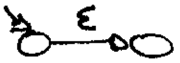
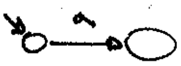
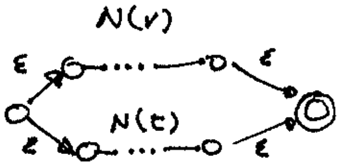
THOMPSON'S CONSTRUCTION

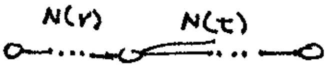
BASE

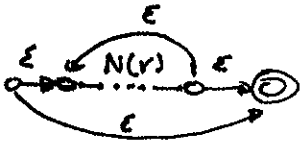
- ε 

- a 

STEP

- $r | t$ 

- $r \cdot s$ 

- r^* 

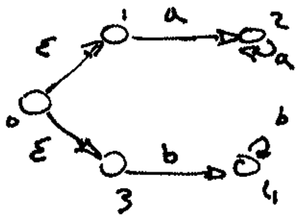
NFA TO DFA

```
INIT:= $\varepsilon$-CLOSURE ({$S_0$})
DSTATES:= {$\varepsilon$-CLOSURE ($S_0$)} .   # $\varepsilon$-CLOSURE ($S_0$) IS NOT MARKED

(REPEAT)
WHILE $\exists T \in$ DSTATES, T ISN'T MARKED:
    MARK T
    FOREACH $a \in \mathcal{A}$:
        U:= $\varepsilon$-closure (MOVE(T,a))
        IF U $\notin$ DSTATES:
            UNMARK (U)
            DSTATES:= DSTATES $\cup$ {U}
        DTRANS [T,a]:= U
```

FINAL STATES := $T \in$ DSTATES, T CONTAINS AT LEAST A FINAL STATE OF THE NFA

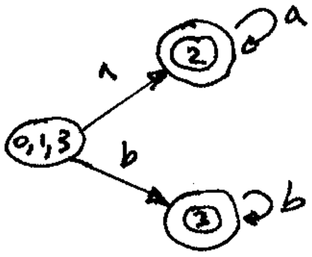
EXAMPLE



→

DSTATES	a	b
→ {0,1,3}	{2}	{4}
{2}	{2}	/
{4}	/	{4}

→



DFA MINIMIZATION (PARTITION REFINEMENT ALGORITHM)

ASSERT: THE DFA TRANSITION FUNCTION IS TOTAL

DEF: GIVEN A DFA WITH A TOTAL TRANSITION FUNCTION δ, A PARTITION $\mathbb{P}$ OF ITS STATES IS REFINABLE IF:

$$\exists\, g, g_1, g_2 \in \mathbb{P} : g_1 \neq g_2$$

$$\exists\, s_1, s_2 \in g\,,\ a \in \mathcal{A} : \begin{cases} \delta(s_1, a) \in g_1 \\ \delta(s_2, a) \in g_2 \end{cases}$$

ONE-STEP REFINEMENT

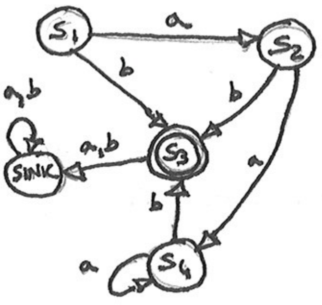
- CHOOSE $g, g_1, g_2 \in \mathbb{P}\,,\ a \in \mathcal{A}$
- SPLIT g INTO THE LARGEST SUBSETS $\hat{g}_j$ SUCH THAT

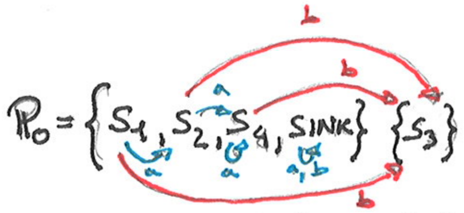
$$\forall j,\ \forall g' \in \mathbb{P},\ \forall \hat{s}_1, \hat{s}_2 \in \hat{g}_j\,,\ \delta(\hat{s}_1, a) \in g' \text{ ~~if~~ iff } \delta(\hat{s}_2, a) \in g'$$

