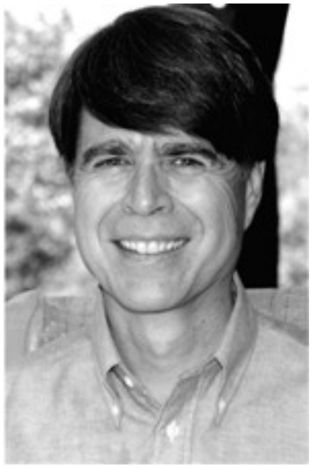
Relation Between NGA and Parsers

- The non-determinism in the NGA is called globally angelic non-determinism.
    - For a given grammar and input string in the language generated by that grammar, the non-deterministic NGA transitions at start nodes have to ensure that the NGA ultimately reaches $S\bullet$ along a complete balanced path that generates the input string.
- Parsing algorithms are deterministic implementations of this globally angelic non-determinism.
- We will examine a universal parsing algorithm by Jay Earley described in his 1968 PhD thesis.

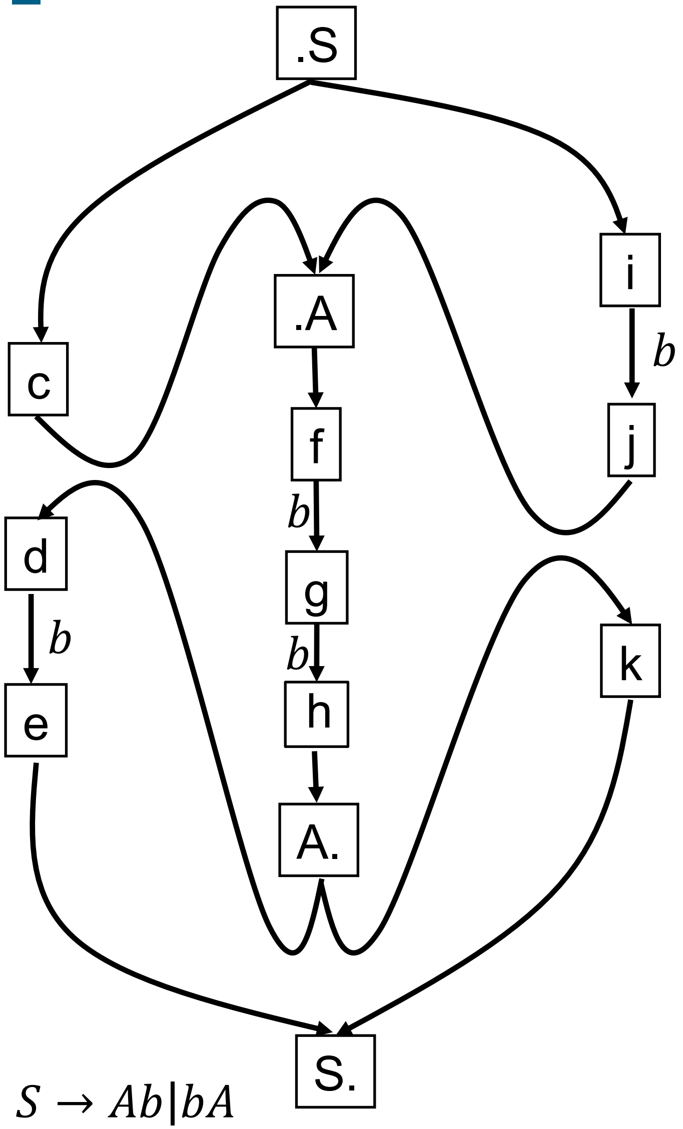


Earley’s Algorithm

- Universal, i.e., can handle any CFG.
- For an input string of length n, the algorithm runs in $O(n^3)$ steps and $O(n^2)$ space.
- Can run faster for particular grammar structures.
  - $O(n^2)$ steps for unambiguous grammars.
  - $O(n)$ steps for LR(k) grammars.

- Difficult to explain in classical parsing theory formalisms.
  - E.g., “top-down restricted breadth-first bottom-up parsing”. (Huh?)
- Has a very simple interpretation in terms of the GFG.
  - Earley’s algorithm is a deterministic implementation of the NGA and is the context-free grammar analog of the well-known ε–closure algorithm for simulating NFAs.
  - While the ε–closure algorithm tracks reachability along prefixes of complete paths, Earley’s algorithm tracks reachability along prefixes of complete *balanced* paths.

