).$

$\stackrel{a(0,1)}{\Longrightarrow} ?- c(k_1, l_1), Q(1, k_1, l_1, x).$

$\stackrel{c(2,3)}{\Longrightarrow} ?- Q(1, 2, 3, x).$

$\stackrel{4}{\Longrightarrow} ?- b(1, 2), d(3, x).$

$\stackrel{b(1,2)}{\Longrightarrow} ?- d(3, x).$

$\stackrel{d(3,4)}{\Longrightarrow} ?- [].$

Applying magic-sets rewriting.
Each argument position is marked as “free” (uninstantiated) or “bound” (instantiated), according to the status of arguments in SLD derivations.
Initial query is assumed to be S(0,x).
SLD derivations generalize leftmost derivations of CFGs to logic programs.

Magic predicates

1: $S^{bf}(i, m) :- P^{bfff}(i, j, k, l), Q^{bbbf}(j, k, l, m).$

2: $P^{bfff}(i, j, k, l) :- a^{bf}(i, j), c^{ff}(k, l).$

3: $P^{bfff}(i, k, l, n) :- a^{bf}(i, j), P^{bfff}(j, k, m, n), c^{fb}(l, m).$

4: $Q^{bbbf}(i, j, k, l) :- b^{bb}(i, j), d^{bf}(k, l).$

5: $Q^{bbbf}(i, k, l, n) :- b^{bf}(i, j), d^{bf}(l, m), Q^{bbbf}(j, k, m, n).$

$?- S(0, x).$ $\quad m\_S(0)$

$\stackrel{1}{\Longrightarrow} ?- P(0, j_1, k_1, l_1), Q(j_1, k_1, l_1, x).$ $\quad m\_P(0)$

$\stackrel{2}{\Longrightarrow} ?- a(0, j_1), c(k_1, l_1), Q(j_1, k_1, l_1, x).$

$\stackrel{a(0,1)}{\Longrightarrow} ?- c(k_1, l_1), Q(1, k_1, l_1, x).$

$\stackrel{c(2,3)}{\Longrightarrow} ?- Q(1, 2, 3, x).$ $\quad m\_Q(1, 2, 3)$

$\stackrel{4}{\Longrightarrow} ?- b(1, 2), d(3, x).$

$\stackrel{b(1,2)}{\Longrightarrow} ?- d(3, x).$

$\stackrel{d(3,4)}{\Longrightarrow} ?- [].$

The fact that S(0,x) is the first goal is represented by the “magic” predicate m_S(0), etc.
Magic predicates take only bound arguments.

Supplementary predicates

$5: Q^{bbbf}(i, k, l, n) :- b^{bf}(i, j), d^{bf}(l, m), Q^{bbbf}(j, k, m, n).$

$?- Q(2, 4, 6, x).$

$\xRightarrow{5} ?- b(2, j), d(6, m), Q(j, 4, m, x).$

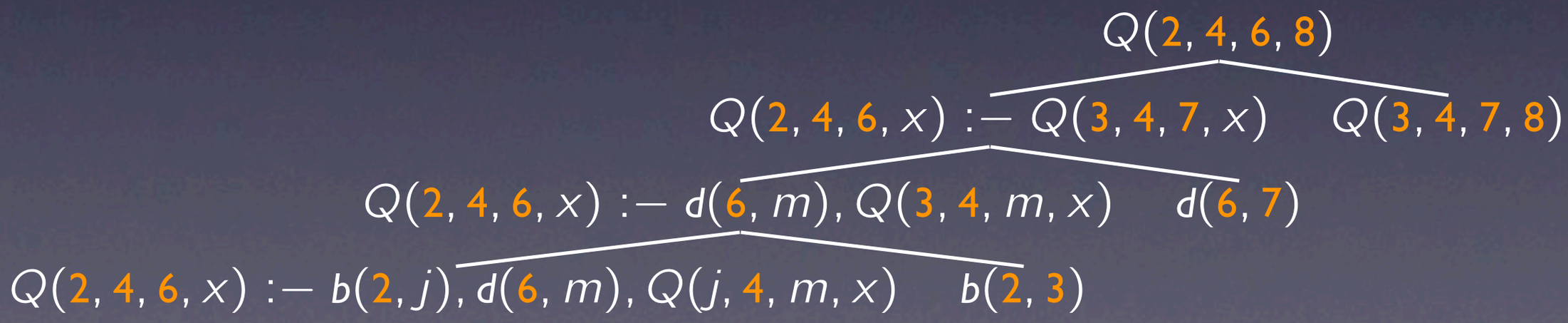
$\xRightarrow{b(2,3)} ?- d(6, m), Q(3, 4, m, x).$

$\xRightarrow{d(6,7)} ?- Q(3, 4, 7, x).$

$\vdots$

$\Longrightarrow ?- [].$

SLD derivation



Here's how rule 4 is converted to Datalog rules.
A Datalog derivation corresponding to an SLD derivation.

