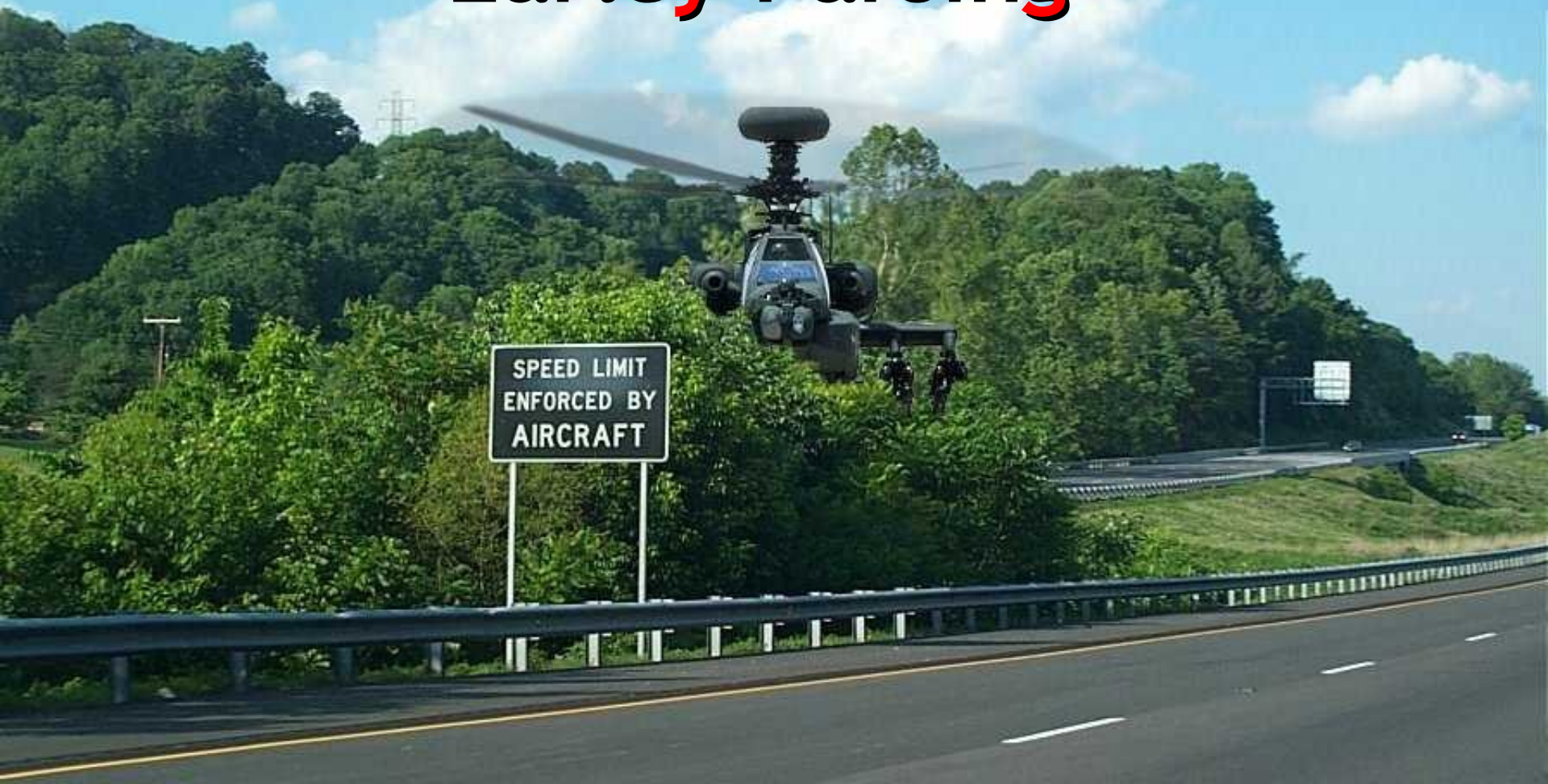


# Top-Down Parsing Earley Parsing



And it's gonna be totally awesome!





# Motivating Question

- Given this grammar G:
  - $E \rightarrow E + T$
  - $E \rightarrow T$
  - $T \rightarrow T * \text{int}$
  - $T \rightarrow \text{int}$
  - $T \rightarrow ( E )$
- Is the string  $\text{int} * (\text{int} + \text{int})$  in  $L(G)$ ?
  - Give a derivation or prove that it is not.



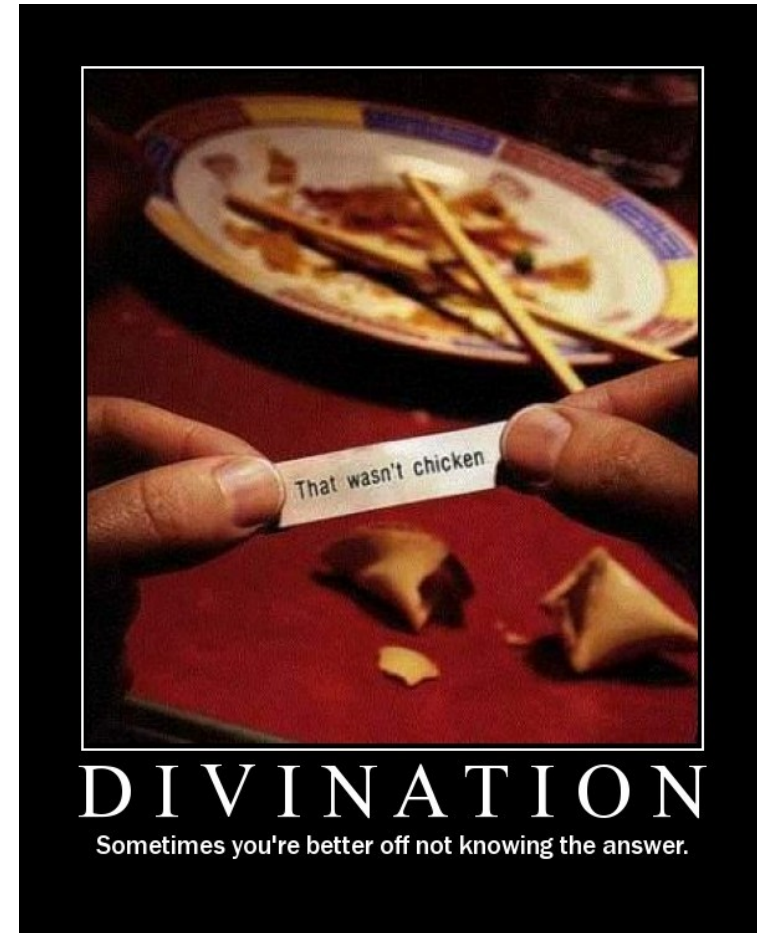
# Revenge of Theory

- How do we tell if DFA  $P$  is equal to DFA  $Q$ ?
  - We can do: “is DFA  $P$  empty?”
    - How?
  - We can do: “ $P := \text{not } Q$ ”
    - How?
  - We can do: “ $P := Q \text{ intersect } R$ ”
    - How?
  - So do: “is  $P \text{ intersect not } Q$  empty?”
- Does this work for CFG  $X$  and CFG  $Y$ ?
- Can we tell if  $s$  is in CFG  $X$ ?