Earley's Algorithm vis-à-vis ε-Closure



$S \to Ab|bA$
$A \to bb$

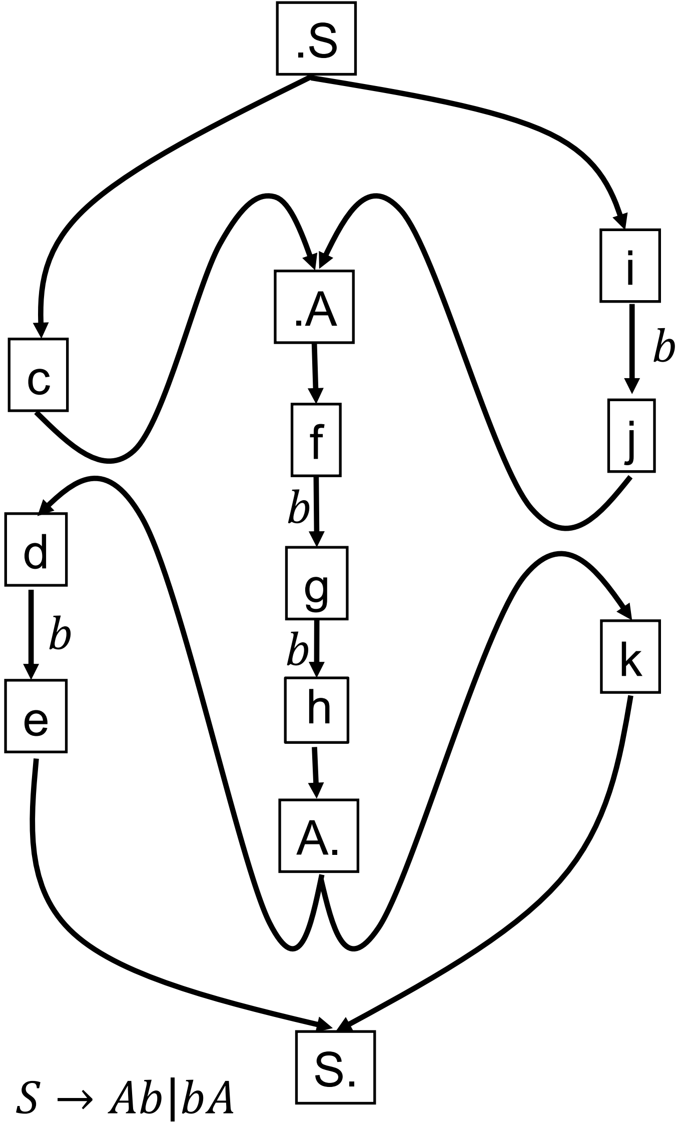
.S .A i f	c

Σ_0 $\bullet\, bbb$

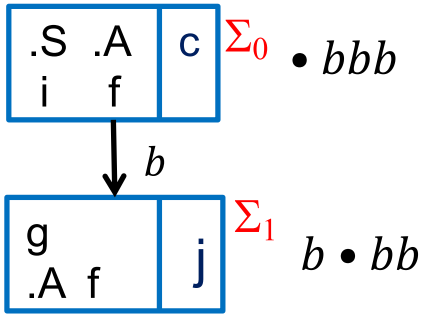
NFA reachability
(ε-closure)

NGA reachability
(Earley)

Earley’s Algorithm vis-à-vis ε-Closure



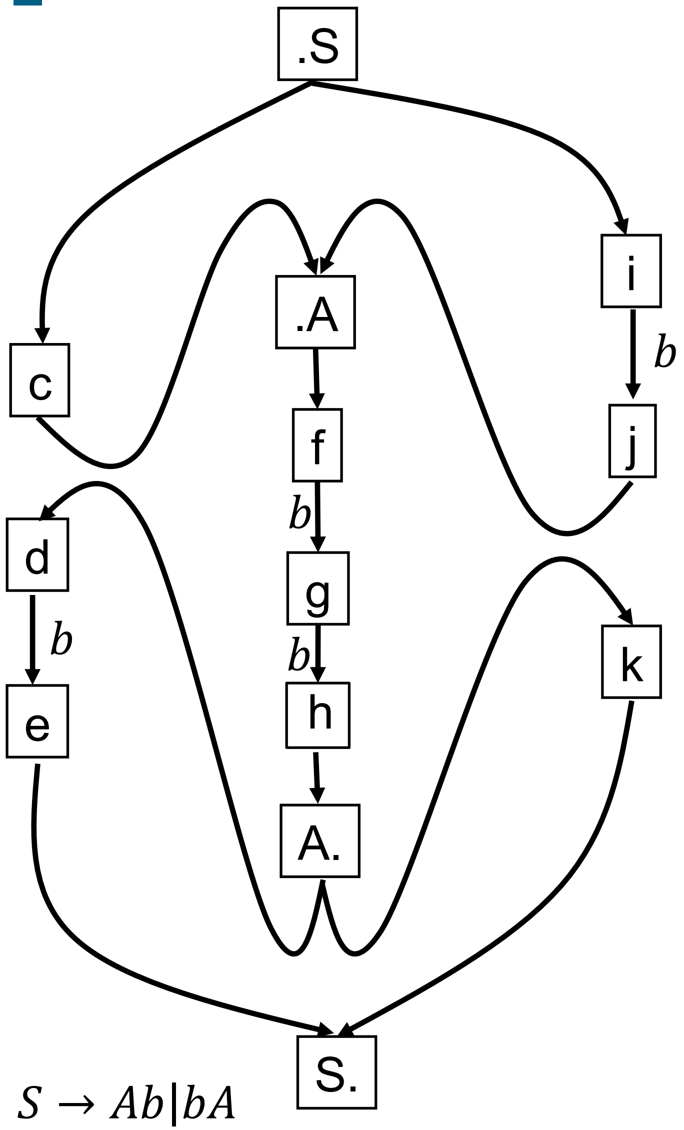
$$S \rightarrow Ab|bA$$
$$A \rightarrow bb$$