Supplementary predicates

$5: Q^{bbbf}(i, k, l, n) :- b^{bf}(i, j), d^{bf}(l, m), Q^{bbbf}(j, k, m, n).$

$?- Q(2, 4, 6, x).$

$\xRightarrow{5} ?- b(2, j), d(6, m), Q(j, 4, m, x).$

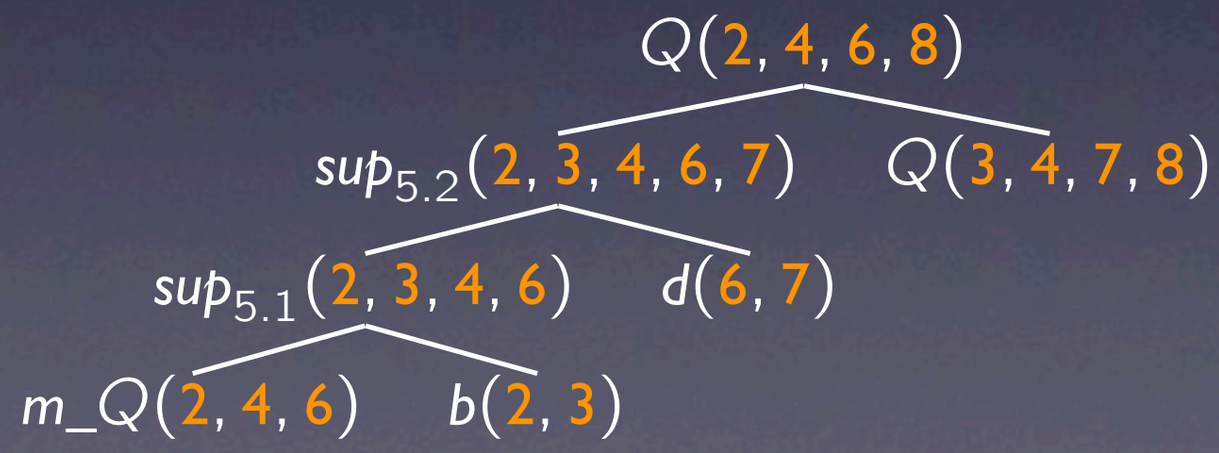
$\xRightarrow{b(2,3)} ?- d(6, m), Q(3, 4, m, x).$

$\xRightarrow{d(6,7)} ?- Q(3, 4, 7, x).$

$\vdots$

$\Longrightarrow ?- [].$

SLD derivation

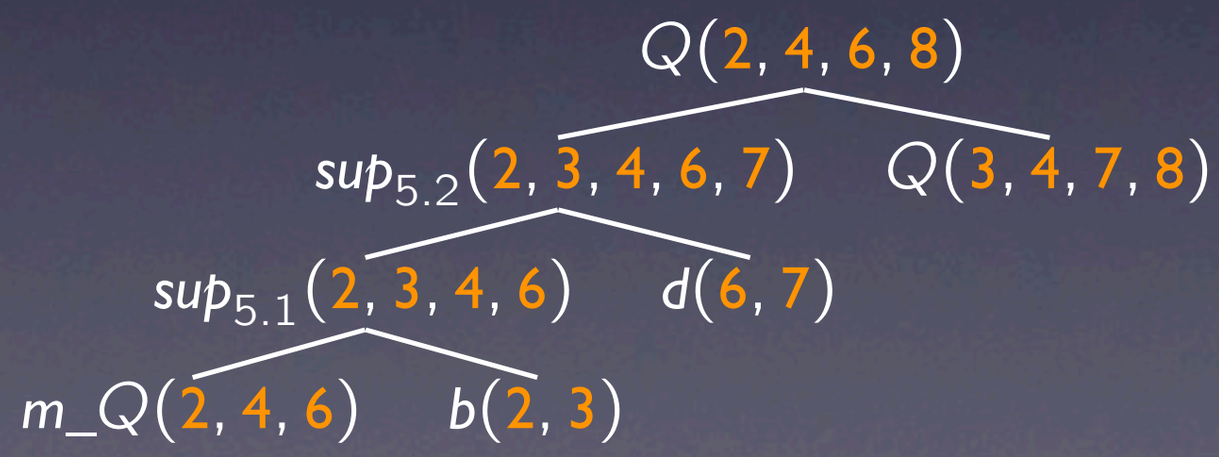


A partially instantiated clause is represented with a magic predicate.
A derived clause is represented with a supplementary predicate.