# Outline

- Recursive Descent Parsing
  - Left Recursion
- Historical Approaches
  - LL, LR, LALR
- Dynamic Programming
- Earley's Algorithm
  - Chart States
  - Operations
  - Example

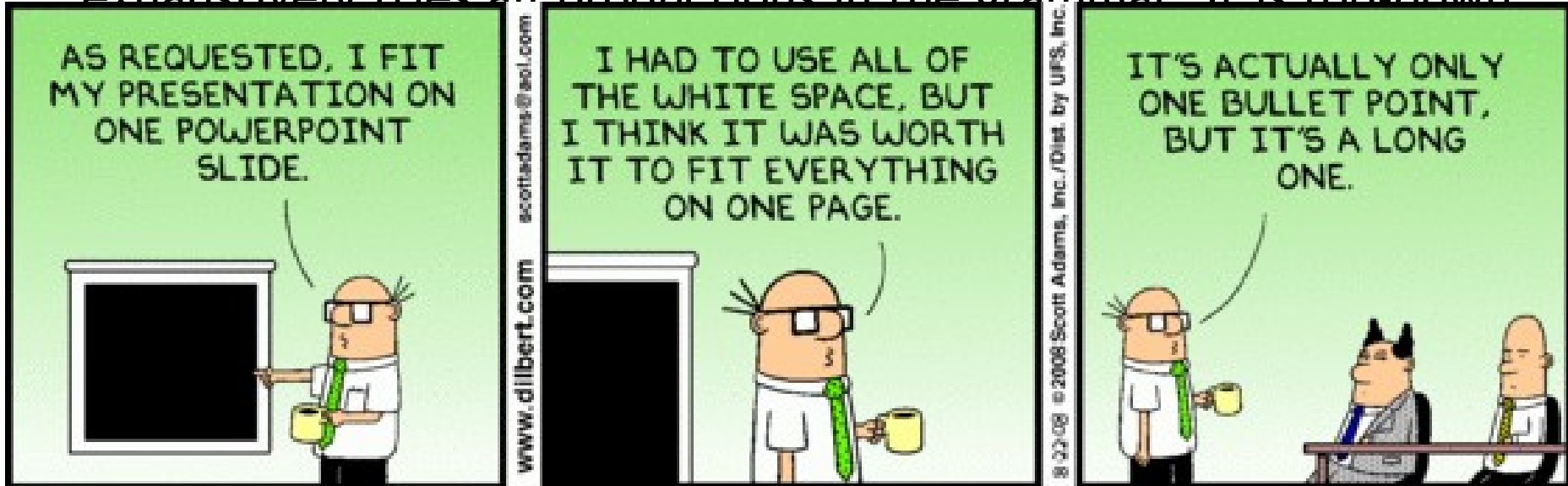


# In One Slide

- A **top-down parser** starts to work from the initial **grammar rules** (rather than from the initial tokens). A **recursive descent parser** exhaustively tries all productions in the grammar. It is top-down, may **backtrack** and cannot handle **left-recursive** grammars. Left recursion can be **eliminated**. Historical approaches such as **LL, LR and LALR** cannot handle all context-free grammars. They can be **efficient**.
- **Dynamic programming** is a problem-solving technique based on **optimal substructure**: solutions to subproblems yield solutions to overall problems.
- **Earley parsers** are top-down and use dynamic programming. An Earley **state** records incremental information: when we started, what has been seen so far, and what we expect to see. The Earley **chart** holds a set of states for each input position. **Shift, reduce** and **closure** operations fill in the chart.
- **You** enjoy parsing. Parsing is **easy** and **fun**.

# In One Slide

- A **top-down parser** starts to work from the initial **grammar rules** (rather than from the initial tokens). A **recursive descent parser** exhaustively tries all productions in the grammar. It is top-down



- **Earley parsers** are top-down and use dynamic programming. An Earley **state** records incremental information: when we started, what has been seen so far, and what we expect to see. The Earley **chart** holds a set of states for each input position. **Shift**, **reduce** and **closure** operations fill in the chart.
- **You** enjoy parsing. Parsing is **easy** and **fun**.

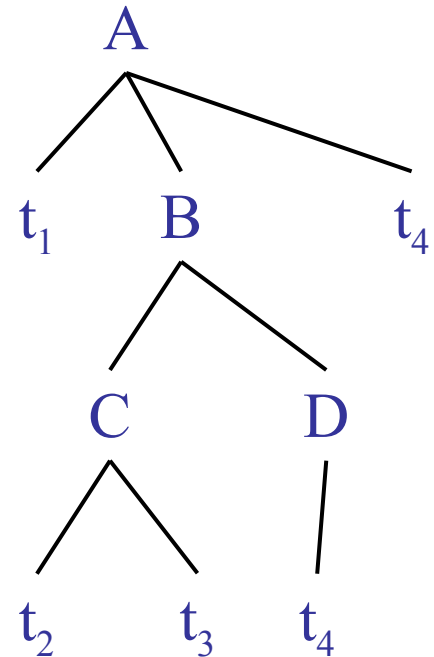
# Intro to Top-Down Parsing

- Terminals are seen in order of appearance in the token stream:

$t_1$   $t_2$   $t_3$   $t_4$   $t_5$

The parse tree is constructed

- From the top
- From left to right





# Recursive Descent Parsing

- We'll try **recursive descent** parsing first
  - “Try all productions exhaustively, backtrack”
- Consider the grammar

$$E \rightarrow T + E \mid T$$

$$T \rightarrow ( E ) \mid \text{int} \mid \text{int} * T$$

- Token stream is: **int \* int**
- Start with top-level non-terminal **E**
  
- Try the rules for **E** in order

# Recursive Descent Example

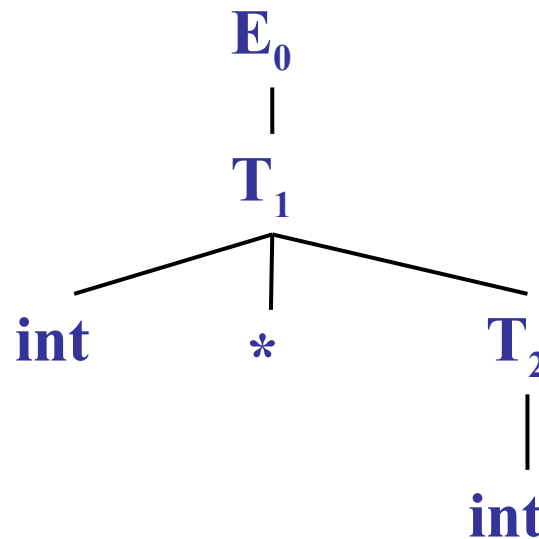
$E \rightarrow T + E \mid T$   
 $T \rightarrow ( E ) \mid \text{int} \mid \text{int} * T$   
*Input = int \* int*

- Try  $E_0 \rightarrow T_1 + E_2$
- Then try a rule for  $T_1 \rightarrow ( E_3 )$ 
  - But ( does not match input token *int*
- Try  $T_1 \rightarrow \text{int}$  . Token matches.
  - But + after  $T_1$  does not match input token \*
- Try  $T_1 \rightarrow \text{int} * T_2$ 
  - This will match but + after  $T_1$  will be unmatched
- Have exhausted the choices for  $T_1$ 
  - **Backtrack** to choice for  $E_0$

# Recursive Descent Example (2)

$E \rightarrow T + E \mid T$   
 $T \rightarrow ( E ) \mid \text{int} \mid \text{int} * T$   
*Input = int \* int*

- Try  $E_0 \rightarrow T_1$
- Follow same steps as before for  $T_1$ 
  - And succeed with  $T_1 \rightarrow \text{int} * T_2$  and  $T_2 \rightarrow \text{int}$
  - With the following parse tree



# Recursive Descent Parsing

- Parsing: given a string of tokens  $t_1 t_2 \dots t_n$ , find its parse tree
- **Recursive descent parsing**: Try all the productions exhaustively
  - At a given moment the **fringe** of the parse tree is:  $t_1 t_2 \dots t_k A \dots$
  - Try all the productions for A: if  $A \rightarrow BC$  is a production, the new fringe is  $t_1 t_2 \dots t_k B C \dots$
  - **Backtrack** if the fringe doesn't match the string
  - Stop when there are no more non-terminals



# When Recursive Descent Does *Not* Work

- Consider a production  $S \rightarrow S a$ :
  - In the process of parsing  $S$  we try the above rule
  - What goes wrong?