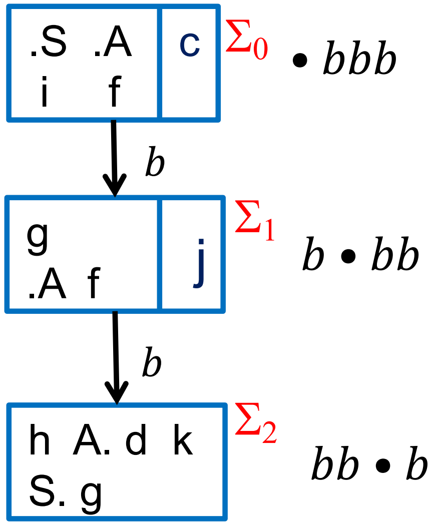
NFA reachability
(ε-closure)

NGA reachability
(Earley)

Earley’s Algorithm vis-à-vis ε-Closure



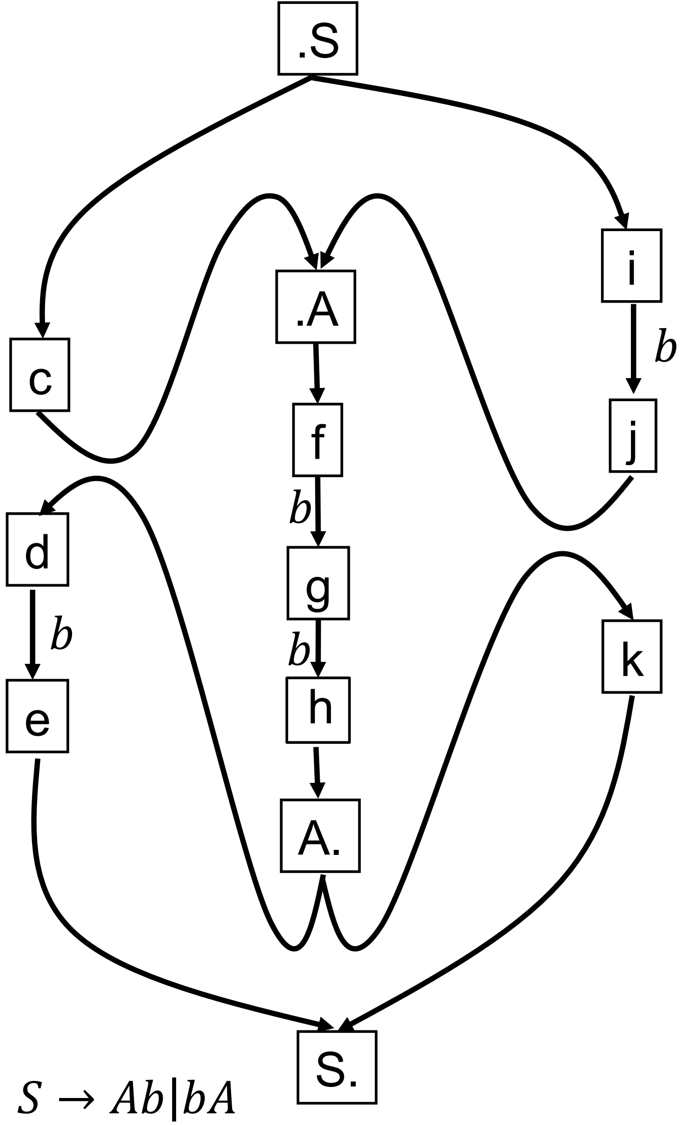
$S \rightarrow Ab|bA$
$A \rightarrow bb$



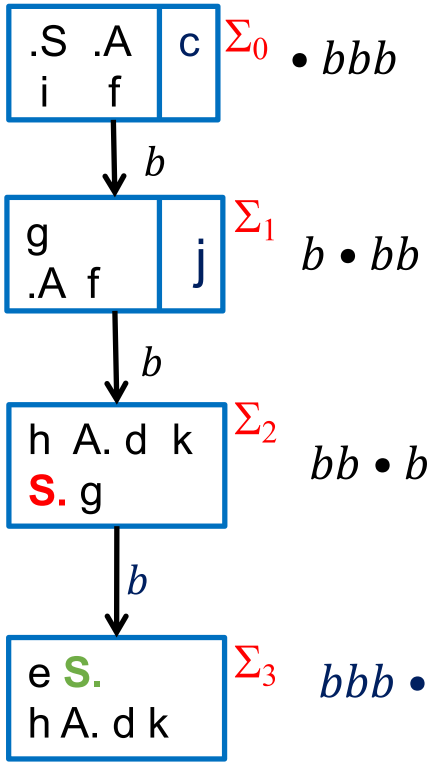
NFA reachability
(ε-closure)

NGA reachability
(Earley)