Rules for supplementary predicates

$$5 : Q^{bbbf}(i, k, l, n) :- b^{bf}(i, j), d^{bf}(l, m), Q^{bbbf}(j, k, m, n).$$

$$sup_{5.1}(i, j, k, l) :- m\_Q(i, k, l), b(i, j).$$
$$sup_{5.2}(i, j, k, l, m) :- sup_{5.1}(i, j, k, l), d(l, m).$$
$$Q(i, k, l, n) :- sup_{5.2}(i, j, k, l, m), Q(j, k, m, n).$$



Rules for magic predicates

1: $S^{bf}(i, m) :- P^{bfff}(i, j, k, l), Q^{bbbf}(j, k, l, m).$

2: $P^{bfff}(i, j, k, l) :- a^{bf}(i, j), c^{ff}(k, l).$

3: $P^{bfff}(i, k, l, n) :- a^{bf}(i, j), P^{bfff}(j, k, m, n), c^{fb}(l, m).$

4: $Q^{bbbf}(i, j, k, l) :- b^{bb}(i, j), d^{bf}(k, l).$

5: $Q^{bbbf}(i, k, l, n) :- b^{bf}(i, j), d^{bf}(l, m), Q^{bbbf}(j, k, m, n).$

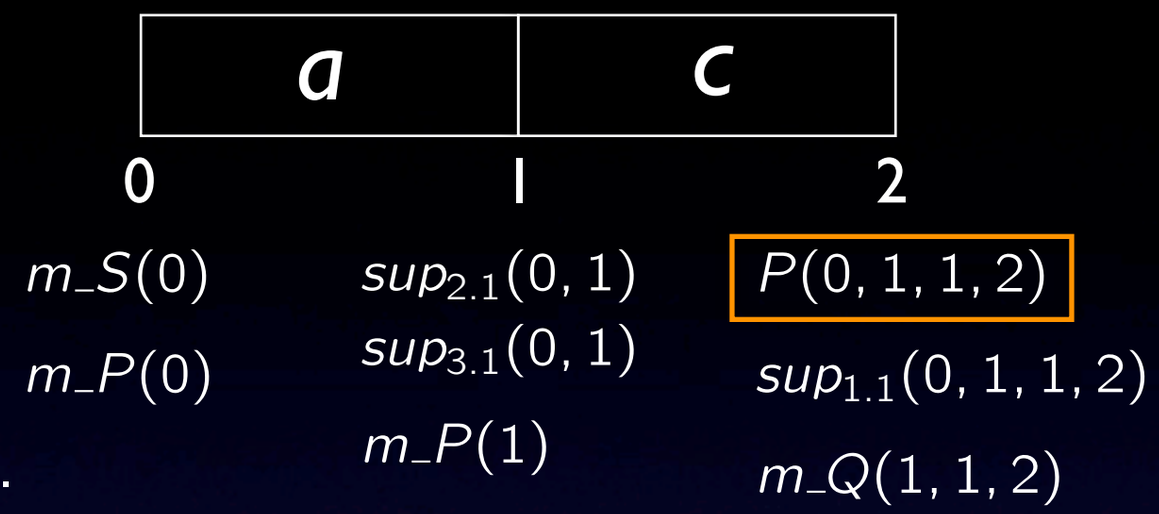
$m\_P(i) :- m\_S(i).$

$m\_Q(j, k, l) :- sup_{1.1}(i, j, k, l).$

$m\_P(j) :- sup_{3.1}(i, j).$

$m\_Q(j, k, m) :- sup_{5.2}(i, j, k, l, m).$

$r_1 : m\_P(i) :- m\_S(i).$
$r_2 : m\_Q(j,k,l) :- sup_{1.1}(i,j,k,l).$
$r_3 : m\_P(j) :- sup_{3.1}(i,j).$
$r_4 : m\_Q(j,k,m) :- sup_{5.2}(i,j,k,l,m).$
$r_5 : sup_{1.1}(i,j,k,l) :- m\_S(i), P(i,j,k,l).$
$r_6 : S(i,m) :- sup_{1.1}(i,j,k,l), Q(j,k,l,m).$
$r_7 : sup_{2.1}(i,j) :- m\_P(i), a(i,j).$
$r_8 : P(i,j,k,l) :- sup_{2.1}(i,j), c(k,l).$
$r_9 : sup_{3.1}(i,j) :- m\_P(i), a(i,j).$
