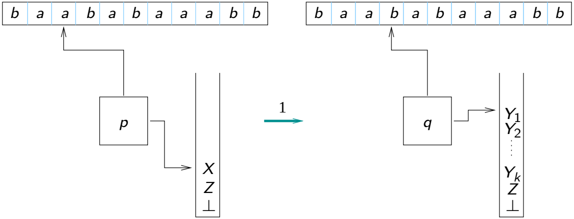
# Outline

# Pushdown Automata + CFG: history

- CFG's were introduced by Noam Chomsky in 1956.
- Oettinger introduced PDA's for parsing applications in 1961.
- Chomsky, Schutzenberger, and Evey showed equivalence of CFG's and PDA's in 1962.

# How a PDA works



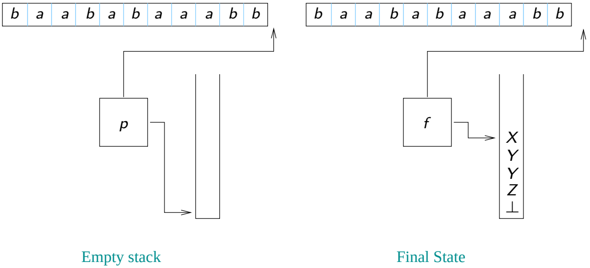

Each step of the PDA looks like:

- Read current symbol and advance head;
- Read and pop top-of-stack symbol
- Push in a string of symbols on the stack.
- Change state.

Each transition Looks like

$$(p, a, X) \rightarrow (q, Y_1 Y_2 \cdots Y_k).$$

# Acceptance




Accept input if

- Input is consumed and stack is empty (Acceptance by "Empty Stack")
- Or, input is consumed and PDA is in a final state (Acceptance by "Final State").

# Example PDA

## Example PDA for $\{a^n b^n \mid n \geq 0\}$

$$
\begin{array}{lcl}
(s, \epsilon, \perp) & \rightarrow & (s, \epsilon) \\
(s, a, \perp) & \rightarrow & (p, A) \\
(p, a, A) & \rightarrow & (p, AA) \\
(p, b, A) & \rightarrow & (q, \epsilon). \\
(q, b, A) & \rightarrow & (q, \epsilon).
\end{array}
$$

# Example PDA

## Example PDA for $\{a^n b^n \mid n \geq 0\}$

$$
\begin{array}{rcl}
(s, \epsilon, \perp) & \rightarrow & (s, \epsilon) \\
(s, a, \perp) & \rightarrow & (p, A) \\
(p, a, A) & \rightarrow & (p, AA) \\
(p, b, A) & \rightarrow & (q, \epsilon). \\
(q, b, A) & \rightarrow & (q, \epsilon).
\end{array}
$$

Illustrate run on input "*aaabbb*".

# Example PDA

## Example PDA for $\{a^n b^n \mid n \geq 0\}$

$$
\begin{array}{lcl}
(s, \epsilon, \perp) & \rightarrow & (s, \epsilon) \\
(s, a, \perp) & \rightarrow & (p, A) \\
(p, a, A) & \rightarrow & (p, AA) \\
(p, b, A) & \rightarrow & (q, \epsilon). \\
(q, b, A) & \rightarrow & (q, \epsilon).
\end{array}
$$

Illustrate run on input "*aaabbb*".

What happens on input "*aaabbbb*"?

# PDA's more formally

A Pushdown Automaton is a structure of the form

$$\mathcal{M} = (Q, A, \Gamma, s, \delta, \perp, F)$$

where

- $Q$ is a finite set of states,
- $A$ is the input alphabet,
- $\Gamma$ is the stack alphabet,
- $s \in Q$ is the start state,
- $\delta \subseteq_{fin} Q \times (A \cup \{\epsilon\}) \times \Gamma \times Q \times \Gamma^*$ is the (non-deterministic) transition relation,
- $\perp \in \Gamma$ is the bottom-of-stack symbol,
- $F \subseteq Q$ is the set of final states.