Earley’s Algorithm vis-à-vis ε-Closure



$S \rightarrow Ab|bA$
$A \rightarrow bb$



Σ_0: .S .A i f | c — $\bullet\, bbb$

b

Σ_1: g .A f | j — $b \bullet bb$

b

Σ_2: h A. d k **S.** g — $bb \bullet b$

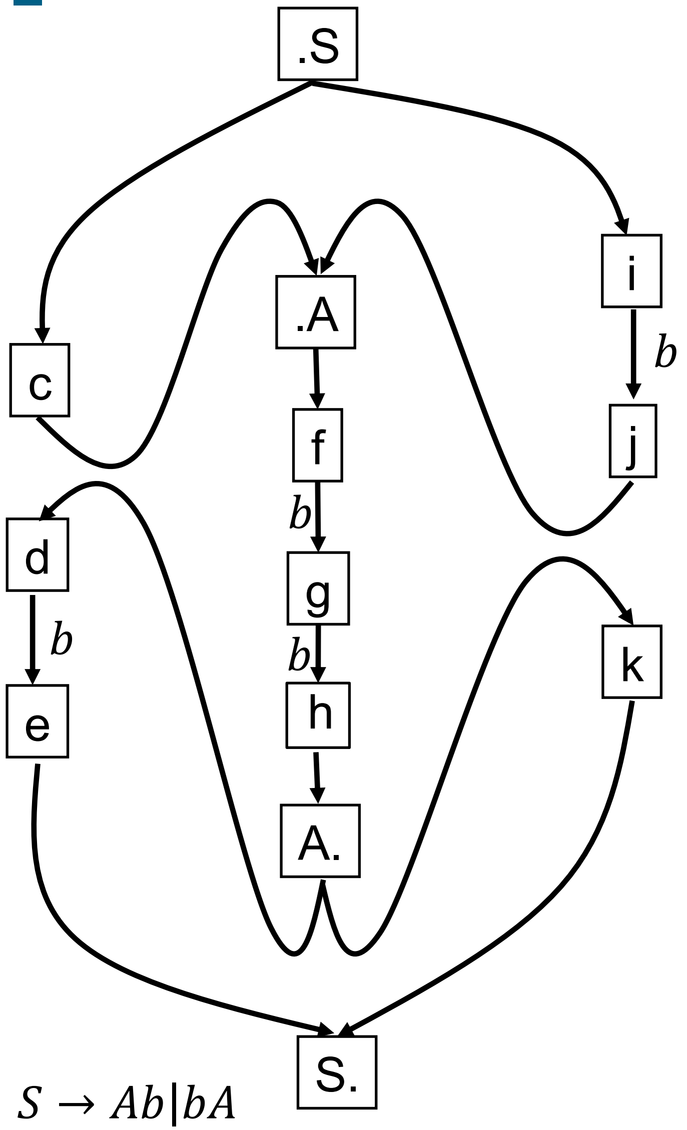
b

Σ_3: e **S.** h A. d k — $bbb\, \bullet$

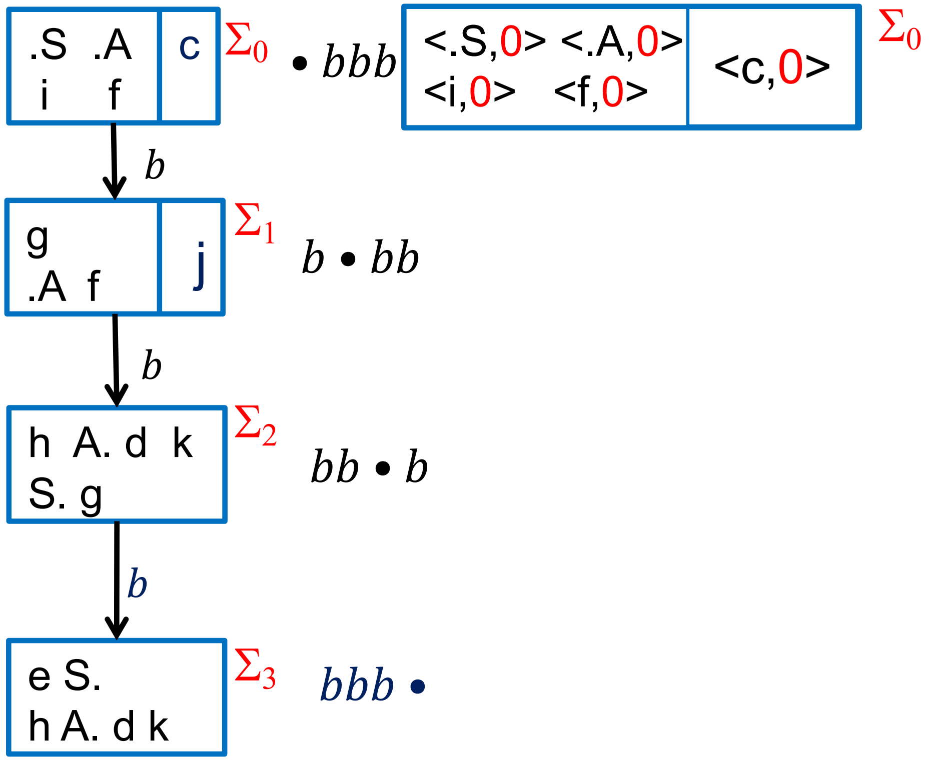
NFA reachability
(ε-closure)

NGA reachability
(Earley)