$r_{10} : sup_{3.2}(i,k,m,n) :- sup_{3.1}(i,j), P(j,k,m,n).$
$r_{11} : P(i,k,l,n) :- sup_{3.2}(i,k,m,n), c(l,m).$
$r_{12} : sup_{4.1}(i,j,k) :- m\_Q(i,j,k), b(i,j).$
$r_{13} : Q(i,j,k,l) :- sup_{4.1}(i,j,k), d(k,l).$
$r_{14} : sup_{5.1}(i,j,k,l) :- m\_Q(i,k,l), b(i,j).$
$r_{15} : sup_{5.2}(i,j,k,l,m) :- sup_{5.1}(i,j,k,l), d(l,m).$
$r_{16} : Q(i,k,l,n) :- sup_{5.2}(i,j,k,l,m), Q(j,k,m,n).$



not prefix-correct!

Magic-sets corresponds to Earley for CFGs, but in general, does not imply prefix-correctness.

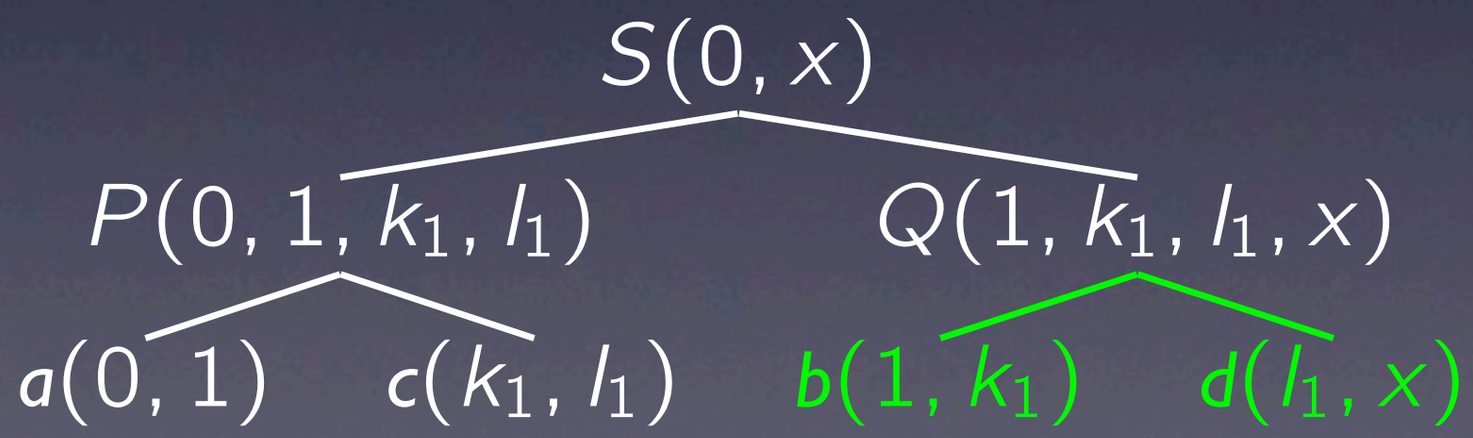
Cause of non-prefix-correctness

SLD derivation

$$?- S(0, x).$$
$$\overset{1}{\Longrightarrow} ?- P(0, j_1, k_1, l_1), Q(j_1, k_1, l_1, x).$$
$$\overset{2}{\Longrightarrow} ?- a(0, j_1), c(k_1, l_1), Q(j_1, k_1, l_1, x).$$
$$\overset{a(0,1)}{\Longrightarrow} ?- c(k_1, l_1), Q(1, k_1, l_1, x).$$
$$\overset{c(1,2)}{\Longrightarrow} ?- Q(1, 1, 2, x).$$
$$\overset{4}{\Longrightarrow} ?- b(1, 1), d(2, x).$$

adornment

$$2\colon P^{bfff}(i, j, k, l) :- a^{bf}(i, j), c^{ff}(k, l).$$



incomplete derivation tree

SLD derivation with the original program is not prefix-correct.
The order of extensional facts at the leaves of the derivation tree does not match the order of terminals in the input.