# When Recursive Descent Does *Not* Work

- Consider a production  $S \rightarrow S a$ :
  - In the process of parsing  $S$  we try the above rule
  - What goes wrong?
- A left-recursive grammar has
$$S \rightarrow^+ S\alpha \quad \text{for some } \alpha$$

Recursive descent does not work in such cases

- It goes into an infinite loop

# What's Wrong With That Picture?



# Elimination of Left Recursion

- Consider the left-recursive grammar

$$S \rightarrow S \alpha \mid \beta$$

- $S$  generates all strings starting with a  $\beta$  and followed by a number of  $\alpha$

- Can rewrite using **right-recursion**

$$S \rightarrow \beta T$$

$$T \rightarrow \alpha T \mid \varepsilon$$



# Example of Eliminating Left Recursion

- Consider the grammar

$$S \rightarrow 1 \mid S 0$$

$$(\beta = 1 \text{ and } \alpha = 0)$$

It can be rewritten as

$$S \rightarrow 1 T$$

$$T \rightarrow 0 T \mid \varepsilon$$



ASSASSIN

They come in all shapes and sizes.

# More Left Recursion Elimination

- In general

$$S \rightarrow S \alpha_1 \mid \dots \mid S \alpha_n \mid \beta_1 \mid \dots \mid \beta_m$$

- All strings derived from  $S$  start with one of  $\beta_1, \dots, \beta_m$  and continue with several instances of  $\alpha_1, \dots, \alpha_n$

- Rewrite as

$$S \rightarrow \beta_1 T \mid \dots \mid \beta_m T$$

$$T \rightarrow \alpha_1 T \mid \dots \mid \alpha_n T \mid \varepsilon$$

# General Left Recursion

- The grammar

$$S \rightarrow A \alpha \mid \delta$$

$$A \rightarrow S \beta$$

is also left-recursive because

$$S \rightarrow^+ S \beta \alpha$$

- This left-recursion can also be eliminated
- See book, Section 2.3
- Detecting and eliminating left recursion are *popular test questions*



Signs

And some of them are not

# Summary of Recursive Descent

- Simple and general parsing strategy
  - **Left-recursion** must be eliminated first
  - ... but that can be done automatically
- Unpopular because of **backtracking**
  - Thought to be too inefficient (repetition)
- We can avoid backtracking
  - Sometimes ...





# Sometimes Things Are Perfect

- The “.ml-lex” format you emit in PA2
- Will be the input for PA3
  - actually the *reference* “.ml-lex” will be used
- It can be “parsed” directly
  - You always know just what to do next
- Ditto with the “.ml-ast” output of PA3
- Just write a few mutually-recursive functions
- They read in the input, one line at a time

# Historical Approaches

- In the past, I/O was slow and memory was small. Many **sacrifices** were made to optimize parsing.
  - Basic idea: “If we don't handle all grammars, we can go faster on simpler grammars.” Also: table → no backtrack.
- **LL( $k$ )** - Left to right scan of input, Leftmost derivation, predict using  $k$  tokens. Parse by making a table.
- **LR( $k$ )** - Left to right scan of input, Rightmost derivation, predict using  $k$  tokens. Parse by making a table.
- **LALR( $k$ )** - Left to right scan of input, Rightmost derivation, predict using  $k$  tokens. Parse by making a table, but merge some states in that table. Yacc, bison, etc. use LALR(1).

# The Sacrifice

- **LL(1) languages** can be LL(1) parsed
  - A language  $Q$  is LL(1) if there exists an LL(1) table such the LL(1) parsing algorithm using that table accepts exactly the strings in  $Q$
  - Essentially, the table has to be perfect: no entry can be ambiguous or multiply defined.
- Sadly,  $LL(1) \neq CFG$ .
- Sadly,  $LR(1) \neq CFG$ .
- Sadly,  $LALR(1) \neq CFG$ .
  - See textbook for definitive Venn diagram.

# The Sacrifice

- **LL(1)**

- A language that is LL(1) is a subset of LR(1).
- Essentially, LR(1) is a superset of LL(1).
- Can be used to parse any language that is LR(1).



that  
entry

- Sadly,  $LL(1) \neq CFG$ .
- Sadly,  $LR(1) \neq CFG$ .
- Sadly,  $LALR(1) \neq CFG$ .
  - See textbook for definitive Venn diagram.



## Q: Books (727 / 842)

- Name 5 of the 9 major characters in A. A. Milne's 1926 books about a "*bear of very little brain*" who composes poetry and eats honey.

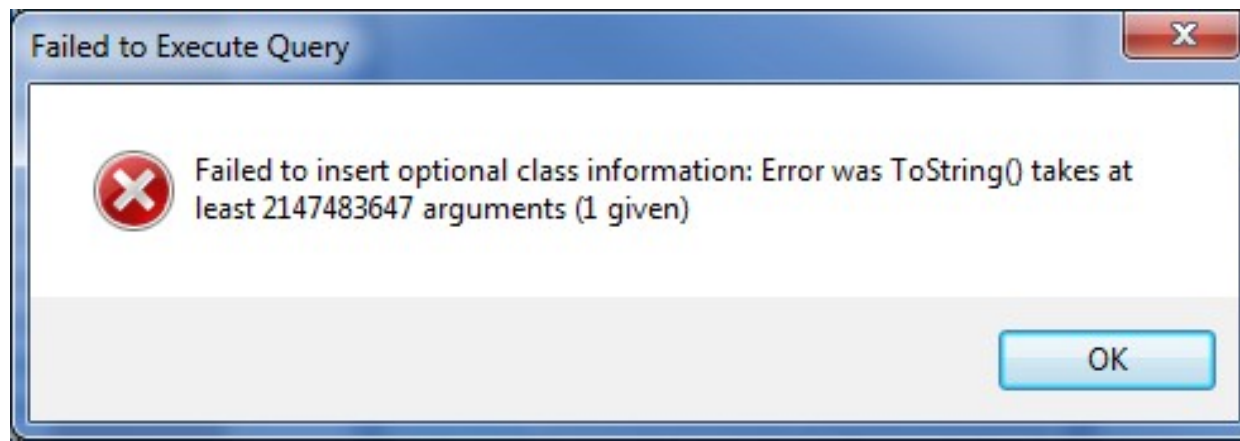
# Dynamic Programming

- “Program” = old word for “table of entries”
  - cf. the program (= “schedule”) at a concert
- **Dynamic Programming** means
  - “Fill in a chart or table at run-time.”
  - Same idea as **memoization**.
- Works when the problem has the **optimal substructure** property
  - Solution to Big Problem can be constructed from solutions to Smaller Subproblems.
- Shortest path, spell checking, ...