Earley’s Algorithm vis-à-vis ε-Closure



$S \rightarrow Ab|bA$
$A \rightarrow bb$



.S .A c
i f
Σ_0
$\bullet\ bbb$

<.S,0> <.A,0> <c,0>
<i,0> <f,0>
Σ_0

b

g j
.A f
Σ_1
$b \bullet bb$

b

h A. d k
S. g
Σ_2
$bb \bullet b$

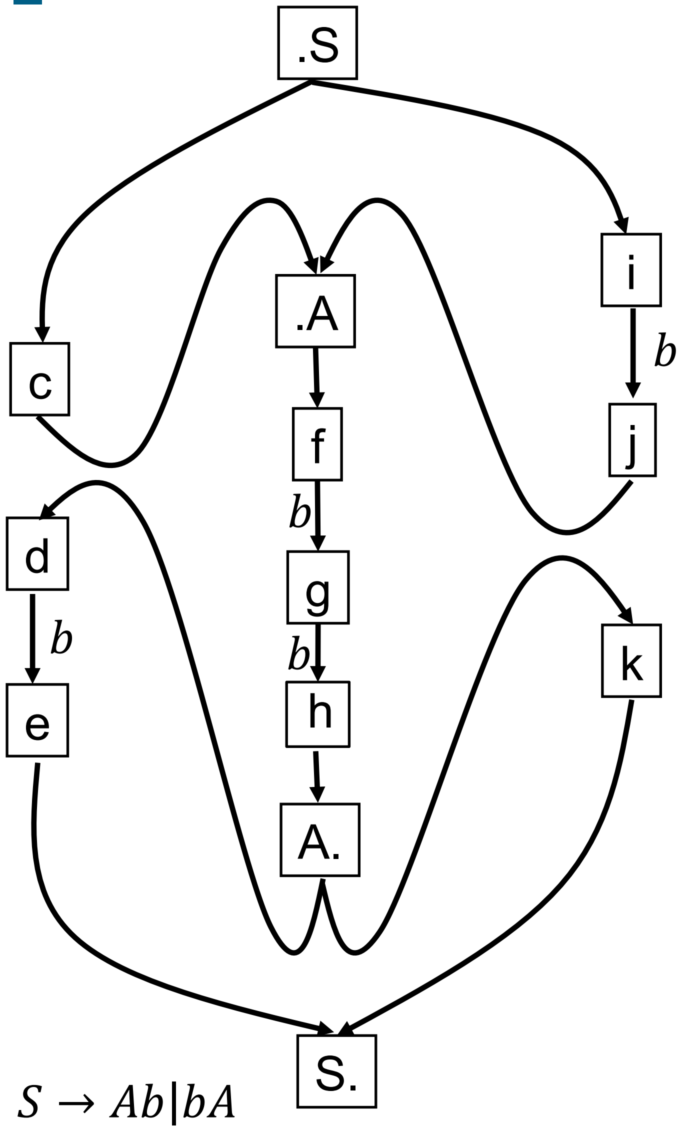
b

e S.
h A. d k
Σ_3
$bbb\ \bullet$

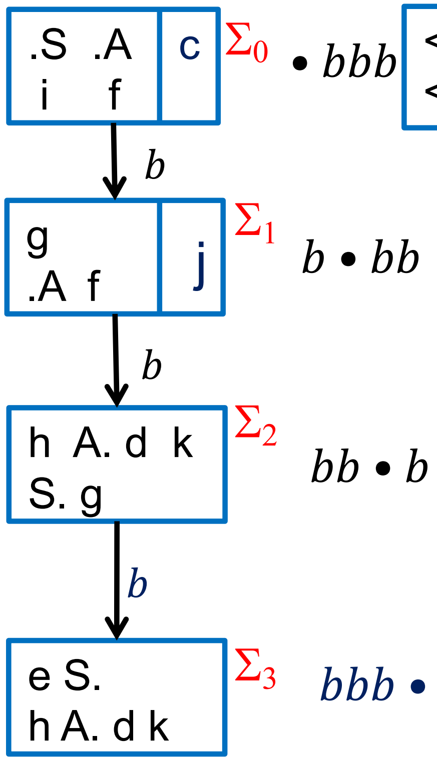
NFA reachability
(ε-closure)

NGA reachability
(Earley)