Securing prefix-correctness

$S(\underset{i}{x_1}\ \underset{j}{y_1}\ \underset{k}{x_2}\ \underset{l}{y_2}\ \underset{m}{}) :- P(x_1, x_2), Q(y_1, y_2).$

~~$S(i, m) :- P(i, j, k, l), Q(j, k, l, m).$~~

$S(i, m) :- P_1(i, j), Q_1(j, k), P(i, j, k, l), Q(j, k, l, m).$

~~$P(i, j, k, l) :- a(i, j), c(k, l).$~~ $P(i, j, k, l) :- aux(i, j), c(k, l).$

~~$P_1(i, j) :- a(i, j).$~~ $P_1(i, j) :- aux(i, j).$

$aux(i, j) :- a(i, j).$

$S^{bf}(i, m) :- P_1^{bf}(i, j), Q_1^{bf}(j, k), P^{bbbf}(i, j, k, l), Q^{bbbf}(j, k, l, m).$

$P^{bbbf}(i, j, k, l) :- aux^{bb}(i, j), c^{bf}(k, l).$

$P_1^{bf}(i, j) :- aux^{bf}(i, j).$

$aux^{bf}(i, j) :- a^{bf}(i, j).$

Add “redundant” subgoals.
The aux predicate “folds” common part of two rules.

New starting point

1: $S^{bf}(i, m) :- P_1^{bf}(i, j), Q_1^{bf}(j, k), P^{bbbf}(i, j, k, l), Q^{bbbf}(j, k, l, m).$

2: $P_1^{bf}(i, j) :- aux_2^{bf}(i, j).$

3: $P^{bbbf}(i, j, k, l) :- aux_2^{bf}(i, j), c^{bf}(k, l).$

4: $aux_2^{bf}(i, j) :- a^{bf}(i, j).$

5: $P_1^{bf}(i, k) :- aux_3^{bff}(i, j, k).$

6: $P^{bbbf}(i, k, l, n) :- aux_3^{bff}(i, j, k), c^{bf}(l, m), P^{bbbf}(j, k, m, n).$

7: $aux_3^{bff}(i, j, k) :- a^{bf}(i, j), P_1^{bf}(j, k).$

8: $Q_1^{bf}(i, j) :- aux_4^{bf}(i, j).$

9: $Q^{bbbf}(i, j, k, l) :- aux_4^{bf}(i, j), d^{bf}(k, l).$

10: $aux_4^{bf}(i, j) :- b^{bf}(i, j).$

11: $Q_1^{bf}(i, k) :- aux_5^{bff}(i, j, k).$

12: $Q^{bbbf}(i, k, l, n) :- aux_5^{bff}(i, j, k), d^{bf}(l, m), Q^{bbbf}(j, k, m, n).$

13: $aux_5^{bff}(i, j, k) :- b^{bf}(i, j), Q_1^{bf}(j, k).$

The result of introducing redundant subgoals to the original program.

Prefix-correctness

SLD derivation

$?- S(0, x).$

$\overset{1}{\Longrightarrow} ?- P_1(0, j_1), Q_1(j_1, k_1), P(0, j_1, k_1, l_1), Q(j_1, k_1, l_1, x).$

$\overset{2}{\Longrightarrow} ?- \mathit{aux}_2(0, j_i), Q_1(j_1, k_1), P(0, j_1, k_1, l_1), Q(j_1, k_1, l_1, x).$

$\overset{4}{\Longrightarrow} ?- a(0, j_1), Q_1(j_1, k_1), P(0, j_1, k_1, l_1), Q(j_1, k_1, l_1, x).$

$\overset{a(0,1)}{\Longrightarrow} ?- Q_1(1, k_1), P(0, 1, k_1, l_1), Q(1, k_1, l_1, x).$

$\overset{8}{\Longrightarrow} ?- \mathit{aux}_4(1, k_1), P(0, 1, k_1, l_1), Q(1, k_1, l_1, x).$

$\overset{10}{\Longrightarrow} ?- b(1, k_1), P(0, j_1, k_1, l_1), Q(1, k_1, l_1, x).$

$\overset{b(1,2)}{\Longrightarrow} ?- P(0, 1, 2, l_1), Q(1, 2, l_1, x).$

$\overset{3}{\Longrightarrow} ?- \mathit{aux}_3(0, 1), c(2, l_1), Q(1, 2, l_1, x).$