# Simple Dynamic Programming

- Dynamic Programming for Fibonacci

- N            1        2        3        4        5        6
- Chart        1        1        2        3        5        8
- $\text{chart}[N] = \text{chart}[N-1] + \text{chart}[N-2]$
- Reduces runtime of Fibon from Bad to Linear



# Dynamic Programming for Parsing

- Dynamic Programming for Fibonacci

- N            1        2        3        4        5        6
- Chart        1        1        2        3        5        8
- $\text{chart}[N] = \text{chart}[N-1] + \text{chart}[N-2]$
- Reduces runtime of Fibo from Bad to Linear

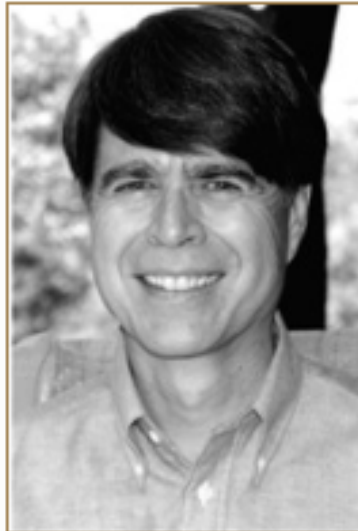
- Dynamic Programming for Parsing

- N            1        2        3        4        5        6
- Tokens        x        +        (        y        \*        z
- Chart        < *I'll explain in a minute* >
- $\text{chart}[N] =$  “list of possible parses of tokens 1...N”

# Earley's Algorithm

- **Earley's Algorithm** is an incremental left-to-right top-down parsing algorithm that works on arbitrary context-free grammars.
- Systematic Hypothesis Testing
  - Before processing token  $\#X$ , it has already considered all hypotheses consistent with tokens  $\#1$  to  $\#X-1$ .
  - Substructure: parsing token  $\#X$  is solved in terms of parsing sequences of tokens from  $\#1$  to  $\#X-1$

# About Jay



Jay Earley, Ph.D., is a transformational psychologist, group leader, psychotherapist, coach, author, teacher, and theorist.

Jay is trained in Internal Family Systems Therapy and assists with professional trainings in IFS. He leads IFS Classes for the general public which teach IFS as a practice for self-help and peer counseling. He is active in the IFS community and has presented a number of workshops at IFS annual conferences. He also teaches classes on Communication from the Heart, based on IFS, interactive groups, and the Pattern System.

He is nationally known for his innovation in the group psychotherapy field. His book, *Interactive Group Therapy: Integrating Interpersonal, Action-Oriented, and Psychodynamic Approaches*, Brunner/Mazel, describes his group therapy method in which people learn interpersonal relationship skills by working directly on their relationships with each other. During his ten years on the east coast, Jay was Director of the Group Therapy Center of Long Island, where he trained group therapists in this method. He has written a number of articles on interactive groups and made numerous presentations at regional and national psychotherapy conferences. He continues to lead interactive therapy groups in the Bay Area.

Jay offers Life Purpose Coaching and Change Agent Coaching, on finding your life purpose and making a difference in the world. He has been writing about and leading workshops on Life Purpose since 1984. He has collected his writings on life purpose into an ebook *Finding Your Life Purpose*.

• • •

Jay also has a Ph.D. in computer science from Carnegie-Mellon University and was formerly on the U.C. Berkeley faculty, where he published 12 computer science papers, one of which was voted one of the best 25 papers of the quarter century by the Communications of the A.C.M.

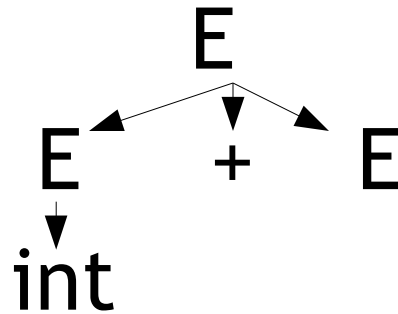


# Parsing State

- Earley is incremental and left-to-right
- Consider:  $E \rightarrow E + E \mid E * E \mid ( E ) \mid \text{int}$
- We use a “•” to mark “where we are now”
- Example:  $E \rightarrow E + \bullet E$ 
  - I am trying to parse  $E \rightarrow E + E$
  - I have already seen  $E +$  (before the dot)
  - I expect to see  $E$  (after the dot)
- General form:  $X \rightarrow a \bullet b$ 
  - $a, b$ : sequences of terminals and non-terminals

# Dot Example

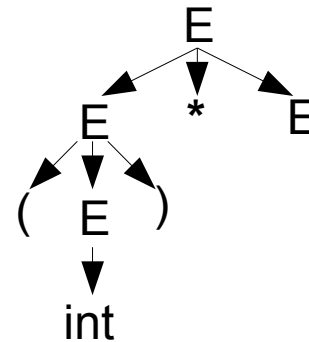
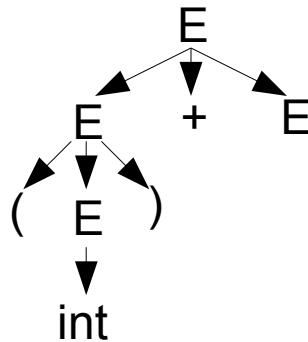
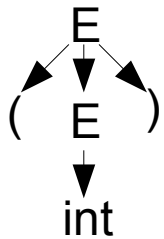
- $E \rightarrow E + E \mid E * E \mid ( E ) \mid \text{int}$



- Input so far: int +
- Current state:  $E \rightarrow E + \bullet E$

# Dotty Example

- $E \rightarrow E + E \mid E * E \mid ( E ) \mid \text{int}$
- Input so far:  $( \text{int} )$
- State could be:  $E \rightarrow ( E ) \bullet$



- But also:  $E \rightarrow E \bullet + E$
- Or:  $E \rightarrow E \bullet * E$

# Origin Positions

- $E \rightarrow E + E \mid E * E \mid ( E ) \mid \text{int}$
- Example Input-So-Far #1: int
  - Possible Current State:  $E \rightarrow \text{int} \bullet$
- Example Input-So-Far #2: int + int
  - Possible Current State:  $E \rightarrow \text{int} \bullet$
- Must track the input position just before the matching of this production began.

# Origin Positions

- $E \rightarrow E + E \mid E * E \mid ( E ) \mid \text{int}$
- Example Input-So-Far #1: int
  - Possible Current State:  $E \rightarrow \text{int} \bullet$ , from 0
- Example Input-So-Far #2: int + int
  - Possible Current State:  $E \rightarrow \text{int} \bullet$ , from 2
- Must track the input position just before the matching of this production began.

int                    +                    int  
0                    1                    2                    3

# Earley States

- Let  $X$  be a non-terminal
- Let  $a$  and  $b$  be (possibly-empty) sequences of terminals and non-terminals
- Let  $X \rightarrow ab$  be a production in your grammar
- Let  $j$  be a position in the input
- Each **Earley State** is a tuple  $\langle X \rightarrow a \bullet b, j \rangle$ 
  - We are currently parsing an  $X$
  - We have seen  $a$ , we expect to see  $b$
  - We started parsing this  $X$  after seeing the first  $j$  tokens from the input.



# Introducing: Parse Tables



## Rolemaster

A table for every occasion

# Earley Parse Table (= Chart)

- An **Earley parsing table** (or **chart**) is a one-dimensional array. Each array element is a **set** of Earley states.
  - **chart[i]** holds the set of valid parsing states we could be in after seeing the first  $i$  input tokens

# Earley Parse Table (= Chart)

- An **Earley parsing table** (or **chart**) is a one-dimensional array. Each array element is a **set** of Earley states.
  - **chart[i]** holds the set of valid parsing states we could be in after seeing the first  $i$  input tokens
- Then the string  $\text{tok}_1 \dots \text{tok}_n$  is in the language of a grammar with start symbol  $S$  *iff*
  - **chart[n]** contains  $\langle S \rightarrow ab \bullet, 0 \rangle$  for some production rule  $S \rightarrow ab$  in the grammar.
  - We then say the parser **accepts** the string.