Earley’s Algorithm vis-à-vis ε-Closure



$$S \to Ab|bA$$
$$A \to bb$$



.S .A | c Σ_0 $\bullet\, bbb$
i f

b

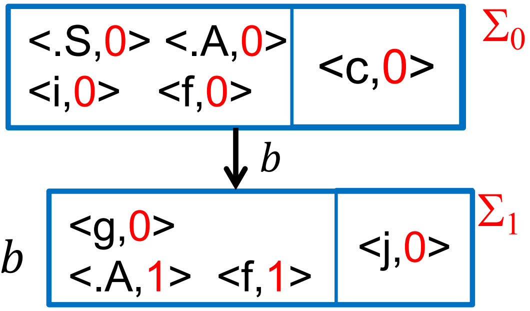
g | j Σ_1 $b \bullet bb$
.A f

b

h A. d k Σ_2 $bb \bullet b$
S. g

b

e S. Σ_3 $bbb \bullet$
h A. d k



<.S,0> <.A,0> | <c,0> Σ_0
<i,0> <f,0>

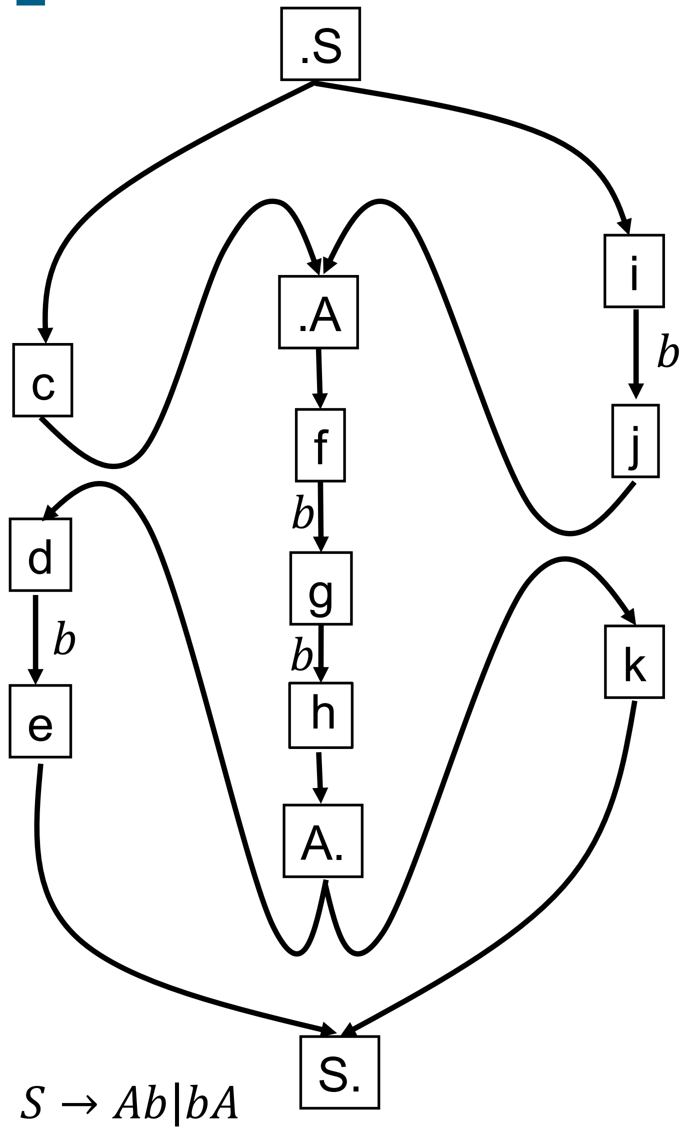
b

<g,0> | <j,0> Σ_1
<.A,1> <f,1>

NFA reachability
(ε-closure)

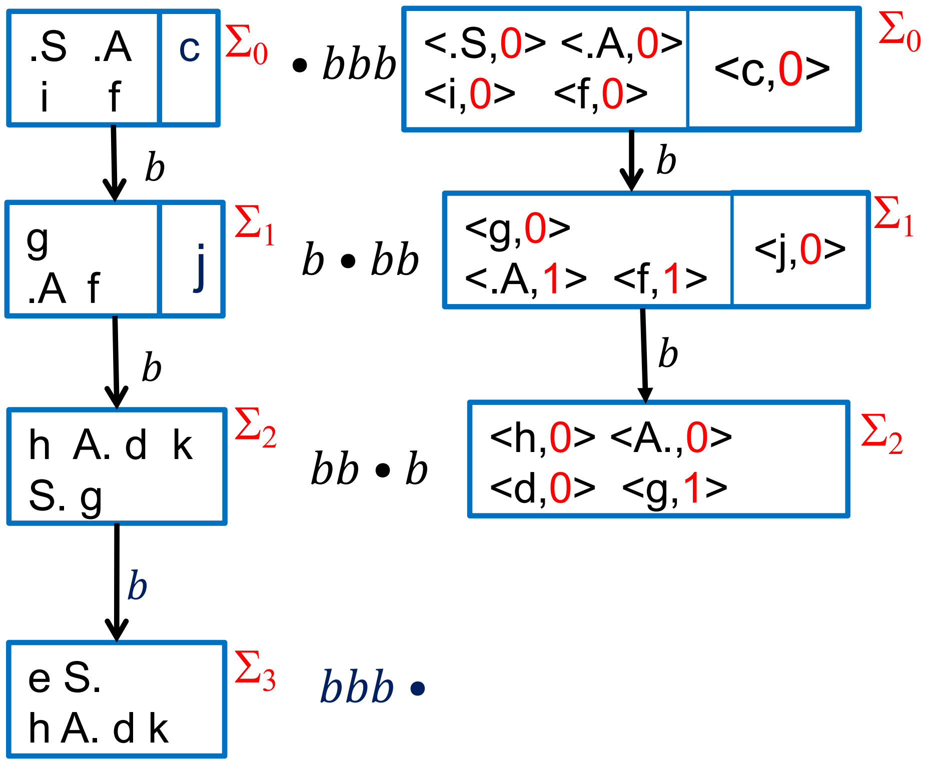
NGA reachability
(Earley)