$\vdots$

$\Longrightarrow ?- c(2, l_1), Q(1, 2, l_1, x).$

$\overset{c(2,3)}{\Longrightarrow} ?- Q(1, 2, 3, x).$

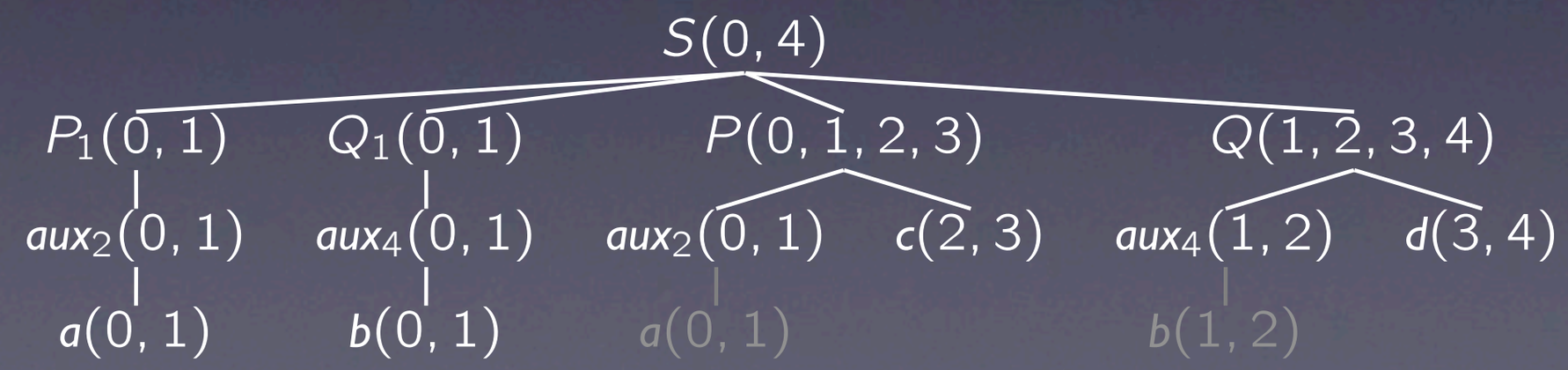
$\overset{9}{\Longrightarrow} ?- \mathit{aux}_4(1, 2), d(3, x).$

$\vdots$

$\Longrightarrow ?- d(3, x).$

$\overset{d(3,4)}{\Longrightarrow} ?- []$

derivation tree



SLD derivation is now prefix-correct.

Magic-sets rewriting

$r_1 : m\_P_1(i) :- m\_S(i).$
$r_2 : m\_Q_1(j) :- sup_{1.1}(i, j).$
$r_3 : m\_P(i, j, k) :- sup_{1.2}(i, j, k).$
$r_4 : m\_Q(j, k, l) :- sup_{1.3}(i, j, k, l).$
$r_5 : m\_aux_2(i) :- m\_P_1(i).$
$r_6 : m\_aux_2(i) :- m\_P(i, j, k).$
$r_7 : m\_aux_3(j) :- m\_P_1(i).$
$r_8 : m\_aux_3(i) :- m\_P(i, k, l).$
$r_9 : m\_P(j, k, m) :- sup_{6.2}(i, j, k, l, m).$
$r_{10} : m\_P_1(j) :- sup_{7.1}(i, j).$
$r_{11} : m\_aux_4(i) :- m\_Q_1(i).$
$r_{12} : m\_aux_4(i) :- m\_Q(i, j, k).$
$r_{13} : m\_aux_5(i) :- m\_Q_1(i).$
$r_{14} : m\_aux_5(i) :- m\_Q(i, k, l).$
$r_{15} : m\_Q(j, k, m) :- sup_{12.2}(i, j, k, l, m).$
$r_{16} : m\_Q_1(j) :- sup_{13.1}(i, j).$

$r_{17} : sup_{1.1}(i, j) :- m\_S(i), P_1(i, j).$
$r_{18} : sup_{1.2}(i, j, k) :- sup_{1.1}(i, j), Q_1(j, k).$
$r_{19} : sup_{1.3}(i, j, k, l) :- sup_{1.2}(i, j, k), P(i, j, k, l).$
$r_{20} : S(i, m) :- sup_{1.3}(i, j, k, l), Q(j, k, l, m).$