# Earley Parsing Algorithm

- Input:

- Grammar G (start symbol S, productions  $X \rightarrow ab$ )
- Input Tokens  $\text{tok}_1 \dots \text{tok}_n$

- Work:

$\text{chart}[0] = \{ \langle S \rightarrow \bullet ab, 0 \rangle \}$

for  $i = 0$  to  $n$

complete  $\text{chart}[i]$  using G and  $\text{chart}[0] \dots \text{chart}[i]$

- Output:

- true iff  $\langle S \rightarrow ab\bullet, 0 \rangle$  in  $\text{chart}[n]$

# Filling In The Chart

- Three operations build up  $\text{chart}[n]$
- The first is called **shift** or **scan**.
  - It corresponds to “seeing the next expected token” or “helping to confirm the current hypothesis” or “we're winning”.
- Example:
  - $\text{chart}[1]$  contains  $\langle E \rightarrow E \bullet + E, 0 \rangle$
  - 2<sup>nd</sup> token is “+”
  - Then put  $\langle E \rightarrow E + \bullet E, 0 \rangle$  in  $\text{chart}[2]$

# Formal shift operation

- Whenever
  - chart[i] contains  $\langle X \rightarrow ab \bullet cd, j \rangle$
  - **c** is a terminal (*not* a non-terminal)
  - the  $(i+1)^{\text{th}}$  input token is **c**
- The **shift** operation
  - Adds  $\langle X \rightarrow abc \bullet d, j \rangle$  to chart[i+1]



# Filling In The Chart (2)

- The second operation is the **closure** or **predictor**.
  - It corresponds to “expanding rewrite rules” or “substituting in the definitions of non-terminals”
- Suppose the grammar is:  
$$S \rightarrow E \quad E \rightarrow E + E \mid E - E \mid \text{int}$$
- If chart[0] has  $\langle S \rightarrow \bullet E, 0 \rangle$  then add
  - $\langle E \rightarrow \bullet E + E, 0 \rangle$
  - $\langle E \rightarrow \bullet E - E, 0 \rangle$
  - $\langle E \rightarrow \bullet \text{int}, 0 \rangle$

# Formal closure operation

- Whenever
  - chart[i] contains  $\langle X \rightarrow ab \bullet cd, j \rangle$
  - $c$  is a non-terminal
  - The grammar contains  $\langle c \rightarrow pqr \rangle$
- The **closure** operation
  - Adds  $\langle c \rightarrow \bullet pqr, i \rangle$  to chart[i]
  - Note  $\langle c \rightarrow \bullet pqr, \underline{i} \rangle$  because “we started parsing this  $c$  after seeing the first  $i$  tokens from the input.”

# Filling In The Chart (3)

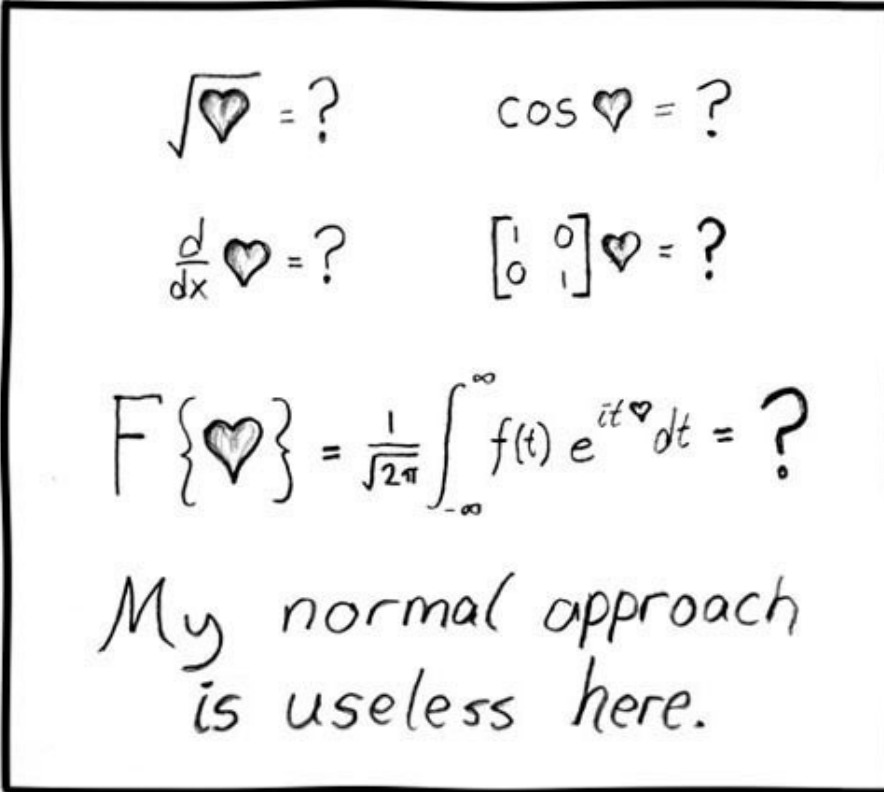
- The third operation is **reduction** or **completion**.
  - It corresponds to “finishing a grammar rewrite rule” or “being done parsing a non-terminal” or “doing a rewrite rule in reverse and then shifting over the non-terminal”.
- Suppose:
  - $E \rightarrow \text{int} \mid E + E \mid E - E \mid ( E )$ , input is “( int”
  - chart[2] contains  $\langle E \rightarrow \text{int} \bullet, 1 \rangle$
  - chart[1] contains  $\langle E \rightarrow ( \bullet E ), 0 \rangle$
  - Then chart[2] +=  $\langle E \rightarrow ( E \bullet ), 0 \rangle$

# Formal reduce operation

- Whenever
  - chart[i] contains  $\langle X \rightarrow ab \cdot , j \rangle$   
*(The dot must be all the way to the right!)*
  - chart[j] contains  $\langle Y \rightarrow q \cdot X r , k \rangle$
- The **reduce** operation
  - Adds  $\langle Y \rightarrow q X \cdot r , k \rangle$  to chart[i]
  - Note  $\langle Y \rightarrow q X \cdot r , \underline{k} \rangle$  because “we started parsing this  $Y$  after seeing the first  $k$  tokens from the input.”

# This is easy and fun.

- This is not as hard as it may seem.



A rectangular box containing handwritten mathematical expressions. The expressions are arranged in three rows. The first row contains two equations:  $\sqrt{\heartsuit} = ?$  and  $\cos \heartsuit = ?$ . The second row contains two equations:  $\frac{d}{dx} \heartsuit = ?$  and  $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \heartsuit = ?$ . The third row contains a single equation:  $F\{\heartsuit\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{it\heartsuit} dt = ?$ . Below the equations, the text "My normal approach is useless here." is written in a cursive-like hand.

$$\sqrt{\heartsuit} = ? \quad \cos \heartsuit = ?$$
$$\frac{d}{dx} \heartsuit = ? \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \heartsuit = ?$$
$$F\{\heartsuit\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{it\heartsuit} dt = ?$$

My normal approach  
is useless here.

- Let's go practice it!

# Shift Practice

- chart[3] contains

$\langle S \rightarrow E \cdot , 0 \rangle$

$\langle E \rightarrow E \cdot + E , 0 \rangle$

$\langle E \rightarrow E \cdot - E , 2 \rangle$

$\langle E \rightarrow \text{int} \cdot , 2 \rangle$

$\langle E \rightarrow E \cdot - E , 0 \rangle$

$\langle E \rightarrow E - E \cdot , 0 \rangle$

$\langle E \rightarrow E \cdot + E , 2 \rangle$

- The 4<sup>th</sup> token is “+”. What does **shift** bring in?



# Shift Practice

- chart[3] contains

< S → E • , 0 >

< E → E • - E , 0 >

< E → E • + E , 0 >

< E → E - E • , 0 >

< E → E • - E , 2 >

< E → E • + E , 2 >

< E → int • , 2 >

- The 4<sup>th</sup> token is “+”. What does **shift** bring in?