REFINEMENT PROCEDURE

```
P := { FINAL_STATES, STATES - FINAL_STATES }
WHILE P IS REFINABLE:
    APPLY ONE-STEP_REFINEMENT
    UPDATE P
```

EXAMPLE

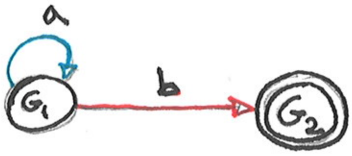




$$\mathbb{P}_0 = \{S_1, S_2, S_4, SINK\}\ \{S_3\}$$

$$\mathbb{P}_1 = \{S_1, S_2, S_4\}\ \{SINK\}\ \{S_3\}$$

$$G_1 = \{S_1, S_2, S_4\}\ ,\quad G_2 = \{S_3\}$$



GRAMMAR TO DFA

STATES := NON-TERMINALS

TRANSITIONS := PRODUCTIONS

FINAL STATES := STATES THAT ACCEPT ε AS INPUT

$S \rightarrow aA \mid bB$ → go to state B with input b

↓ go to state A with input a

PUMPING LEMMA FOR REGULAR LANGUAGES

LET $\mathcal{L}$ BE A REGULAR LANGUAGE.

THEN $\exists\, p>0$ S.T. $\forall z \in \mathcal{L} : |z| \geq p$, $\exists\, u,v,w$ S.T.

1) $z = uvw$
2) $|uv| \leq p$
3) $|v| > 0$ $(v \neq \varepsilon)$
4) $\forall k \geq 0,\ uv^k w \in \mathcal{L}$

PROOF

BY $\mathcal{L}$ REGULAR, $\exists$ DFA WITH n STATES. CONSIDER $w \in \mathcal{L} : |w| = m \geq n$, $w = a_1 a_2 \dots a_m$

$\forall i \in \{0,1,\dots,n\}$ LET $p_i = \hat{\delta}(q_0, a_1 a_2 \dots a_i)$ WHERE q_0 IS THE START STATE

$\Rightarrow$ (PIDGEONHOLE PRINCIPLE) THE $(n+1)$ p_i'S CANNOT BE DISTINCT HAVING ONLY n STATES

$\Rightarrow \exists\, i,j,\ 0 \leq i < j \leq n : p_i = p_j$. CONSIDER UVZ AS $\begin{cases} x = a_1 a_2 \dots a_i \\ v = a_{i+1} a_{i+2} \dots a_j \\ z = a_{j+1} a_{j+2} \dots a_m \end{cases}$

IF $i=0$: x = EMPTY, IF $j=n=m$: z = EMPTY (y CANNOT BE EMPTY SINCE $i<j$)

TAKE $uv^k w$ AS INPUT FOR THE DFA. THEN:

IF $K=0$: THE DFA GOES FROM p_0 TO p_i WITH u. SINCE $p_i = p_j$, IT GOES FROM p_i TO AN ACCEPTING STATE WITH w

$\Rightarrow$ THE DFA ACCEPTS uw

IF $K>0$: THE DFA GOES FROM p_0 TO p_i WITH u, ~~FROM~~ CIRCLES FROM p_i TO p_i K TIMES WITH v^k AND FROM p_i TO ACCEPTING STATE WITH w

$\Rightarrow$ THE DFA ACCEPTS $uv^k w$ $\forall K > 0$

$\Rightarrow$ THE DFA ACCEPTS $uv^k w$ $\forall K \geq 0 \Rightarrow uv^k w \in \mathcal{L}$

▨

USAGE

ASSUME $\mathcal{L}$ IS REGULAR, SHOW THESIS IS FALSE

$$\forall p \in \mathbb{N}^+, \exists z \in \mathcal{L}, |z| \geq p : \forall uvw \text{ S.T. } z = uvw, |uv| \leq p, |v| > 0 \Rightarrow \exists\, i \geq 0 .\ uv^i w \notin \mathcal{L}$$

PUMPING LEMMA FOR CONTEXT-FREE LANGUAGES

LET $\mathcal{L}$ BE A CONTEXT-FREE LANGUAGE

THEN $\exists\ p \in \mathbb{N}^+$ SUCH THAT, $\forall z \in \mathcal{L} : |z| > p$,