$r_{21} : P_1(i, j) :- m\_P_1(i), aux_2(i, j).$
$r_{22} : sup_{3.1}(i, j, k) :- m\_P(i, j, k), aux_2(i, j).$
$r_{23} : P(i, j, k, l) :- sup_{3.1}(i, j, k), c(k, l).$
$r_{24} : aux_2(i, j) :- m\_aux_2(i), a(i, j).$
$r_{25} : P_1(i, k) :- m\_P_1(i), aux_3(i, j, k).$
$r_{26} : sup_{6.1}(i, j, k, l) :- m\_P(i, k, l), aux_3(i, j, k).$
$r_{27} : sup_{6.2}(i, j, k, l, m) :- sup_{6.1}(i, j, k, l), c(l, m).$
$r_{28} : P(i, k, l, n) :- sup_{6.2}(i, j, k, l, m), P(j, k, m, n).$
$r_{29} : sup_{7.1}(i, j) :- m\_aux_3(i), a(i, j).$
$r_{30} : aux_3(i, j, k) :- sup_{7.1}(i, j), P_1(j, k).$
$r_{31} : Q_1(i, j) :- m\_Q_1(i), aux_4(i, j).$
$r_{32} : sup_{9.1}(i, j, k) :- m\_Q(i, j, k), aux_4(i, j).$
$r_{33} : Q(i, j, k, l) :- sup_{9.1}(i, j, k), d(k, l).$
$r_{34} : aux_4(i, j) :- m\_aux_4(i), b(i, j).$
$r_{35} : Q_1(i, k) :- m\_Q_1(i), aux_5(i, _j, k).$
$r_{36} : sup_{12.1}(i, j, k, l) :- m\_Q(i, k, l), aux_5(i, j, k).$
$r_{37} : sup_{12.2}(i, j, k, l, m) :- sup_{12.1}(i, j, k, l), d(l, m).$
$r_{38} : Q(i, k, l, n) :- sup_{12.2}(i, j, k, l, m), Q(j, k, m, n).$
$r_{39} : sup_{13.1}(i, j) :- m\_aux_5(i), b(i, j).$
$r_{40} : aux_5(i, j, k) :- sup_{13.1}(i, j), Q_1(j, k).$

Application to TAG

$Q(i, l, m, p) :- B(i, j, o, p), a(j, k), H(k, l, m, n), R(n, o).$

$aux(i, j, k, l) :- B_1(i, j), a(j, k), H_1(k, l).$

$Q_1(i, l) :- aux(i, j, k, l).$

$Q(i, l, m, p) :- aux(i, j, k, l), H(k, l, m, n), R(n, o), B(i, j, o, p).$

Direct conversion to Datalog.
Essentially the method of Lang.

Magic-sets rewriting

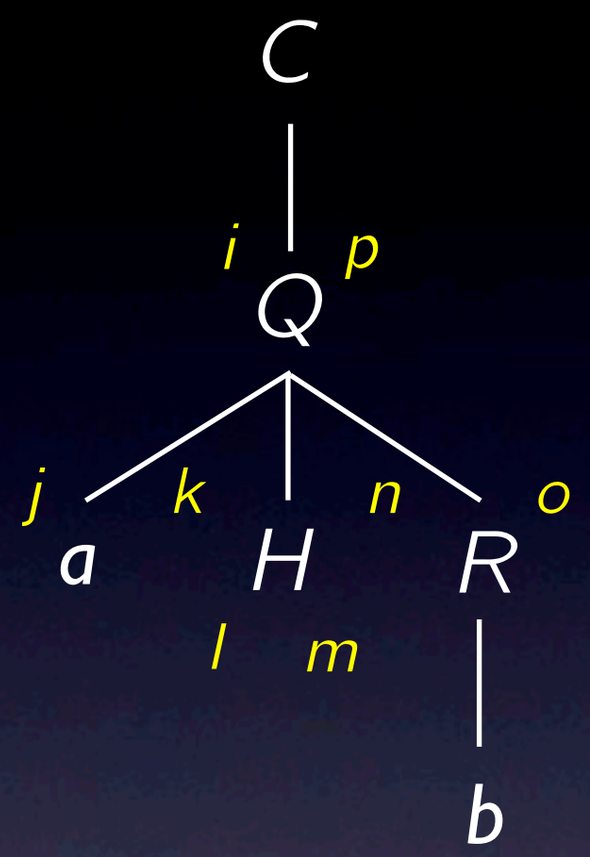
$m\_B_1(i) :- m\_aux(i).$
$m\_H_1(k) :- sup_{1.2}(i, j, k).$
$m\_aux(i) :- m\_Q_1(i).$
$m\_aux(i) :- m\_Q(i, l, m).$
$m\_H(k, l, m) :- sup_{3.1}(i, j, k, l, m).$
$m\_R(n) :- sup_{3.2}(i, j, l, m, n).$
$m\_B(i, j, o) :- sup_{3.3}(i, j, l, m, o).$