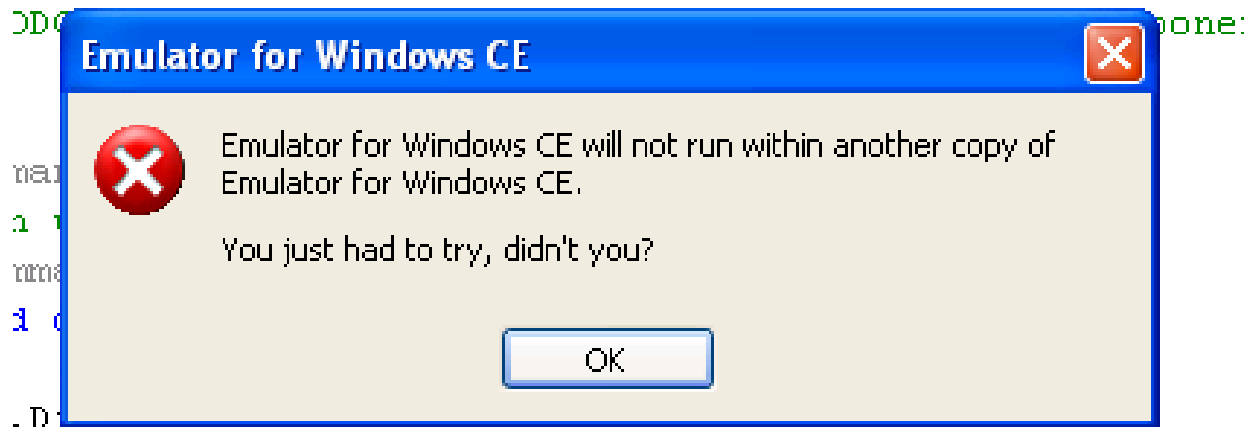
< E → E + • E , 0 >

< E → E + • E , 2 >

... are both added to chart[4].

# Closure Practice

- Grammar is
  - $S \rightarrow E$        $E \rightarrow E + E \mid E - E \mid ( E ) \mid \text{int}$
- `chart[4]` contains:
  - $\langle E \rightarrow E + \bullet E, 0 \rangle \langle E \rightarrow E + \bullet E, 2 \rangle$
- What does the **closure** operation bring in?



# Closure Practice

- Grammar is
  - $S \rightarrow E$        $E \rightarrow E + E \mid E - E \mid ( E ) \mid \text{int}$
- $\text{chart}[4]$  contains:
  - $\langle E \rightarrow E + \bullet E, 0 \rangle$   $\langle E \rightarrow E + \bullet E, 2 \rangle$
- What does the **closure** operation bring in?
  - $\langle E \rightarrow \bullet E + E, 4 \rangle$   $\langle E \rightarrow \bullet E - E, 4 \rangle$
  - $\langle E \rightarrow \bullet ( E ), 4 \rangle$        $\langle E \rightarrow \bullet \text{int}, 4 \rangle$
  - ... are all added to  $\text{chart}[4]$ .

# Reduction Practice

- `chart[4]` contains:
  - $\langle E \rightarrow E + \bullet E, 0 \rangle$   $\langle E \rightarrow E + \bullet E, 2 \rangle$
  - $\langle E \rightarrow \bullet E + E, 4 \rangle$   $\langle E \rightarrow \bullet E - E, 4 \rangle$
  - $\langle E \rightarrow \bullet ( E ) , 4 \rangle$   $\langle E \rightarrow \bullet \text{int} , 4 \rangle$
- `chart[5]` contains:
  - $\langle E \rightarrow \text{int} \bullet , 4 \rangle$
- What does the **reduce** operator bring in?



# Reduction Practice

- chart[4] contains:

$\langle E \rightarrow E + \bullet E, 0 \rangle \langle E \rightarrow E + \bullet E, 2 \rangle$

$\langle E \rightarrow \bullet E + E, 4 \rangle \langle E \rightarrow \bullet E - E, 4 \rangle$

$\langle E \rightarrow \bullet ( E ) , 4 \rangle \quad \langle E \rightarrow \bullet \text{int} , 4 \rangle$

- chart[5] contains:

-  $\langle E \rightarrow \text{int} \bullet , 4 \rangle$

- What does the **reduce** operator bring in?

$\langle E \rightarrow E + E \bullet , 0 \rangle \langle E \rightarrow E + E \bullet , 2 \rangle$

$\langle E \rightarrow E \bullet + E , 4 \rangle \langle E \rightarrow E \bullet - E , 4 \rangle$

- ... are all added to chart[5]. (Plus more in a bit!)

# Earley Parsing Algorithm

- Input: CFG  $G$ , Tokens  $\text{tok}_1 \dots \text{tok}_n$
- Work:
  - $\text{chart}[0] = \{ \langle S \rightarrow \bullet ab, 0 \rangle \}$
  - for  $i = 0$  to  $n$ 
    - repeat
      - use shift, reduce and closure on  $\text{chart}[i]$
      - until no new states are added
- Output:
  - true iff  $\langle S \rightarrow ab\bullet, 0 \rangle$  in  $\text{chart}[n]$

# Massive Earley Example

## Grammar

$S \rightarrow F$

$F \rightarrow id ( A )$

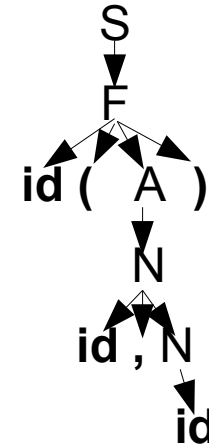
$A \rightarrow N$

$A \rightarrow \epsilon$

$N \rightarrow id$

$N \rightarrow id , N$

Input  
id ( id , id )

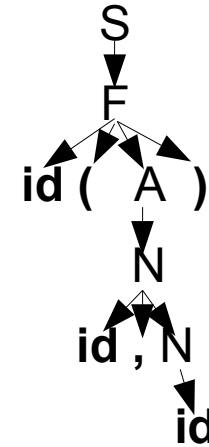


# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
id ( id , id )



chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
id		(	id	,	id	)



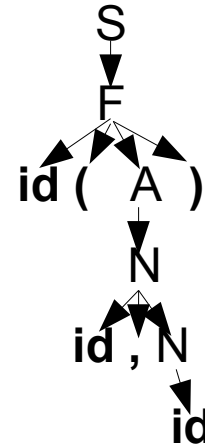
# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

## Input

id ( id , id )



id

(

id

)

chart[0]

chart[1]

chart[2]

chart[5]

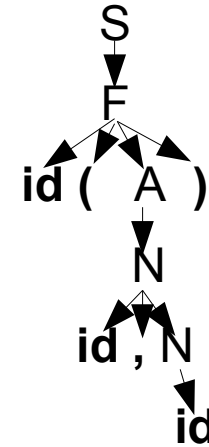
chart[6]

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
id ( id , id )



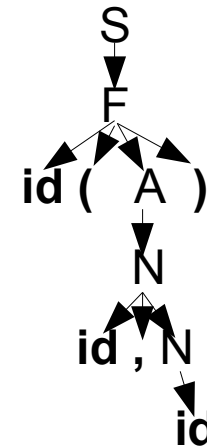
	id	(	id	,	id	)
chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$						

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



	id	(	id	,	id	)
chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$						
$F \rightarrow \bullet id ( A ) , 0$						