$\exists$ uvwxy SUCH THAT

1) $z = uvwxy$
2) $|vwx| \leq p$
3) $|vx| > 0$
4) $\forall i \in \mathbb{N}$, $uv^i wx^i y \in \mathcal{L}$

PROOF

BY $\mathcal{L}$ IS CONTEXT-FREE, $\exists\ \mathcal{G}$ CONTEXT-FREE GRAMMAR SUCH THAT $\mathcal{L} = \mathcal{L}(\mathcal{G})$

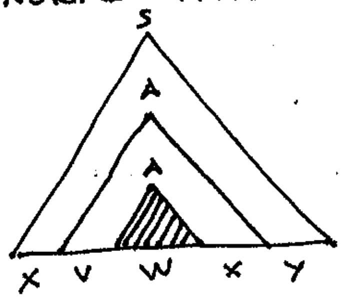
($\mathcal{G}$ IN CHOMSKY NORMAL FORM)

LET $\mathcal{G} = (V, T, S, P)$ AND LET $n = |V \setminus T|$

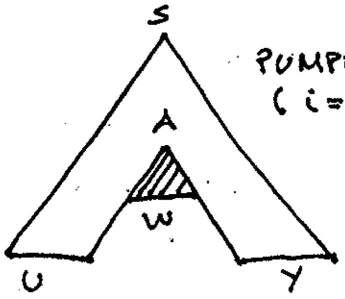
p IS THEN THE LENGTH OF THE LONGEST WORD BELONGING TO $\mathcal{L}(\mathcal{G})$ THAT CAN BE GENERATED BY DERIVATION TREES WHOSE DEPTH IS $\leq n$

SO, IF $|z| > p$, AT LEAST A NON-TERMINAL IS REPEATED TWICE IN THE PATH

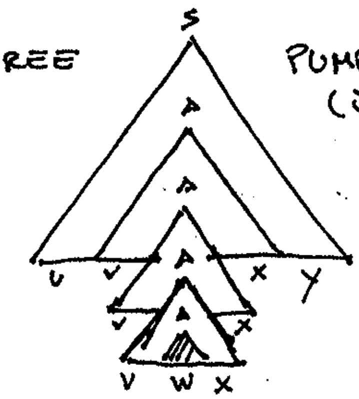
NORMAL TREE



PUMPED TREE
(i=0)



PUMPED TREE
(i>0)



PREDICTIVE TOP-DOWN PARSING

```
INPUT: $w\$$, TOP-DOWN PARSING TABLE FOR $G$
OUTPUT: A LEFTMOST DERIVATION OF $w$ IF $w \in \mathcal{L}(G)$, ERROR() OTHERWISE
INIT: $w\$$ IN THE BUFFER

REPEAT:
    $X := $ TOP()
    $y := $ ip
    IF $X$ IS A TERMINAL OR $\$$:
        IF $x = y$:
            POP($X$)
            MOVE ip FORWARD
        ELSE: ERROR()
    ELSE IF $T[X, y] = X \to Y_1 \ldots Y_K$:
        POP($X$)
        PUSH($Y_K \ldots Y_1$)
        OUTPUT "$X \to Y_1 \ldots Y_K$"
    ELSE: ERROR()
UNTIL: $X = \$$
```

FIRST (α)

```
BASE CASE  $\alpha = X$
IF $X$ IS A TERMINAL OR $\varepsilon$: $X$
IF $X$ IS A NON-TERMINAL:
    IF $X \to \varepsilon \in P$: ADD $\varepsilon$ TO FIRST($X$)
    IF $X \to Y_1 \ldots Y_K \in P$:
        FOR $j = 1 \ldots K$:
            IF $\varepsilon \in$ FIRST($Y_j$): ADD $\varepsilon$ TO FIRST($X$)
        IF $\varepsilon \in$ FIRST(($Y_1$) $\cup \ldots \cup$ ($Y_j$)) & $a \in$ FIRST($Y_{j+1}$):
            ADD $a$ TO FIRST($X$)

INDUCTIVE CASE  $\alpha = X_1 \ldots X_n$
FOR $i = 1 \ldots n$:
    IF $b \in$ FIRST($X_i$) & $\varepsilon \in$ FIRST($X_j$) FOR EACH $j < i$:
        ADD $b$ TO FIRST($\alpha$)
    IF $\varepsilon \in$ FIRST($X_j$) FOR EACH $\le j \le n$:
        ADD $\varepsilon$ TO FIRST($\alpha$)
```