# Configurations, runs, etc. of a PDA

- A configuration of $\mathcal{M}$ is of the form $(p, u, \gamma) \in Q \times A^* \times \Gamma^*$, which says "$\mathcal{A}$ is in state $p$, with unread input $u$, and stack contents $\gamma$".
- Initial configuration of $\mathcal{M}$ on input $w$ is $(s, w, \perp)$.
- 1-step transition of $\mathcal{M}$: If $(p, a, X) \to (q, \alpha)$ is a transition in $\delta$, then

$$(p, au, X\beta) \overset{1}{\Rightarrow} (q, u, \alpha\beta).$$

- Similarly, if $(p, \epsilon, X) \to (q, \alpha)$ is a transition in $\delta$, then

$$(p, u, X\beta) \overset{1}{\Rightarrow} (q, u, \alpha\beta).$$

- $\mathcal{M}$ accepts $w$ by empty stack if $(s, w, \perp) \overset{*}{\Rightarrow} (q, \epsilon, \epsilon)$.
- $\mathcal{M}$ accepts $w$ by final state if $(s, w, \perp) \overset{*}{\Rightarrow} (f, \epsilon, \gamma)$ for some $f \in F$.
- Language accepted by $\mathcal{M}$ is denoted $L(\mathcal{M})$.

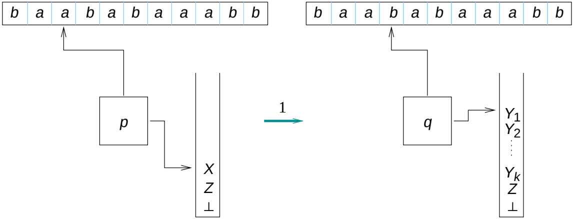
# Exercise

Design PDA's for the following languages:

- Balanced Parenthesis
- $\{a, b\}^* - \{ww \mid w \in \{a, b\}^*\}$.

# Outline

# How a PDA works



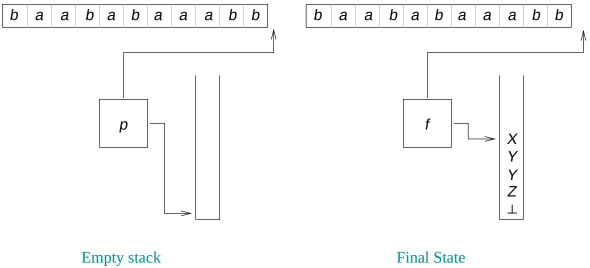

Each step of the PDA looks like:

- Read current symbol and advance head;
- Read and pop top-of-stack symbol
- Push in a string of symbols on the stack.
- Change state.

Each transition Looks like

$$(p, a, X) \rightarrow (q, Y_1 Y_2 \cdots Y_k).$$

# Acceptance




Accept input if

- Input is consumed and stack is empty (Acceptance by “Empty Stack”)
- Or, input is consumed and PDA is in a final state (Acceptance by “Final State”).

# Equivalence of acceptance criteria

## Claim

- Given a PDA $M$ that accepts by Final State we can give a PDA $M'$ that accepts by Empty Stack such that $L(M') = L(M)$.
- Conversely, given a PDA $M$ that accepts by Empty Stack we can give a PDA $M'$ that accepts by Final State such that $L(M') = L(M)$.

In fact given a PDA $M$ we can construct a PDA $M'$ that accepts the same language as $M$, by both acceptance criteria.

# From Final State to ES/FS

What is the problem in doing this?

# From Final State to ES/FS

What is the problem in doing this?

- $M$ may reject an input by emptying its stack.

# From Final State to ES/FS

What is the problem in doing this?

- $M$ may reject an input by emptying its stack.

- Let $M = (Q, A, \Gamma, s, \delta, \perp, F)$.
- Define $M' = (Q \cup \{s', t\}, A, \Gamma \cup \{\perp\!\!\!\perp\}, s', \delta', \perp\!\!\!\perp, \{t\})$, where $\delta'$ is $\delta$ plus the transitions:

$$
\begin{array}{lcll}
(s', \epsilon, \perp\!\!\!\perp) & \to & (s, \perp\perp\!\!\!\perp) & \\
(s, a, \perp) & \to & (p, A) & \\
(f, \epsilon, X) & \to & (t, X) & \text{for } X \in \Gamma \cup \{\perp\!\!\!\perp\} \\
(t, \epsilon, X) & \to & (t, \epsilon) & \text{for } X \in \Gamma \cup \{\perp\!\!\!\perp\}.
\end{array}
$$

- Argue that if $w \in L(M)$ then $w \in L(M')$.
- Argue that if $w \in L(M')$ then $w \in L(M)$.

