II.6.1. Push Down Automata

II.6.2. Equivalence of Final-State and Empty-Stack-PDAs

II.6.3. Equivalence of CFG and PDA
II.6.1. Push Down Automata

II.6.2. Equivalence of Final-State and Empty-Stack-PDAs

II.6.3. Equivalence of CFG and PDA
For this subsection no additional material has been added yet.
II.6.1. Push Down Automata

II.6.2. Equivalence of Final-State and Empty-Stack-PDAs

II.6.3. Equivalence of CFG and PDA
Theorem (Repeated from Main Material)

Let $L$ be a language. The following are equivalent:

1. $L = L(P)$ for a final state PDA $P$.
2. $L = L(P)$ for an empty stack PDA $P$. 
Proof of \((1) \Rightarrow (2)\)

Let \(P = (T, Q, \Gamma, q_0, Z_0, F, \delta)\).

The idea for constructing an empty stack PDA \(P'\) is as follows:

- \(P'\) operates essentially as \(P\).
- If \(P\) reaches an accepting state, then \(P'\) can switch to a special state. In that state it empties using \(\epsilon\)-transition the stack and therefore accepts the string.
II.6.2. Equivalence of Final-State and Empty-Stack-PDAs

Proof of \( (1) \Rightarrow (2) \)

- However it might be that \( P \) and therefore as well \( P' \) empties the stack without having reached an accepting state.
- In order to prevent that we make sure that there is a special symbol at the bottom of the stack, which can only be popped when we are in \( q_{final} \).
- So we start in a new initial state \( q_0' \) with a new initial stack symbol \( Z_0' \).
- From that initial configuration \( P' \) moves to state \( q_0 \) and pushes \( Z_0 \) on the stack.
- We denote the transition relation for \( P' \) by \( a \rightarrow_0 \), similarly for the derived relation \( w \rightarrow_0^* \).
### Resulting PDA

**PDA** \( P' \)  
**terminals** \( T \)  
**states** \( Q' := Q \cup \{ q_0', q_{\text{final}} \} \)  
**stack alphab.** \( \Gamma' := \{ Z_0' \} \cup \Gamma \)  
**start state** \( q_0' \)  
**start stack** \( Z_0' \)  
**final** \( \emptyset \)  
**transitions**  
\[ (q_0', Z_0') \xrightarrow{\epsilon} (q_0, Z_0 Z_0') \]  
\[ (q, Z) \xrightarrow{a} (q', z') \text{ if } (q, Z) \xrightarrow{a} (q', z') \]  
\[ (q, Z) \xrightarrow{\epsilon} (q_{\text{final}}, Z) \text{ for } q \in F, Z \in \Gamma' \]  
\[ (q_{\text{final}}, Z) \xrightarrow{\epsilon} (q_{\text{final}}, \epsilon) \text{ for } Z \in \Gamma' \]
Proof of $L_{\text{final}}(P) = L_{\text{empty}}(P')$

We show that $L_{\text{final}}(P) = L_{\text{empty}}(P')$ by showing

1. $L_{\text{final}}(P) \subseteq L_{\text{empty}}(P')$, 
2. $L_{\text{empty}}(P') \subseteq L_{\text{final}}(P)$. 
Proof of $L_{\text{final}}(P) \subseteq L_{\text{empty}}(P')$ 

Assume $w \in L_{\text{final}}(P)$. 

Then 
\[ (q_0, Z_0) \xrightarrow{w}^* (q, z) \text{ for some } q \in F, \quad z \in \Gamma^* \]

Then 
\[ (q'_0, Z'_0) \xrightarrow{\epsilon}^0 (q_0, Z_0 Z'_0) \xrightarrow{w}^0 (q, zZ'_0) \xrightarrow{\epsilon}^0 (q_{\text{final}}, zZ'_0) \xrightarrow{\epsilon}^0 (q_{\text{final}}, \epsilon) \]

so $w \in L_{\text{empty}}(P')$. 
Proof of $L_{empty}(P') \subseteq L_{final}(P)$

Assume $w \in L_{empty}(P')$.