FOLLOW (A)

A IS A NON-TERMINAL

```
1) ADD $ TO FOLLOW(A)
2) FOR EACH $B \to \alpha A \beta$ : ADD FIRST$(\beta) \setminus \{\varepsilon\}$ TO FOLLOW(A)
3) FOR EACH $B \to \alpha A$ : ADD FOLLOW (B) TO FOLLOW (A)
4) FOR EACH $B \to \alpha A \beta$ :
       IF $\varepsilon \in$ FIRST$(\beta)$ : ADD FOLLOW(B) TO FOLLOW (A)
```

CONSTRUCTION OF PREDICTIVE PARSING TABLE

INPUT: A GRAMMAR G

OUTPUT: PREDICTIVE PARSING TABLE FOR G

```
FOR EACH $A \to \alpha$ :
    FOR EACH $b \in$ FIRST$(\alpha)$ : $M[A,b] = A \to \alpha$
    IF $\varepsilon \in$ FIRST$(\alpha)$ :
        $\forall x \in$ FOLLOW (A) : $M[A,x] = A \to \alpha$
INSERT ERROR() IN EVERY EMPTY ENTRY
```

***

IF THERE ARE NO MULTIPLY DEFINED ENTRIES IN M THEN G IS SAID TO BE LL(1)

***

AMBIGUITY : THERE ARE 2 LEFTMOST (OR RIGHTMOST) DERIVATIONS THAT YELD THE SAME RESULT

LEFT-RECURSION : $A \xrightarrow{*} A\alpha$

LEFT-FACTORIZATION : $A \to \alpha\beta_1 \mid \alpha\beta_2$

ELIMINATION OF GENERAL LEFT-RECURSION

```
1. FIX AN ORDERING OF THE NON-TERMINALS OF $G$, SAY $A_1, \dots, A_n$
2. FOR $i = 1, \dots, n$ :
       FOR $j = 1, \dots, (i-1)$ :
           LET $A_j \to \delta_1 \mid \dots \mid \delta_k$ BE ALL THE PRODUCTIONS FOR $A_j$
           REPLACE $A_i \to A_j \gamma$ WITH $A_i \to \delta_1 \gamma \mid \dots \mid \delta_k \gamma$
       ELIMINATE IMMEDIATE LEFT-RECURSION FOR $A_i$
```

NOTE: THIS ALGORITHM DOESN'T WORK IF:

a) G HAS CYCLES ($A \xrightarrow{*} A$)

b) G HAS ε-PRODUCTIONS

LR PARSING

INPUT : STRING w , GRAMMAR G , PARSING TABLE M

OUTPUT : RIGHTMOST DERIVATION OF w IF $w \in \mathcal{L}(G)$
ERROR() OTHERWISE

INIT: STARTING SYMBOL S_0 ON THE STACK , w\$ IN THE INPUT BUFFER

ALGORITHM :

```
LET b BE THE FIRST SYMBOL OF w$
WHILE TRUE :
    LET s BE THE TOP OF THE STACK
    IF (M[s,b] = shift n) :
        PUSH (b) , PUSH(n)
        LET b BE THE NEXT INPUT SYMBOL
    ELIF (M[s,b] = reduce "$A \to \beta$") :
        POP 2*$|\beta|$ SYMBOLS FROM THE STACK
        LET m BE THE TOP OF THE STACK
        LET n BE SUCH THAT M[m,A] = goto n
        PUSH A , PUSH n
        OUTPUT "$A \to \beta$"
    ELIF (M[s,b] = accept) :
        ~~break~~
        BREAK
    ELSE : ERROR()
```

ELIMINATION OF LEFT-FACTORIZATION

LET $A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \cdots \mid \alpha\beta_n \mid \gamma_1 \mid \cdots \mid \gamma_K$ BE ALL THE PRODUCTION FOR A (WHERE α IS THE LONGEST COMMON PREFIX)