# From Empty Stack to ES/FS

- Let $M = (Q, A, \Gamma, s, \delta, \perp)$.
- Define $M' = (Q \cup \{s', t\}, A, \Gamma \cup \{⫫\}, s', \delta', ⫫, \{t\})$, where $\delta'$ is $\delta$ plus the transitions:

$$\begin{array}{lcl} (s', \epsilon, ⫫) & \rightarrow & (s, \perp ⫫) \\ (q, \epsilon, ⫫) & \rightarrow & (t, ⫫) \\ (t, \epsilon, ⫫) & \rightarrow & (t, \epsilon). \end{array}$$

- Argue that if $w \in L(M)$ then $w \in L(M')$.
- Argue that if $w \in L(M')$ then $w \in L(M)$.

# Outline

# CFG = PDA

**Theorem (Chomsky-Evey-Schutzenberger)**

*The class of languages definable by Context-Free Grammars and Pushdown Automata coincide.*

# From CFG to PDA

Leftmost derivation: A derivation in which at each step the left-most non-terminal is rewritten.

**CFG $G_4$**

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Leftmost derivation in $G_4$:

$$\underline{S}$$

# From CFG to PDA

Leftmost derivation: A derivation in which at each step the left-most non-terminal is rewritten.

**CFG $G_4$**

$$S \quad \rightarrow \quad (S) \mid SS \mid \epsilon.$$

Leftmost derivation in $G_4$:

$$\underline{S} \quad \Rightarrow \quad (\underline{S})$$

# From CFG to PDA

Leftmost derivation: A derivation in which at each step the left-most non-terminal is rewritten.

**CFG $G_4$**

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Leftmost derivation in $G_4$:

$$\begin{aligned} \underline{S} &\Rightarrow (\underline{S}) \\ &\Rightarrow (\underline{S}S) \end{aligned}$$

# From CFG to PDA

Leftmost derivation: A derivation in which at each step the left-most non-terminal is rewritten.

**CFG $G_4$**

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Leftmost derivation in $G_4$:

$$\begin{aligned} \underline{S} &\Rightarrow (\underline{S}) \\ &\Rightarrow (\underline{S}S) \\ &\Rightarrow (\underline{S}SS) \end{aligned}$$

# From CFG to PDA

Leftmost derivation: A derivation in which at each step the left-most non-terminal is rewritten.

**CFG $G_4$**

$$S \quad \rightarrow \quad (S) \mid SS \mid \epsilon.$$

Leftmost derivation in $G_4$:

$$\begin{aligned} \underline{S} \quad & \Rightarrow \quad (\underline{S}) \\ & \Rightarrow \quad (\underline{S}S) \\ & \Rightarrow \quad (\underline{S}SS) \\ & \Rightarrow \quad ((\underline{S})SS) \end{aligned}$$

# From CFG to PDA

Leftmost derivation: A derivation in which at each step the left-most non-terminal is rewritten.

**CFG $G_4$**

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Leftmost derivation in $G_4$:

$$\begin{aligned}
\underline{S} &\Rightarrow (\underline{S}) \\
&\Rightarrow (\underline{S}S) \\
&\Rightarrow (\underline{S}SS) \\
&\Rightarrow ((\underline{S})SS) \\
&\Rightarrow ((\underline{S}S)SS)
\end{aligned}$$

# From CFG to PDA

Leftmost derivation: A derivation in which at each step the left-most non-terminal is rewritten.

**CFG $G_4$**

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Leftmost derivation in $G_4$:

$$\begin{aligned}\underline{S} &\Rightarrow (\underline{S})\\ &\Rightarrow (\underline{S}S)\\ &\Rightarrow (\underline{S}SS)\\ &\Rightarrow ((\underline{S})SS)\\ &\Rightarrow ((\underline{S}S)SS)\\ &\Rightarrow (((\underline{S})S)SS)\end{aligned}$$

# From CFG to PDA

Leftmost derivation: A derivation in which at each step the left-most non-terminal is rewritten.