Earley's Algorithm vis-à-vis ε-Closure



$S \rightarrow Ab|bA$

$A \rightarrow bb$



.S .A i f | c Σ_0 $\bullet\, bbb$

b

g .A f | j Σ_1 $b \bullet bb$

b

h A. d k S. g Σ_2 $bb \bullet b$

b

e S. h A. d k Σ_3 $bbb\, \bullet$

<.S,0> <.A,0> <i,0> <f,0> | <c,0> Σ_0

b

<g,0> <.A,1> <f,1> | <j,0> Σ_1

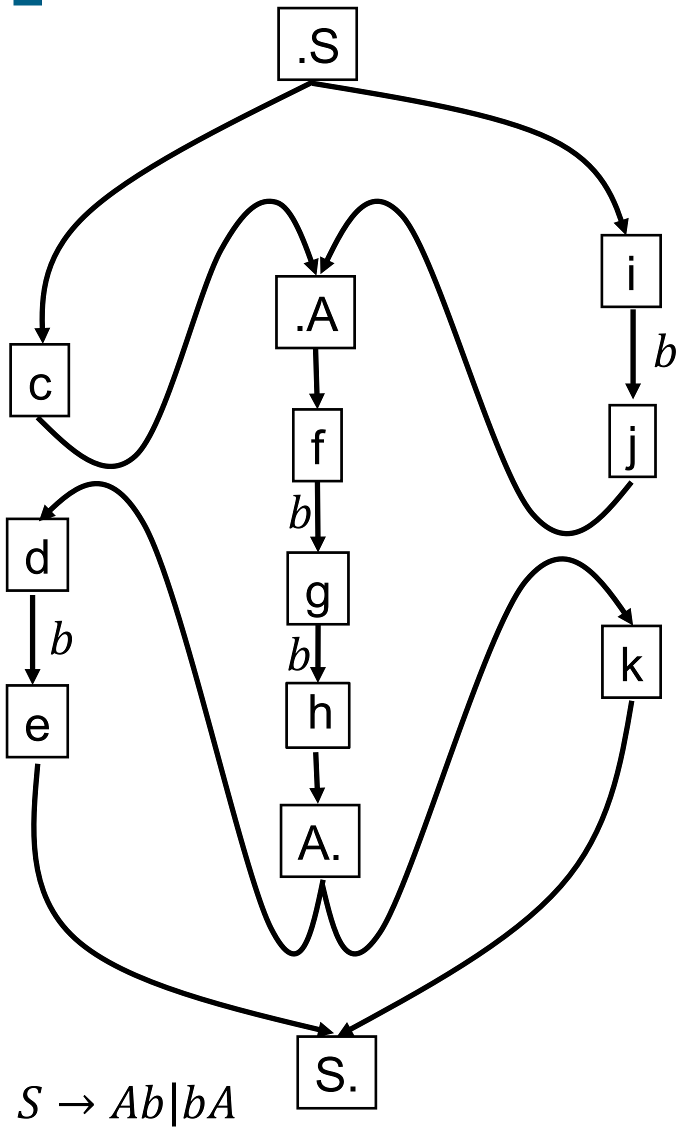
b

<h,0> <A.,0> <d,0> <g,1> Σ_2

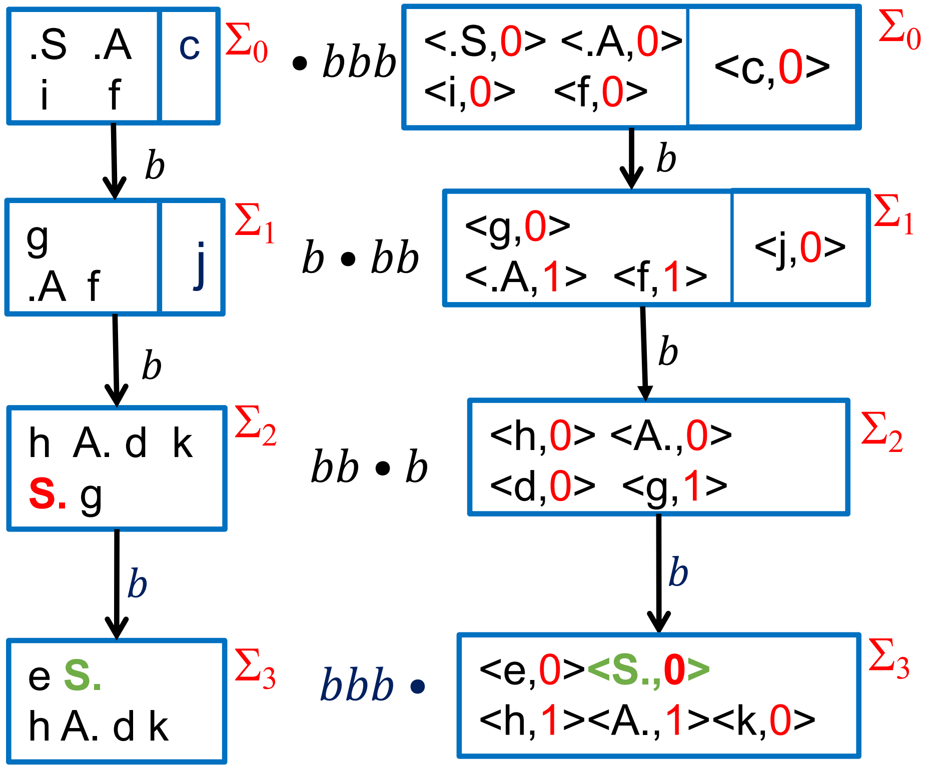
NFA reachability
(ε-closure)

NGA reachability
(Earley)