Then

$$(q_0', Z_0') \xrightarrow{w}^* (q, \epsilon) \text{ for some } q \in Q', \quad z \in \Gamma'^*$$

- Since $Z_0' \neq \epsilon$, the first step must have been

  $$(q_0', Z_0') \xrightarrow{\epsilon} (q_0, Z_0Z_0')$$

- Then there must have been some transitions inherited from $P$.

- Since there are no transitions of this kind with stack symbol $Z_0'$, the symbol $Z_0'$ must have stayed at the bottom.

- So we never reached an empty state while staying continuously in states in $Q$. 
II.6.2. Equivalence of Final-State and Empty-Stack-PDAs

Proof of $L_{\text{empty}}(P') \subseteq L_{\text{final}}(P)$

$$(q'_0, Z'_0) \xrightarrow{\epsilon} (q_0, Z_0Z'_0) \xrightarrow{w}^*(q, \epsilon)$$

- Let $(q', zZ'_0)$ be the last configuration, where we stayed continuously since reaching state $q_0$ in a state in $Q$.
- Then we have
  $$(q'_0, Z'_0) \xrightarrow{\epsilon} (q_0, Z_0Z'_0) \xrightarrow{w_0}^0 (q', zZ'_0) \xrightarrow{a} (q'', z') \xrightarrow{w_1}^0 (q, \epsilon)$$

  s.t. $w = w_0 aw_1$,

  $$(q_0, Z_0) \xrightarrow{w_0}^* (q', z)$$

  and $q'' \not\in Q$.
- Then state $q'' = q_{\text{final}}$, $q' \in F$, $a = \epsilon$. Furthermore we obtain that
  $$(q_{\text{final}}, z') \xrightarrow{w_1}^0 (q, \epsilon)$$

  must have been a sequence of pops from the stack so we have $w_1 = \epsilon$, $q = q_{\text{final}}$. 
Proof of $L_{\text{empty}}(P') \subseteq L_{\text{final}}(P)$

Therefore $w = w_0aw_1 = w_0$, the overall transition was

$$(q'_0, Z'_0) \xrightarrow{\epsilon} (q_0, Z_0Z'_0) \xrightarrow{w} (q', zZ'_0) \xrightarrow{\epsilon} (q_{\text{final}}, zZ'_0) \xrightarrow{\epsilon} (q_{\text{final}}, \epsilon)$$

and we have

$$(q_0, Z_0) \xrightarrow{w} (q', z)$$

Therefore $w \in L(P)^*$. 
Proof of \((2) \Rightarrow (1)\)

Let \(P = (T, Q, \Gamma, q_0, Z_0, F, \delta)\).
The idea for constructing a final state PDA \(P'\) is as follows:

- \(P'\) keeps as before a special stack symbol \(Z_0'\) at the bottom of the stack.
- It operates as \(P\), until it observes that the top stack symbol is \(Z_0'\).
- This indicates that \(P\) would have reached an empty stack and therefore accepted the string.
- Therefore \(P'\) moves into a special accepting state \(q_{\text{final}}\) and terminates.
### II.6.2. Equivalence of Final-State and Empty-Stack-PDAs

#### Resulting PDA

<table>
<thead>
<tr>
<th>PDA</th>
<th>$P'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminals</td>
<td>$T$</td>
</tr>
<tr>
<td>states</td>
<td>$Q' := Q \cup {q_0', q_{final}}$</td>
</tr>
<tr>
<td>stack alphab.</td>
<td>$\Gamma' := {Z_0'} \cup \Gamma$</td>
</tr>
<tr>
<td>start state</td>
<td>$q_0'$</td>
</tr>
<tr>
<td>start stack</td>
<td>$Z_0'$</td>
</tr>
<tr>
<td>final</td>
<td>$q_{final}$</td>
</tr>
</tbody>
</table>
| transitions  | $(q_0', Z_0') \xrightarrow{\varepsilon} _0 (q_0, Z_0 Z_0')$
|              | $(q, Z) \xrightarrow{a} _0 (q', z')$ if $(q, Z) \xrightarrow{a} (q', z')$
|              | $(q, Z_0) \xrightarrow{\varepsilon} _0 (q_{final}, \varepsilon)$ for $q \in Q$ |
Proof of $L_{\text{empty}}(P) = L_{\text{final}}(P')$

Again we show

1. $L_{\text{empty}}(P) \subseteq L_{\text{final}}(P')$.
2. $L_{\text{final}}(P') \subseteq L_{\text{empty}}(P)$. 
Proof of $L_{\text{empty}}(P) \subseteq L_{\text{final}}(P')$

Assume $w \in L_{\text{empty}}(P)$.