**CFG $G_4$**

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Leftmost derivation in $G_4$:

$$\begin{aligned} \underline{S} &\Rightarrow (\underline{S}) \\ &\Rightarrow (\underline{S}S) \\ &\Rightarrow (\underline{S}SS) \\ &\Rightarrow ((\underline{S})SS) \\ &\Rightarrow ((\underline{S}S)SS) \\ &\Rightarrow (((\underline{S})S)SS) \\ &\Rightarrow ((()\underline{S})SS) \end{aligned}$$

# From CFG to PDA

Leftmost derivation: A derivation in which at each step the left-most non-terminal is rewritten.

**CFG $G_4$**

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Leftmost derivation in $G_4$:

$$\begin{aligned}
\underline{S} &\Rightarrow (\underline{S}) \\
&\Rightarrow (\underline{S}S) \\
&\Rightarrow (\underline{S}SS) \\
&\Rightarrow ((\underline{S})SS) \\
&\Rightarrow ((\underline{S}S)SS) \\
&\Rightarrow (((\underline{S})S)SS) \\
&\Rightarrow ((()\underline{S})SS) \\
&\Rightarrow ((())\underline{S}S)
\end{aligned}$$

# From CFG to PDA

Leftmost derivation: A derivation in which at each step the left-most non-terminal is rewritten.

**CFG $G_4$**

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Leftmost derivation in $G_4$:

$$\begin{aligned}
\underline{S} &\Rightarrow (\underline{S}) \\
&\Rightarrow (\underline{S}S) \\
&\Rightarrow (\underline{S}SS) \\
&\Rightarrow ((\underline{S})SS) \\
&\Rightarrow ((\underline{S}S)SS) \\
&\Rightarrow (((\underline{S})S)SS) \\
&\Rightarrow ((()\underline{S})SS) \\
&\Rightarrow ((())\underline{S}S) \\
&\Rightarrow ((())(\underline{S})S) \\
&\Rightarrow ((())()\underline{S}) \quad \Rightarrow ((())()).
\end{aligned}$$

# From CFG to PDA

Let $G = (N, A, S, P)$ be a CFG. Assume WLOG that all rules of $G$ are of the form

$$X \to cB_1 B_2 \cdots B_k$$

where $c \in A \cup \{\epsilon\}$ and $k \geq 0$.

- Idea: Define a PDA $M$ that simulates a leftmost derivation of $G$.
- Define $M = (\{s\}, A, N, s, \delta, \perp)$ where $\delta$ is given by:

$$(s, c, X) \quad \to \quad (s, B_1 B_2 \cdots B_k),$$

  whenever $X \to cB_1 B_2 \cdots B_k$ is a production in $G$.

## From PDA to CFG

- First show that we can go over to a PDA $M'$ with single state.
- Then simulate $M'$ by a CFG.

# From PDA to single-state PDA

- Let $M = (Q, A, \Gamma, s, \delta, \perp, \{t\})$ be the given PDA.
- Define $M' = (\{u\}, A, Q \times \Gamma \times Q, u, \delta', (s, \perp, t), \{u\})$, where $\delta'$ is given by

$$(u, c, (p, A, q_k)) \rightarrow (u, (q_0 B_1 q_1)(q_1 B_2 q_2) \cdots (q_{k-1} B_k q_k))$$

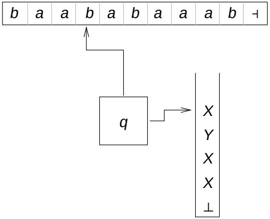
whenever $(p, c, A) \rightarrow (q, (B_1 B_2 \cdots B_k))$ is a transition of $M$. In particular:

$$(u, c, (p, A, q)) \rightarrow (u, \epsilon)$$

if $(p, c, A) \rightarrow (q, \epsilon)$ is a transition of $M$.

# Outline

# Deterministic PDA's




A PDA with restrictions that:

- **At most** one move possible in any configuration.
  - For any state $p$, $a \in A$, and $X \in \Gamma$: at most one move of the form $(p, a, X) \to (q, \gamma)$ or $(p, \epsilon, X) \to (q, \gamma)$.
  - Effectively, a DPDA must see the current state, and top of stack, and decide whether to make $\epsilon$-move or read input and move.
- Accepts by final state.
- We need right-end marker "⊣" for the input.