REPLACE THEM WITH

$A \to \alpha A' \mid \gamma_1 \mid \cdots \mid \gamma_K$

$A' \to \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n$

PROPERTIES OF LL(1) GRAMMARS

1. NO AMBIGUOUS GRAMMAR IS LL(1)
2. NO LEFT-RECURSIVE GRAMMAR IS LL(1)
3. NO ~~GRAMMA~~ LEFT-FACTORIZABLE GRAMMAR IS LL(1)
4. G IS LL(1) IF AND ONLY IF:

   $\forall\ A \to \alpha \mid \beta \in P$

   a) $\text{FIRST}(\alpha) \cap \text{FIRST}(\beta) = \emptyset$

   b) $\varepsilon \in \text{FIRST}(\alpha) \Rightarrow \text{FIRST}(\beta) \cap \text{FOLLOW}(A) = \emptyset$

   $\varepsilon \in \text{FIRST}(\beta) \Rightarrow \text{FIRST}(\alpha) \cap \text{FOLLOW}(A) = \emptyset$

CLOSURE (I)

```
REPEAT: FOR EACH $A \to \alpha.B\beta$ IN I & FOR EACH $B \to \gamma$ IN G:
            IF $B \to .\gamma \notin I$ : ADD IT TO I
UNTIL: SATURATION
RETURN: I
```

GOTO (I, X) = CLOSURE (K)

K = SET OF ITEMS OF THE SHAPE $A \to \alpha X.\beta$ SUCH THAT $A \to \alpha.B\beta \in I$

GENERATION OF THE COLLECTION OF ITEMS

```
C := CLOSURE ({$S' \to .S$})
REPEAT: FOR EACH SET $I \in C$ &
            FOR EACH SYMBOL $X \in G$ SUCH THAT GOTO $(I,X) \neq \emptyset$ & GOTO $(I,X) \notin C$:
                ADD GOTO(I, X) TO C
UNTIL: SATURATION
```

CONSTRUCTION OF THE SLR PARSING TABLE

INPUT: ENRICHED GRAMMAR $\mathcal{G}$ ($\mathcal{G} \cup \{S' \to S\}$)

OUTPUT: SLR PARSING TABLE FOR $\mathcal{G}$

```
BUILD THE COLLECTION OF SETS OF ITEMS FOR $\mathcal{G}$, SAY $I_0, \dots, I_n$
FOR EACH $I_j$ IN $I_0 \dots I_n$:
    IF $A \to \alpha . B\beta \in I_j$ AND GOTO $(I_j, a) = I_K$:
        $M[j,a] :=$ shift K
    IF $A \to \alpha . \in I_j$:
        $M[j,x] :=$ reduce "$A \to \alpha.$" FOR EACH $x \in$ FOLLOW$(A)$
    IF $S' \to S. \in I_j$:
        $M[j,\$] :=$ accept
```

(IF THERE ARE MULTIPLY DEFINED ENTRIES THE GRAMMAR CANNOT BE PARSED USING SLR (AND IS NOT LALR))

```
    IF GOTO$(I_j, A) = I_K$:
        $M[j,A] :=$ goto K
ERROR() IN ALL EMPTY ENTRIES
```

STATE S_0 FOR THE PARSER IS THE SET OF ITEMS CONTAINING $S' \to . S$

CLOSURE (I) FOR LALR

```
FOR EACH $[A \to \alpha . B\beta, x] \in I$:
    FOR EACH $B \to \gamma \in \mathcal{G}$:
        FOR EACH $b \in$ FIRST$(\beta x)$:
            ADD $[B \to . \gamma, b]$ to $I$
```

GOTO (I, X) FOR LALR

```
$J := \emptyset$
FOR EACH $[A \to \alpha . X\beta, x] \in I$:
    ADD $[A \to \alpha X . \beta, x]$ to $J$
RETURN $J$
```

LOOKAHEAD PROPAGATION GRAPH CONSTRUCTION

NODES PAIRS $(I_j$, KERNEL LR(0) ITEM IN $I_j)$ FOR EVERY I_j AND FOR EVERY KERNEL ITEM IN I_j

LABELS FOR EACH NODE V THERE IS FW(V)