Then

$$(q_0, Z_0) \xrightarrow{w}^* (q, \epsilon)$$

for some $q \in Q$

Then

$$(q_0', Z_0') \xrightarrow{\epsilon}_0 (q_0, Z_0 Z_0') \xrightarrow{w}_0^* (q, Z_0') \xrightarrow{\epsilon}_0 (q_{final}, \epsilon)$$

so $w \in L_{\text{final}}(P')$. 
II.6.2. Equivalence of Final-State and Empty-Stack-PDAs

Proof of $L_{\text{final}}(P') \subseteq L_{\text{empty}}(P)$

Assume $w \in L_{\text{final}}(P')$.

Then

$$(q'_0, Z'_0) \xrightarrow{w}^* (q_{\text{final}}, z) \text{ for some } z \in \Gamma'^*$$

- Since $q'_0$ is not accepting, the first step must have been

  $$(q'_0, Z'_0) \xrightarrow{\epsilon}^0 (q_0, Z_0 Z'_0)$$

- Then there must have been some transitions inherited from $P$.

- Since there are no transitions of this kind with stack symbol $Z'_0$, the symbol $Z'_0$ must have stayed at the bottom while having transitions inherited from $P$. 
II.6.2. Equivalence of Final-State and Empty-Stack-PDAs

Proof of \( L_{\text{final}}(P') \subseteq L_{\text{empty}}(P) \)

\[
(q'_0, Z'_0) \xrightarrow{\epsilon} (q_0, Z_0 Z'_0) \xrightarrow{w} (q_{\text{final}}, z)
\]

- Let \((q', zZ'_0)\) be the last configuration, where we stayed continuously since reaching state \(q_0\) in a state in \(Q\).
- Then we have

\[
(q'_0, Z'_0) \xrightarrow{\epsilon} (q_0, Z_0 Z'_0) \xrightarrow{w_0}^* (q', z' Z'_0) \xrightarrow{a} (q'', z'') \xrightarrow{w_1}^* (q_{\text{final}}, z)
\]

s.t. \(w = w_0 a w_1\),

\[
(q_0, Z_0) \xrightarrow{w_0}^* (q', z')
\]

and \(q'' \notin Q\).

- Then state \(q'' = q_{\text{final}}, a = \epsilon, (q_{\text{final}}, z) = (q'', z''), w_1 = \epsilon, \) and since \(z' \in \Gamma^*\) and we get into \(q_{\text{final}}\) only using stack symbol \(Z'_0\), we get \(z' = \epsilon\).
II.6.2. Equivalence of Final-State and Empty-Stack-PDAs

Proof of $L_{\text{final}}(P') \subseteq L_{\text{empty}}(P)$

So $w = w_0aw_1 = w_0$, and we have

$$
(q_0', Z_0') \xrightarrow{\epsilon}^0 (q_0, Z_0Z_0') \xrightarrow{w}^* (q', Z_0') \xrightarrow{\epsilon}^0 (q_{\text{final}}, \epsilon)
$$

and

$$
(q_0, Z_0) \xrightarrow{w}^* (q', \epsilon)
$$

Therefore $w \in L(P)^*$. 
II.6.1. Push Down Automata

II.6.2. Equivalence of Final-State and Empty-Stack-PDAs

II.6.3. Equivalence of CFG and PDA
Theorem (Repetition from Main Slides)

**Theorem**

Let $L$ be a language. The following are equivalent:

1. $L = L(G)$ for a CFG $G$.
2. $L = L(P)$ for an empty stack PDA $P$. 

LL Parsing

See main slides.
PDA based on LR-Parsing

- **LR-parsing** stands for left-to-right parsing based on a rightmost derivation.
- LR parsing constructs a rightmost derivation bottom up.
- It is therefore an example of a **bottom up parser**.
Principles of LR Parsing

- Remember that for LL parsing we had the invariant
  - Having consumed string \( w \) and stack \( v \) then \( S \Rightarrow^* wv \).
- For LR parsing the invariant will be
  - After having consumed \( w \) and having stack \( v \) then \( v^R \Rightarrow^* w \).
    So from the stack in reverse order we can derive the string consumed so far.