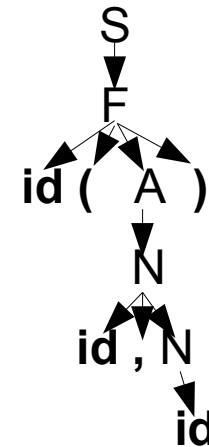
Closure on F

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
	id	(	id	,	id	)
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$					
$F \rightarrow \bullet id ( A ) , 0$						

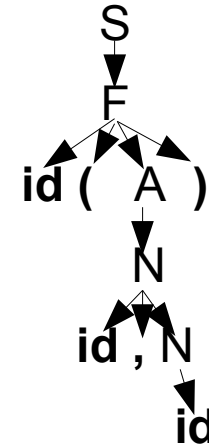
Shift on "id"

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



id	(	id	,	id	)	
chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$				
$F \rightarrow \bullet id ( A ) , 0$						

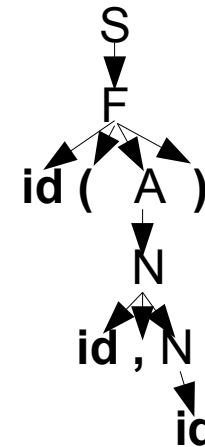
Shift on "("

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



id ( id , id )

chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$				
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$				
		$A \rightarrow \bullet , 2$				

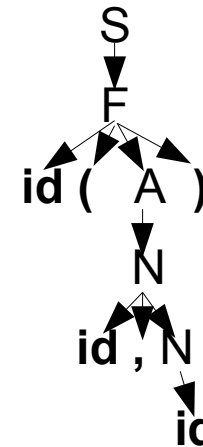
Closure on A

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



id	(	id	,	id	)	
chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$				
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$				
		$A \rightarrow \bullet , 2$				
		$N \rightarrow \bullet id , 2$				
		$N \rightarrow \bullet id , N , 2$				

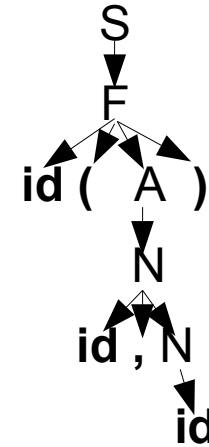
Closure on N

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$				
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$				
		$A \rightarrow \bullet , 2$				
		$N \rightarrow \bullet id , 2$				
		$N \rightarrow \bullet id , N , 2$				
		$F \rightarrow id ( A \bullet ) , 0$				

Reduce on A

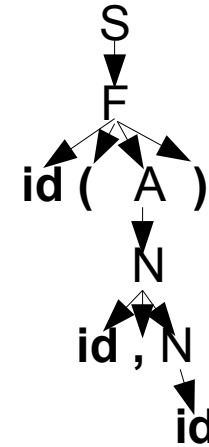


# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$	$N \rightarrow id \bullet , 2$			
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$	$N \rightarrow id \bullet , N , 2$			
		$A \rightarrow \bullet , 2$				
		$N \rightarrow \bullet id , 2$				
		$N \rightarrow \bullet id , N , 2$				
		$F \rightarrow id ( A \bullet ) , 0$				

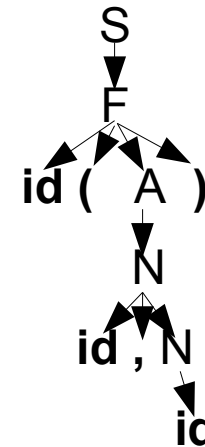
Shift on "id"

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$	$N \rightarrow id \bullet , 2$			
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$	$N \rightarrow id \bullet , N , 2$			
		$A \rightarrow \bullet , 2$	$A \rightarrow N \bullet , 2$			
		$N \rightarrow \bullet id , 2$				
		$N \rightarrow \bullet id , N , 2$				
		$F \rightarrow id ( A \bullet ) , 0$				

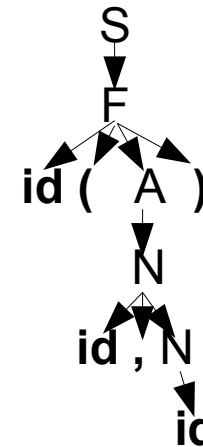
Reduce on N

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$	$N \rightarrow id \bullet , 2$			
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$	$N \rightarrow id \bullet , N , 2$			
		$A \rightarrow \bullet , 2$	$A \rightarrow N \bullet , 2$			
		$N \rightarrow \bullet id , 2$	$F \rightarrow id ( A \bullet ) , 0$			
		$N \rightarrow \bullet id , N , 2$				
		$F \rightarrow id ( A \bullet ) , 0$				

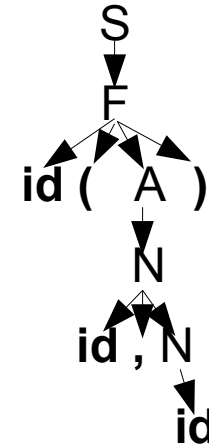
Reduce on A

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



	id	(	id	,	id	)
chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$	$N \rightarrow id \bullet , 2$	$N \rightarrow id , \bullet N , 2$		
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$	$N \rightarrow id \bullet , N , 2$			
		$A \rightarrow \bullet , 2$	$A \rightarrow N \bullet , 2$			
		$N \rightarrow \bullet id , 2$	$F \rightarrow id ( A \bullet ) , 0$			
		$N \rightarrow \bullet id , N , 2$				
		$F \rightarrow id ( A \bullet ) , 0$				

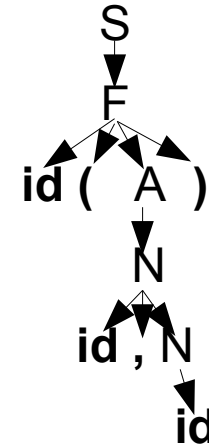
Shift on “,”

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



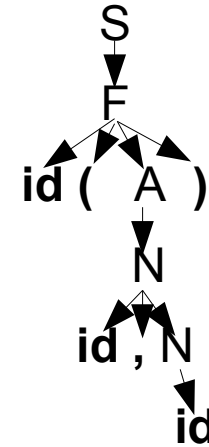
chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$	$N \rightarrow id \bullet , 2$	$N \rightarrow id , \bullet N , 2$		
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$	$N \rightarrow id \bullet , N , 2$	$N \rightarrow \bullet id , 4$		
		$A \rightarrow \bullet , 2$	$A \rightarrow N \bullet , 2$	$N \rightarrow \bullet id , N , 4$		
		$N \rightarrow \bullet id , 2$	$F \rightarrow id ( A \bullet ) , 0$			
		$N \rightarrow \bullet id , N , 2$				
		$F \rightarrow id ( A \bullet ) , 0$				

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



id ( id , id )

chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$	$N \rightarrow id \bullet , 2$	$N \rightarrow id , \bullet N , 2$	$N \rightarrow id \bullet , 4$	
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$	$N \rightarrow id \bullet , N , 2$	$N \rightarrow \bullet id , 4$	$N \rightarrow id \bullet , N , 4$	
		$A \rightarrow \bullet , 2$	$A \rightarrow N \bullet , 2$	$N \rightarrow \bullet id , N , 4$		
		$N \rightarrow \bullet id , 2$	$F \rightarrow id ( A \bullet ) , 0$			
		$N \rightarrow \bullet id , N , 2$				
		$F \rightarrow id ( A \bullet ) , 0$				

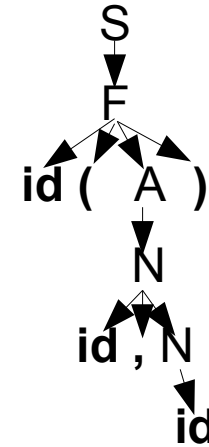
Shift on "id"