Earley’s Algorithm vis-à-vis ε-Closure



$S \rightarrow Ab|bA$
$A \rightarrow bb$



Σ_0 : .S .A i f | c — $\bullet\, bbb$ — <.S,0> <.A,0> <i,0> <f,0> | <c,0>

b

Σ_1 : g .A f | j — $b \bullet bb$ — <g,0> <.A,1> <f,1> | <j,0>

b

Σ_2 : h A. d k S. g — $bb \bullet b$ — <h,0> <A.,0> <d,0> <g,1>

b

Σ_3 : e S. h A. d k — $bbb \bullet$ — <e,0><S.,0> <h,1><A.,1><k,0>

NFA reachability
(ε-closure)

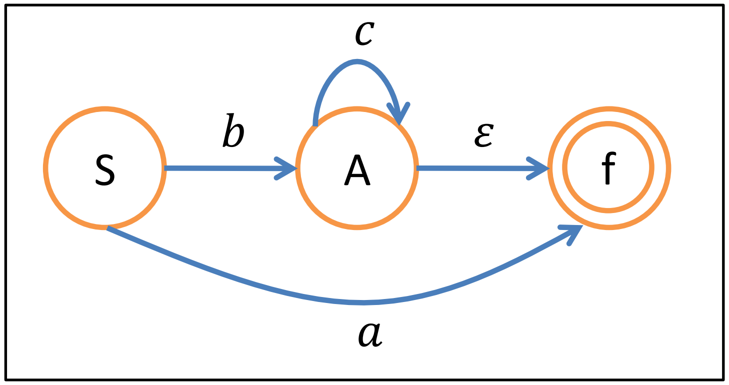
NGA reachability
(Earley)

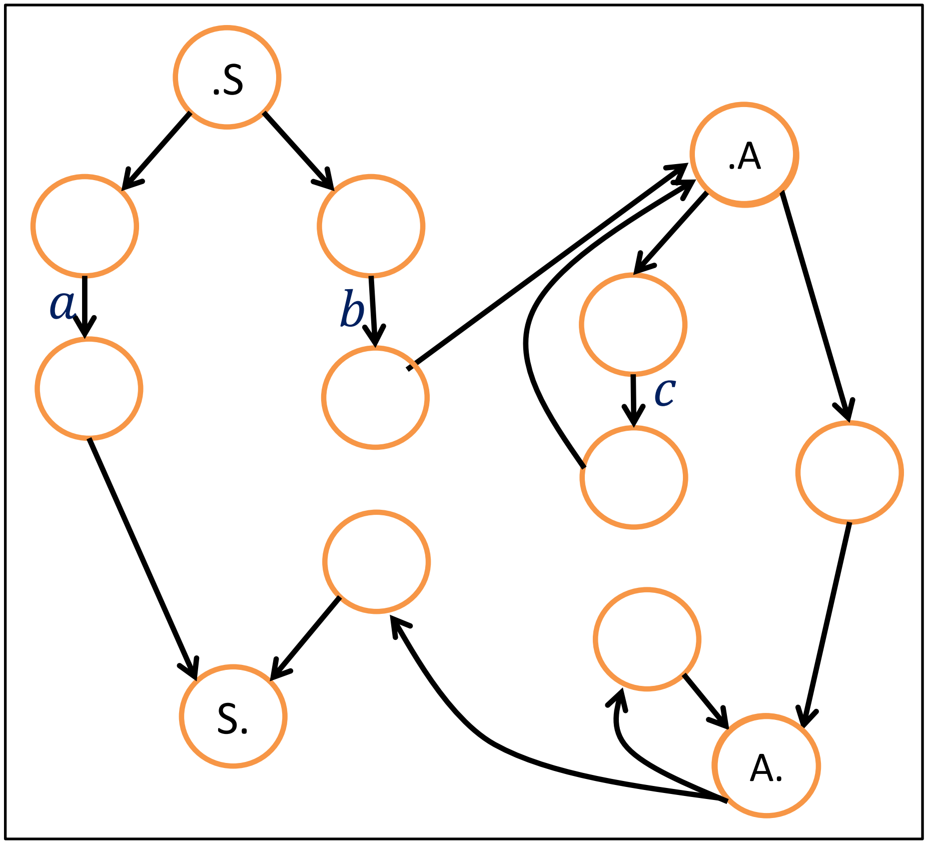
Simplifications of Earley's Algorithm

- Grammars for programming languages have certain properties.
  - Unambiguous
    - Each sentence in language is produced by one complete balanced path.
    - Caveat: May need to follow multiple parallel paths *during* parsing, but only one will survive at the end.
  - Parse tree is produced incrementally as string is read.
    - Distinct from ambiguity.
    - E.g., $A \rightarrow bAb|b$ is unambiguous but can't be incrementally parsed.
- Two important grammar classes
  - LL: parse tree can be produced incrementally in *pre-order*.
  - LR: parse tree can be produced incrementally in *post-order*.
- Parsers for LL and LR grammars are optimized versions of the Earley parser that exploit GFG structure to match calls and returns without using tags.
  - In particular, LL parsers (e.g., recursive-descent) need to follow just a single path through the GFG, and can therefore use the runtime stack to track return addresses.

Exploiting Structure: Regular Grammars

$$S \rightarrow a|bA$$
$$A \rightarrow cA|\varepsilon$$





- Tail-call optimization for NGA
  - If the last symbol in production is a non-terminal, call node does not have to push return node on stack (“replace recursion by iteration”).
  - This is the case for a right-linear grammar, so we can eliminate the stack.
- In this case, the GFG devolves to a NFA.