- Since we are generating a nondeterministic PDA, we don’t need information to decide which action to take.

- LR parsers constructed by compiler generators are deterministic. In order to allow a decision, they put between the symbols from \( L \cup N \) elements representing a state, which gives information about what is on the stack in order to give the information needed to decide which action is to be taken.
Whereas LL parsing has no state, for LR parsing we need a state, since we need sometimes to pop several elements from the stack in sequence in order to replace them by a nonterminal.

This state will occur in the final PDA.
Initially we haven’t consumed anything yet, but we need a start stack symbol. We call this symbol $Z_0$.

This stack symbol will stay at the bottom of the stack, and will only be popped if we accept the string.

So the invariant needs to be modified in so far that from the stack excluding this bottom symbol in reverse order we can derive the string consumed so far.
PDA based on LR-Parsing

- We take as example the same grammar as before, so we have the grammar with the rules

\[
\begin{align*}
S & \rightarrow AC \\
A & \rightarrow aAb \\
A & \rightarrow ab \\
C & \rightarrow cCd \\
C & \rightarrow cd
\end{align*}
\]

- We consider a right-most derivation

\[
S \Rightarrow AC \Rightarrow AcCd \Rightarrow Accdd \Rightarrow aAbccdd \Rightarrow aabbcdd
\]
Example of LR Parsing

\[ S \rightarrow AC \]
\[ A \rightarrow aAb \quad A \rightarrow ab \]
\[ C \rightarrow cCd \quad C \rightarrow cd \]

\[ S \Rightarrow AC \Rightarrow AcCd \Rightarrow Accdd \Rightarrow aAbccdd \Rightarrow aabbccdd \]

- We start on our PDA with stack \( Z_0 \) for the start symbol.
- We consume the first letter \( a \) and push it on the stack.
  - So we have stack \( aZ_0 \).
- We consume the next letters \( a, b \) and push them on the stack.
  - So we have stack \( baaZ_0 \).
- Now we reduce \( ba \) to \( A \) by using the rule \( A \rightarrow ab \).
- So we pop \( a \) and \( b \) from the stack and push \( A \) on to it.
  - So we have stack \( AaZ_0 \).
- We consume letter \( b \) and push it on the stack.
  - We have stack \( bAaZ_0 \).
- Now we reduce \( bAa \) to \( A \) by using the rule \( A \rightarrow aAb \).
- So we pop \( bAa \) from the stack and replace it by \( A \).
  - We have stack \( AZ_0 \).
Example of LR Parsing

\[ S \rightarrow AC \]
\[ A \rightarrow aAb \quad A \rightarrow ab \]
\[ C \rightarrow cCd \quad C \rightarrow cd \]
\[ S \Rightarrow AC \Rightarrow AcCd \Rightarrow Accdd \Rightarrow aAbccdd \Rightarrow aabbccdd \]

- We have stack \( AZ_0 \).
- We consume the letters \( c, c, d \) and push them on the stack.
  - So we have stack \( dccAZ_0 \).
- Now we reduce \( dc \) to \( C \) by using the rule \( C \rightarrow cd \).
- So we pop \( d \) and \( c \) from the stack and push \( C \) on to it.
  - So we have stack \( CcAZ_0 \).
- We consume letter \( d \) and push it on the stack.
  - We have stack \( dCcAZ_0 \).
- Now we reduce \( dCc \) to \( C \) by using the rule \( C \rightarrow dCc \).
- So we pop \( dCc \) from the stack and replace it by \( C \).
  - We have stack \( CAZ_0 \).
Example of LR Parsing

\[ S \rightarrow AC \]
\[ A \rightarrow aAb \quad A \rightarrow ab \]
\[ C \rightarrow cCd \quad C \rightarrow cd \]
\[ S \Rightarrow AC \Rightarrow AcCd \Rightarrow Accdd \Rightarrow aAbccdd \Rightarrow aabbccdd \]

- We have stack \( CAZ_0 \).
- Now we reduce \( CA \) to \( S \) by using the rule \( S \rightarrow AC \).
- So we pop \( CA \) from the stack and replace it by \( S \).
  - We have stack \( SZ_0 \).
- Now we see that we have the start symbol and below the bottom of the stack.
  - We pop those two symbols and accept with empty stack.
In the above example we have optimized the PDA by when using a rule immediately popping a symbol, and when the last symbol is popped immediately replacing it by the non terminal to which it was reduced.