INIT
$$\begin{cases} FW((I_0, S' \to .S)) = \{\$\} \\ FW(v) = \emptyset \text{ FOR EVERY OTHER NODE} \end{cases}$$

EDGES

```
FOR EACH $v = (I_K, A \to \alpha.\beta)$:
    $J$ = CLOSURE ( {[ $A \to \alpha.\beta$, 2]})        # 8 IS A NEW SYMBOL
    FOR EACH [$B \to .$, x] ∈ J:                        # "DONE" PRODUCTIONS
        LET $v' = (I_K, B \to .)$
        IF x = 2 : ADD EDGE (v, v')                     # PROPAGATION IF SAME SYMBOL
        ELSE: ADD x TO FW(v')                           # GENERATION IF DIFFERENT
    FOR EACH [$B \to \gamma.Y\delta$, x]:               # CONSIDER [illegible]
        LET $v'$ = (GOTO $(I_K, Y)$, $B \to \gamma Y.\delta$)
        IF x = 2 : ADD EDGE (v, v')                     # PROPAGATION
        ELSE: ADD x TO FW(v')                           # GENERATION
```

OBSERVATIONS

① LET $v = (S, B \to \gamma.a\delta)$

IF CLOSURE $(\{[B \to \gamma.a\delta, 2]\}) = \{[B \to \gamma.a\delta, 2]\}$:

V ALWAYS PROPAGATES (AND <u>ONLY</u> PROPAGATES) TO (GOTO(S,a), $B \to \gamma a.\delta$)

② LET $v = (S, i)$ WITH $i = B \to \alpha.$

IF CLOSURE $(\{[i, 8]\}) = \{[i, 2]\}$ AND GOTO IS DEFINED FROM i :

<NOTHING TO DO>

LALR PARSING

- COLLECTION OF SETS OF LR(0) ITEMS (AS IN SLR)
- KERNELS OF LR(0) ITEMS
- LOOKAHEAD PROPAGATION GRAPH
- CHECK FW(V) FOR NODES V SUCH THAT $V = (i, B \to \alpha.)$

PARSING TABLE

SAME AS SLR EXCEPT :

```
IF $B \to \alpha. \in I_j$ AND $B \neq S'$ AND $x \in FW((j, B \to \alpha.))$ :
    $T[j,x]$ = reduce "$B \to \alpha.$"
```

SYNTAX-DIRECTED DEFINITIONS

ATTRIBUTES AND RULES FOR THE SYMBOLS OF CONTEXT-FREE GRAMMARS

SYNTHESIZED ATTRIBUTE

$A \to X_1 \dots X_j \, Y \, X_{j+1} \dots X_K$

AN ATTRIBUTE $A.a$ DEFINED AS A FUNCTION OF ATTRIBUTES OF $A, X_1, \dots X_K, Y$

INHERITED ATTRIBUTE

$B \to X_1 \dots X_j \, A \, X_{j+1} \dots X_n$

AN ATTRIBUTE $A.a$ DEFINED AS A FUNCTION OF $B, X_1, \dots, X_n$

DEPENDENCY GRAPH

```
FOR NODE IN THE PARSE TREE :
    FOR SYMBOL ASSOCIATED WITH NODE :
        SET A NODE OF THE DEPENDENCY GRAPH
FOR EACH ATTRIBUTE $A.a$ :
    FOR EACH ATTRIBUTE $X.x$ USED TO DEFINE $A.a$ :
        SET EDGE FROM $X.x$ TO $A.a$        # x IS NEEDED TO COMPUTE a
```

L-ATTRIBUTED DEFINITIONS

DEFINITIONS WHERE ATTRIBUTES ARE EITHER SYNTHESIZED OR INHERITED AND

```
FOR EACH $A \to X_1 \dots X_K$ :
    FOR EACH $X_j.i$ (ATTRIBUTE) :
        $X_j.i$ ONLY USES INHERITED ATTRIBUTES OF $A$ OR
                          INHERITED OR SINTHESIZED ATTRIBUTES OF $X_1, \dots, X_{j-1}$
```

SYNTAX-DRIVEN TRANSLATIONS

SEMANTIC ACTIONS IN THE BODY OF THE PRODUCTIONS