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



	id	(	id	,	id	)
chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$	$N \rightarrow id \bullet , 2$	$N \rightarrow id , \bullet N , 2$	$N \rightarrow id \bullet , 4$	
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$	$N \rightarrow id \bullet , N , 2$	$N \rightarrow \bullet id , 4$	$N \rightarrow id \bullet , N , 4$	
		$A \rightarrow \bullet , 2$	$A \rightarrow N \bullet , 2$	$N \rightarrow \bullet id , N , 4$	$N \rightarrow id , N \bullet , 2$	
		$N \rightarrow \bullet id , 2$	$F \rightarrow id ( A \bullet ) , 0$			
		$N \rightarrow \bullet id , N , 2$				
		$F \rightarrow id ( A \bullet ) , 0$				

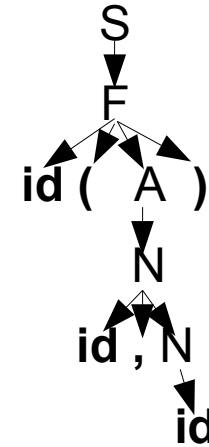
Reduce on N

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$	$N \rightarrow id \bullet , 2$	$N \rightarrow id , \bullet N , 2$	$N \rightarrow id \bullet , 4$	
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$	$N \rightarrow id \bullet , N , 2$	$N \rightarrow \bullet id , 4$	$N \rightarrow id \bullet , N , 4$	
		$A \rightarrow \bullet , 2$	$A \rightarrow N \bullet , 2$	$N \rightarrow \bullet id , N , 4$	$N \rightarrow id , N \bullet , 2$	
		$N \rightarrow \bullet id , 2$	$F \rightarrow id ( A \bullet ) , 0$		$A \rightarrow N \bullet , 2$	
		$N \rightarrow \bullet id , N , 2$				
		$F \rightarrow id ( A \bullet ) , 0$				

Reduce on N

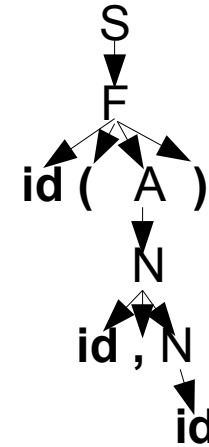


# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



	id	(	id	,	id	)
chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$	$N \rightarrow id \bullet , 2$	$N \rightarrow id , \bullet N , 2$	$N \rightarrow id \bullet , 4$	
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$	$N \rightarrow id \bullet , N , 2$	$N \rightarrow \bullet id , 4$	$N \rightarrow id \bullet , N , 4$	
		$A \rightarrow \bullet , 2$	$A \rightarrow N \bullet , 2$	$N \rightarrow \bullet id , N , 4$	$N \rightarrow id , N \bullet , 2$	
		$N \rightarrow \bullet id , 2$	$F \rightarrow id ( A \bullet ) , 0$		$A \rightarrow N \bullet , 2$	
		$N \rightarrow \bullet id , N , 2$			$F \rightarrow id ( A \bullet ) , 0$	
		$F \rightarrow id ( A \bullet ) , 0$				

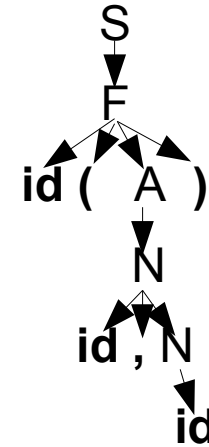
Reduce on A

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



id ( id , id )

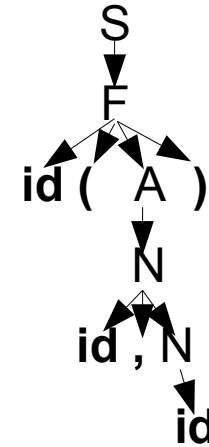
chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$	$N \rightarrow id \bullet , 2$	$N \rightarrow id , \bullet N , 2$	$N \rightarrow id \bullet , 4$	$F \rightarrow id ( A ) \bullet , 0$
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$	$N \rightarrow id \bullet , N , 2$	$N \rightarrow \bullet id , 4$	$N \rightarrow id \bullet , N , 4$	
		$A \rightarrow \bullet , 2$	$A \rightarrow N \bullet , 2$	$N \rightarrow \bullet id , N , 4$	$N \rightarrow id , N \bullet , 2$	
		$N \rightarrow \bullet id , 2$	$F \rightarrow id ( A \bullet ) , 0$		$A \rightarrow N \bullet , 2$	
		$N \rightarrow \bullet id , N , 2$			$F \rightarrow id ( A \bullet ) , 0$	
		$F \rightarrow id ( A \bullet ) , 0$				

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$	$N \rightarrow id \bullet , 2$	$N \rightarrow id , \bullet N , 2$	$N \rightarrow id \bullet , 4$	$F \rightarrow id ( A ) \bullet , 0$
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$	$N \rightarrow id \bullet , N , 2$	$N \rightarrow \bullet id , 4$	$N \rightarrow id \bullet , N , 4$	$S \rightarrow F \bullet , 0$
		$A \rightarrow \bullet , 2$	$A \rightarrow N \bullet , 2$	$N \rightarrow \bullet id , N , 4$	$N \rightarrow id , N \bullet , 2$	
		$N \rightarrow \bullet id , 2$	$F \rightarrow id ( A \bullet ) , 0$		$A \rightarrow N \bullet , 2$	
		$N \rightarrow \bullet id , N , 2$			$F \rightarrow id ( A \bullet ) , 0$	
		$F \rightarrow id ( A \bullet ) , 0$				

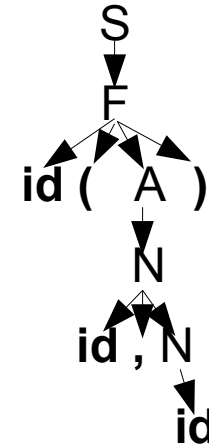
Reduce on F

# Massive Earley Example

## Grammar

$S \rightarrow F$   
 $F \rightarrow id ( A )$   
 $A \rightarrow N$   
 $A \rightarrow \epsilon$   
 $N \rightarrow id$   
 $N \rightarrow id , N$

Input  
 $id ( id , id )$



	id	(	id	,	id	)
chart[0]	chart[1]	chart[2]	chart[3]	chart[4]	chart[5]	chart[6]
$S \rightarrow \bullet F , 0$	$F \rightarrow id \bullet ( A ) , 0$	$F \rightarrow id ( \bullet A ) , 0$	$N \rightarrow id \bullet , 2$	$N \rightarrow id , \bullet N , 2$	$N \rightarrow id \bullet , 4$	$F \rightarrow id ( A ) \bullet , 0$
$F \rightarrow \bullet id ( A ) , 0$		$A \rightarrow \bullet N , 2$	$N \rightarrow id \bullet , N , 2$	$N \rightarrow \bullet id , 4$	$N \rightarrow id \bullet , N , 4$	$S \rightarrow F \bullet , 0$
		$A \rightarrow \bullet , 2$	$A \rightarrow N \bullet , 2$	$N \rightarrow \bullet id , N , 4$	$N \rightarrow id , N \bullet , 2$	
		$N \rightarrow \bullet id , 2$	$F \rightarrow id ( A \bullet ) , 0$		$A \rightarrow N \bullet , 2$	
		$N \rightarrow \bullet id , N , 2$			$F \rightarrow id ( A \bullet ) , 0$	
		$F \rightarrow id ( A \bullet ) , 0$				



# Homework

- WA2 (written homework) due Monday
- Read Chapter 2.3.3, etc.
- Read earley.py
- Midterm 1 Next Wednesday