It is easier to present the PDA without this optimization, and we do so in the following.

This means that we have as well states $A \rightarrow w \cdot$, and $A \rightarrow \cdot w$, for the beginning and end of a reduce sequence (where we replace a stack consisting of $w$ by $A$).
Resulting PDA (LR Parsing)

- We obtain a PDA with states.
- Stack symbols are $\Gamma := \{Z_0\} \cup T \cup N$.
- States are
  - $q_0$,
  - $A \rightarrow vw$ where $A \rightarrow vw$ is a production,
  - $\rightarrow Z_0 \cdot S$.
- Start state is $q_0$, start stack symbol is $Z_0$. 
Resulting PDA (LR Parsing)

We obtain the following transitions \((a \in T, Z \in \Gamma, w, v \in (T \cup N)^*, A, B \in N, A \rightarrow w, B \rightarrow uZv \in P)\)

\[
\begin{align*}
(q_0, Z) & \rightarrow a (q_0, aZ) \\
(q_0, Z) & \rightarrow \epsilon (A \rightarrow w \cdot, Z) \\
(B \rightarrow uZ \cdot v, Z) & \rightarrow \epsilon (B \rightarrow u \cdot Zv, \epsilon) \\
(A \rightarrow w, Z) & \rightarrow \epsilon (q_0, AZ) \\
(q_0, S) & \rightarrow \epsilon (\rightarrow Z_0 \cdot S, \epsilon) \\
(\rightarrow Z_0 \cdot S, Z_0) & \rightarrow \epsilon (q_0, \epsilon)
\end{align*}
\]
Correctness of the LL Parser

We are going to show that for the LL Parser $P$ we had given $L(P) = L(G)$. We repeat the definition of $P$:

<table>
<thead>
<tr>
<th>PDA</th>
<th>$P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminals</td>
<td>$T$</td>
</tr>
<tr>
<td>states</td>
<td>single state</td>
</tr>
<tr>
<td>stack alphab.</td>
<td>$T \cup N$</td>
</tr>
<tr>
<td>start stack</td>
<td>$S$</td>
</tr>
<tr>
<td>transitions</td>
<td>$A \xrightarrow{\epsilon} w$ if $A \rightarrow w \in P$</td>
</tr>
<tr>
<td></td>
<td>$a \xrightarrow{a} \epsilon$ if $a \in T$</td>
</tr>
</tbody>
</table>
Proof of $L(P) \subseteq L(G)$

- Remember that because of single state configurations are given by the stack only.
- We show by induction on the length of the derivation:
  - If in $P$ we have for $x \in T^*, w \in (T \cup N)^*$
    \[ S \xrightarrow{x}^* w \]
    then we have in $G$
    \[ S \Rightarrow^* xw \]
- Then we get
  
  $w \in L(P)$ implies $S \xrightarrow{w}^* \epsilon$
  implies $S \Rightarrow^* w$
  implies $w \in L(G)$

so $L(P) \subseteq L(G)$. 
II.6.3. Equivalence of CFG and PDA

Proof of $L(P) \subseteq L(G)$

Show:
$S \xrightarrow[^*]{} w$ implies $S \Rightarrow[^*] xw$

- Base case: Length zero.
  Then $x = \epsilon$, $S = w$, and we trivially get $S \Rightarrow[^*] xw$.

- Induction step.
  Assume
  
  $S \xrightarrow[^{x_1}]{} w_1 \xrightarrow[^a]{} w$

  where $w_1 \in (T \cup N)^*$, $x_1 \in T^*$, $a \in T \cup \{\epsilon\}$, $x = x_1 a$.

