Automata - CFG & PDA Cheat Sheet

by Vipera via cheatography.com/128346/cs/25668/

CFG Definition

Context-Free Grammar:	**G = (V,T,S,P)**
V: Set of variables	{S}
T: Set of terminal symbols	{a,b}
S: Start variable	S
P: Set of productions	{S→aSb, S→ε}

ONLY ONE variable → String of variables **and** terminals

Union of Two Languages

Example:	$L=\{0^n1^n \mid n\geq 0\} \cup \{1^n0^n \mid n\geq 0\}$
Break problem in two	$\mathbf{S^1}\to 0S^11\mid\varepsilon$ $\mathbf{S^2}\to 1S^20\mid\varepsilon$
Merge	$\mathbf{S\to S^1\mid S^2}$ $S^1\to 0S^11\mid\varepsilon$ $S^2\to 1S^20\mid\varepsilon$

Simplifications of CFG

Substitution *($B\to y^1$)*	A→xBz $B\to y^1$	$A\to xBz\mid xy^1z$
	A→xBBz $B\to y^1$	$A\to xBBz\mid xBy^1z \mid xy^1Bz\mid xy^1y^1z$
Removing ε *(B →ε)*	A→xBz B→ε	A→xBz|xz
Unit Production *(A →B)*	A→B B→bb	A→bb B→bb
Useless Productions	A→aA (infinite)	∴ remove
	Unreachable from S	∴ remove

Step 1: Remove Nullable Variables
Step 2: Remove Unit-Production
Step 3: Remove Useless Variables

DFA to CFG

1. Create variable $\mathbf{R^i}$ for every state $\mathbf{q^i}$
2. Create rule $\mathbf{R^i \to aR^j}$ for every transition $\mathbf{\delta(q^i,a) \to q^j}$
3. For accept states $\mathbf{q^i}$ create rule $\mathbf{R^i \to \varepsilon}$
4. For initial state $\mathbf{q^0}$ make $\mathbf{R^0}$ the start variable

Conversion to Chomsky Normal Form

Step 0:	If start symbol (S) is on the right hand side, change start symbol $S^0\to S$
Step 1:	Remove Nullable variables $(A\to\varepsilon)$ and Unit productions $(A\to B)$
Step 2:	For every symbol *a* add $T^a\to a$
Step 3:	Replace $A\to C^1C^2...C^n$ with $A\to C^1V^1$ $V^1\to C^2V^2$ ... $V^{n-2}\to C^{n-1}C^n$

Chomsky form only has productions in forms
A→BC
A→a

Greibach Normal Form

All Productions have form:	$A\to aV^1V^2..V^k$: $k\geq 0$
Example	
S→abSb S→aa	$S\to aT^bST^b$ $S\to aT^a$ $T^a\to a$ $T^b\to b$

PDA

Transitions: **a, b → c** means when input is a, remove *b* from stack and add *c*

If the automaton attempts to pop from empty stack then it halts and rejects input.

PDA (cont)

A string is accepted if there is a computation such that:
All the input is consumed
The last state is an accepting state

PDA Formalities

PDA Representation	**M=(Q,Σ,Γ,δ,q0,z,F)**
Q: States	{q0,q1,q2}
Σ: Input Alphabet	{a,b}
Γ: Stack Alphabet	{a,b,$}
δ: Transition Functions	δ(q,a,w1)={(q2,w2)}
q0: Initial State	q0
z: Stack Start Symbol	$
F: Accept States	{q2}

CFG to PDA

Start with PDA of $q0 \to^{\varepsilon,\varepsilon\to S}\to q1 \to^{\varepsilon,\$\to\$}\to q2$

For each CFG production $A\to w$ add **ε,A→w**

For each CFG terminal *a* add **a,a→ε**

"Easy" PDA to CFG

For the pair of transitions:
$\to^{a,\varepsilon\to t}\to \quad \to^{b,t\to\varepsilon}\to$

Add the production: $A^{pq}\to aA^{rs}b$
For each state **p** add: $A^{pp}\to\varepsilon$
For each state-triple (p,q,r) add: $A^{pr}\to A^{pq}A^{qr}$

For initial state and accept state:
→ &

Add the production: $S\to A^{0a}$

Easy PDAs:
• Have only 1 accept state
• When accepting a string, the stack is empty (only inital symbol)
• Each transition pushes **or** pops

PDA to "Easy" PDA

1. The PDA has a single accept state	Create new accept state and make ε,ε→ε transitions from old accept states to the new
2. Use new initial stack symbol #	New initial state, that transitions to a new state with ε,ε→@ (auxiliary symbol) that transitions to the old initial state with ε,ε→$
3. On acceptance the stack contains only stack symbol #	Old accept state transitions to new to new accept state with ε,@→ε, ανδ self loops with ε,x→ε where ∀x∈Γ-{@,#}
4. Transitions can't push **and** pop	Replace any →$^{σ,a→b}$→ with →$^{σ,a→ε}$→ →$^{ε,ε→b}$→
5. 4. Transitions can't neither push nor pop	Replace any →$^{σ,ε→ε}$→ with →$^{σ,ε→∂}$→ →$^{ε,∂→ε}$→