# Example DPDA

## Example DPDA for $\{a^n b^n \mid n \geq 0\}$

$$\begin{array}{lcl}
(s, a, \perp) & \rightarrow & (p, A\perp) \\
(p, a, A) & \rightarrow & (p, AA) \\
(p, b, A) & \rightarrow & (q, \epsilon) \\
(q, b, A) & \rightarrow & (q, \epsilon) \\
(q, \dashv, \perp) & \rightarrow & (t, \perp) \\
(s, \dashv, \perp) & \rightarrow & (t, \perp).
\end{array}$$

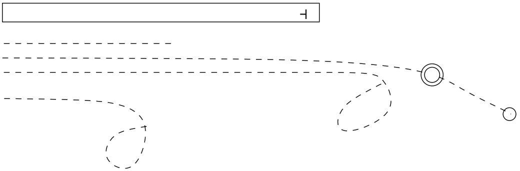
# DCFL's are closed under complementation

**Theorem (Closure under complementation)**

*The class of languages definable by Deterministic Pushdown Automata (i.e. DCFL's) is closed under complementation.*

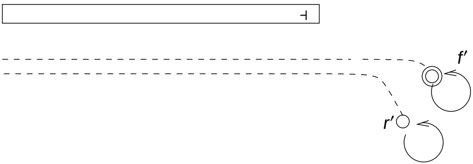
# Problem with complementing a DPDA

Try flipping final and non-final states: Problems?




Loops denote an infinite sequence of $\epsilon$-moves.

# Desirable form of DPDA




Now we can make $r'$ unique accepting state, to accept complement of $M$.

# Construction - Step 1

Let $M = (Q, A, \Gamma, s, \delta, \perp, F)$ be given DPDA. First construct DPDA $M'$ which

- Does not get stuck due to no transition or stack empty.
- Has only "sink" final states.

# Construction - Step 1

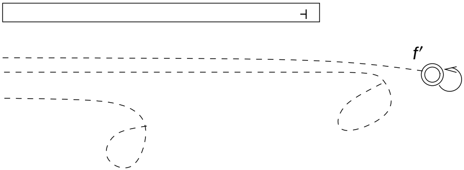
Define $M' = (Q \cup Q' \cup \{s_1, r, r'\}, A, \Gamma \cup \{⫫\}, s_1, \delta', ⫫, F')$ where

- $Q' = \{q' \mid q \in Q\}$ and $F' = \{f' \mid f \in F\}$.
- $\delta'$ is obtained from $\delta$ as follows:
  - Assume $M$ is "complete" (does not get stuck due to no transition). (If not, add a dead state and add transitions to it.)
  - Make sure $M'$ never empties its stack, keep track of whether we have seen end of input (primed states) or not (unprimed states):

$$
\begin{array}{lcll}
(s_1, \epsilon, ⫫) & \rightarrow & (s, \perp ⫫) & \\
(p, \epsilon, ⫫) & \rightarrow & (r, ⫫) & (p \in Q) \\
(p', \epsilon, ⫫) & \rightarrow & (r', ⫫) & (p' \notin F') \\
(p, \dashv, X) & \rightarrow & (q', \gamma) & \text{if } (p, \dashv, X) \rightarrow (q, \gamma) \in \delta. \\
(p', \epsilon, X) & \rightarrow & (q', \gamma) & \text{if } (p, \epsilon, X) \rightarrow (q, \gamma) \in \delta. \\
(r, a, X) & \rightarrow & (r, X) & \\
(r, \dashv, X) & \rightarrow & (r', X) & \\
(r', \epsilon, X) & \rightarrow & (r', X) & \\
(f', \epsilon, X) & \rightarrow & (f', X) & (f \in F) \text{ Also drop trans. going from } f'.
\end{array}
$$

# After Step 1

DPDA $M'$ only has the following kinds of behaviours now:



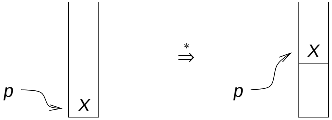

Loops denote an infinite sequence of $\epsilon$-moves.

# Construction - Step 2

A spurious transition in $M'$ is a transition of the form $(p, \epsilon, X) \to (q, \gamma)$ such that

$$(p, \epsilon, X) \overset{*}{\Rightarrow} (p, \epsilon, X\alpha)$$

for some stack contents $\alpha$.