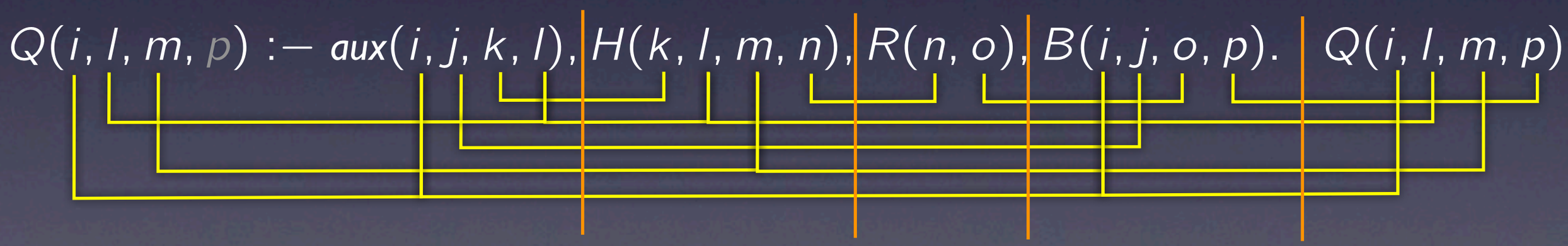
$sup_{1.1}(i, j) :- m\_aux(i), B_1(i, j).$
$sup_{1.2}(i, j, k) :- sup_{1.1}(i, j), a(j, k).$
$aux(i, j, k, l) :- sup_{1.2}(i, j, k), H_1(k, l).$
$Q_1(i, l) :- m\_Q_1(i, l), aux(i, j, k, l).$
$sup_{3.1}(i, j, k, l, m) :- m\_Q(i, l, m), aux(i, j, k, l).$
$sup_{3.2}(i, j, l, m, n) :- sup_{3.1}(i, j, k, l, m), H(k, l, m, n).$
$sup_{3.3}(i, j, l, m, o) :- sup_{3.2}(i, j, l, m, n), R(n, o).$
$Q(i, l, m, p) :- sup_{3.3}(i, j, l, m, o), B(i, j, o, p).$

$O(n^6)$

Apply the technique for MCFGs.

Time and space complexity

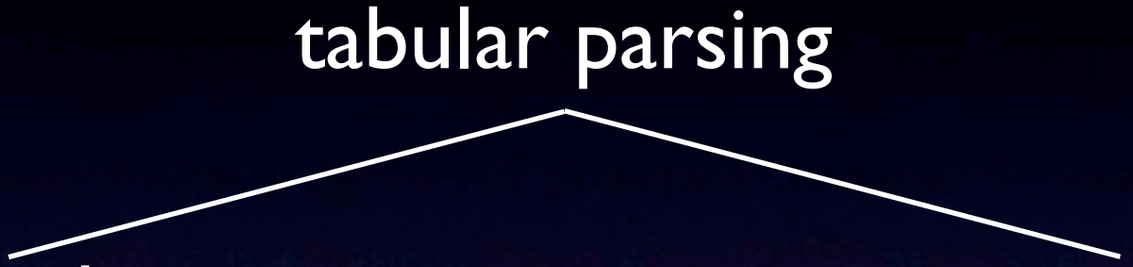




Optimal complexity bounds are obtained without any fine-tuning

The arity of supplementary predicate indicated by the number of yellow lines crossing orange.
The same time and space complexity as Nederhof's prefix-correct Earley recognizer for TAG, previously known optimal bound.
Nothing ad hoc except the introduction of redundant subgoals.
A special case of genenral technique for MCFGs.

How parsing should be approached



program transformation

(grammar → deduction system)

Datalog program

canonical chart parsing
(≈ seminaive bottom-up evaluation)

Datalog program

- Use well-established, general formalism/method
- Avoids ad hoc techniques as much as possible
- Easier to understand and easier to prove correct

Some people even divide parsing schema into conversion to PDA and “PDA tabulation”. This is not necessarily a natural approach.

How parsing should be approached

Earley parsing

parsing schema

introduction of redundant predicates
+ magic-sets rewriting
(grammar → deduction system)

Datalog program

control algorithm

canonical chart parsing
(≈ seminaive bottom-up evaluation)

Datalog program

- Use well-established, general formalism/method
- Avoids ad hoc techniques as much as possible
- Easier to understand and easier to prove correct

In the case of Earley, introduction of redundant predicates plus magic-sets rewriting.