  - Case 1: $w_1 = Aw_2$, $A \rightarrow u$, $a = \epsilon$, $w = uw_2$, $x = x_1$ and the transition is
    
    $S \xrightarrow[^x]{} w_1 = Aw_2 \xrightarrow[^\epsilon]{} uw_2$

    By IH
    
    $S \Rightarrow[^*] xw_1 = xAw_2$

    and by $A \rightarrow u$ we have
    
    $S \Rightarrow[^*] xw_1 = xAw_2 \Rightarrow xuw_2 = xw$
Proof of $L(P) \subseteq L(G)$

Show:

$S \xrightarrow{x}^* w$ implies $S \Rightarrow^* xw$

We have $S \xrightarrow{x_1}^* w_1 \xrightarrow{a} w$

- Case 2: $w_1 = aw$, $a \in T$, and the transition is

  $$S \xrightarrow{x_1}^* w_1 = aw \xrightarrow{a} w$$

Then by IH

  $$S \Rightarrow^* x_1 w_1 = x_1 aw = xw$$
Proof of $L(G) \subseteq L(P)$

- We show by induction on the length of the left-most derivation:
  - If in $G$ we have for $x \in T^*, A \in N, w \in (T \cup N)^*$
    \[ S \Rightarrow^* xAw \]
    for a left-most derivation, then we have in $P$
    \[ S \xrightarrow{x}^* Aw \]
  - If in $G$ we have for $x \in T^*$ for a left-most derivation
    \[ S \Rightarrow^* x \]
    then we have in $P$
    \[ S \xrightarrow{x}^* \epsilon \]
  - Then we get
    \[ w \in L(G) \quad \text{implies} \quad S \Rightarrow^* w \]
    \[ \text{implies} \quad S \xrightarrow{w}^* \epsilon \]
    \[ \text{implies} \quad w \in L(P) \]

so $L(G) \subseteq L(P)$.
Proof of $L(G) \subseteq L(P)$

Show:
\[ S \Rightarrow^* xAw \implies S \xrightarrow{x}^* Aw \]
\[ S \Rightarrow^* x \implies S \xrightarrow{x}^* \epsilon \]

- Base case: Length zero.
  Then $x = w = \epsilon$, $S = A$, and we trivially get $S \xrightarrow{\epsilon}^* Aw$

- Induction step.
  Assume
  \[ S \Rightarrow^* x_1 Bw_1 \Rightarrow xAw \text{ or } S \Rightarrow^* x_1 Bw_1 \Rightarrow x \]
  where $x_1 \in T^*$, $B \in N$, $w_1 \in (T \cup N)^*$.
  By IH
  \[ S \xrightarrow{x_1}^* Bw_1 \]
Proof of $L(G) \subseteq L(P)$

Show:
$S \Rightarrow^* xAw$ implies $S \xrightarrow{x}^* Aw$
$S \Rightarrow^* x$ implies $S \xrightarrow{x}^* \epsilon$

We have
$S \Rightarrow^* x_1Bw_1 \Rightarrow xAw$ or $S \Rightarrow^* x_1Bw_1 \Rightarrow x$
and by IH $S \xrightarrow{x_1}^* Bw_1$.

- Case 1: $S \Rightarrow^* x_1Bw_1 \Rightarrow xAw$. Then there exist a production $B \rightarrow u$, $x_1uw_1 = xAw$. Therefore there exists $x_2$ s.t. $x = x_1x_2$. Now we get

$$S \xrightarrow{x_1}^* Bw_1 \xrightarrow{\epsilon} uw_1 = x_2Aw \xrightarrow{x_2}^* Aw$$

and by $x = x_1x_2$ follows the assertion.
Proof of $L(P) \subseteq L(G)$

Show:
$S \Rightarrow^* xAw$ implies $S \xrightarrow{x}^* Aw$
$S \Rightarrow^* x$ implies $S \xrightarrow{x}^* \epsilon$

We have
$S \Rightarrow^* x_1 B w_1 \Rightarrow xAw$ or $S \Rightarrow^* x_1 B w_1 \Rightarrow x$
and by IH $S \xrightarrow{x_1}^* B w_1$

- Case 2: $S \Rightarrow^* x_1 B w_1 \Rightarrow x$.
  Then there exist a production $B \rightarrow u$, $x_1 u w_1 = x$.
  Now we get
  $$S \xrightarrow{x_1}^* B w_1 \xrightarrow{\epsilon} u w_1 \xrightarrow{u w_1}^* \epsilon$$
  and by $x_1 u w_1 = x$ follows the assertion.