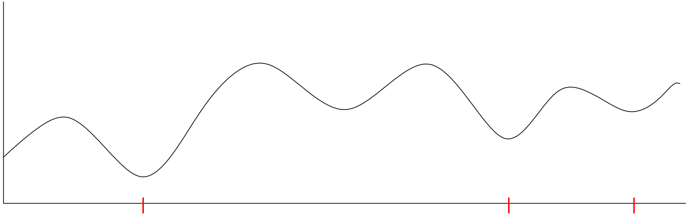

Identify spurious transitions in $M'$ and remove them:
If $(p, \epsilon, X) \to (q, \gamma)$ is a spurious transition, replace it with

$$\begin{array}{lcll} (p, \epsilon, X) & \to & (r, X) & \text{If } p \in Q \\ (p, \epsilon, X) & \to & (r', X) & \text{If } p \in Q' - F'. \end{array}$$

# Correctness

Argue that:

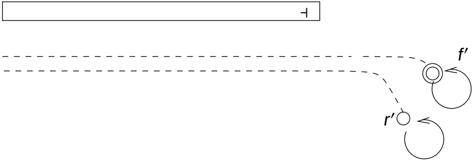
- Deleting a spurious transition (starting from a non-$F'$-final state) does not change the language of $M'$.
- All infinite loops use a spurious transition.
  - Look at graph of stack height along infinite loop, and argue that there are infinitely many future minimas.



  - Further look at transitions applied at these points and observe that one must repeat.
- Thus replacing spurious transitions as described earlier will remove the remaining undesirable loops from $M'$'s behaviours.

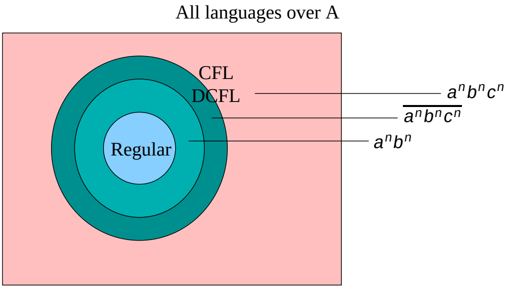
# Complementing

- Resulting $M''$ has the desired behaviour (every run either reaches a final sink state or the reject sink state $r'$.).



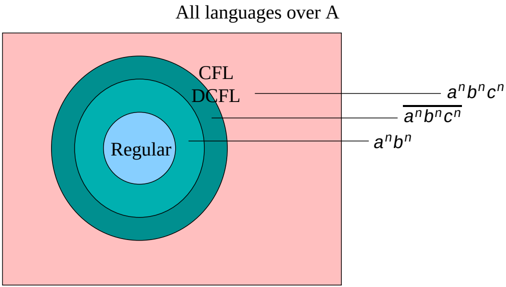

- Now make $r'$ unique final state to complement the language of $M$.

# Closure Properties of DCFL's



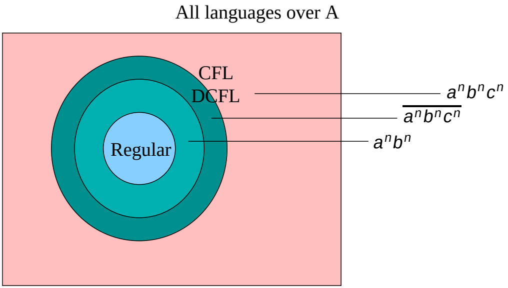

| | Closed? |
|---|---|
| Complementation | |

# Closure Properties of DCFL's



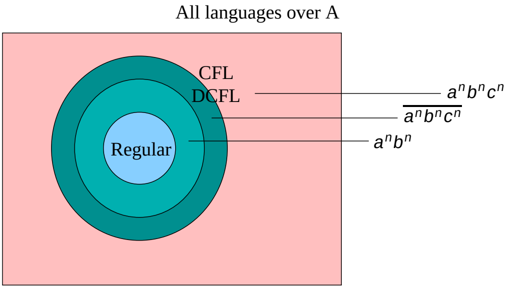

| | Closed? |
|---|---|
| Complementation | √ |
| Union | |

# Closure Properties of DCFL's




| | Closed? |
|---|---|
| Complementation | $\surd$ |
| Union | X |
| Intersection | |

# Closure Properties of DCFL's




| | Closed? |
|---|---|
| Complementation | √ |
| Union | X |
| Intersection | X |