

CS 373: Theory of Computation

Manoj Prabhakaran Mahesh Viswanathan

University of Illinois, Urbana-Champaign

Fall 2008

Normal Forms for Grammars

Normal Forms for Grammars

It is typically easier to work with a context free language if given a CFG in a **normal form**.

Normal Forms for Grammars

It is typically easier to work with a context free language if given a CFG in a **normal form**.

Normal Forms

A grammar is in a normal form if its production rules have a special structure:

Normal Forms for Grammars

It is typically easier to work with a context free language if given a CFG in a **normal form**.

Normal Forms

A grammar is in a normal form if its production rules have a special structure:

- **Chomsky Normal Form:** Productions are of the form $A \rightarrow BC$ or $A \rightarrow a$

Normal Forms for Grammars

It is typically easier to work with a context free language if given a CFG in a **normal form**.

Normal Forms

A grammar is in a normal form if its production rules have a special structure:

- **Chomsky Normal Form:** Productions are of the form $A \rightarrow BC$ or $A \rightarrow a$
- **Greibach Normal Form** Productions are of the form $A \rightarrow a\alpha$, where $\alpha \in V^*$

Normal Forms for Grammars

It is typically easier to work with a context free language if given a CFG in a **normal form**.

Normal Forms

A grammar is in a normal form if its production rules have a special structure:

- **Chomsky Normal Form:** Productions are of the form $A \rightarrow BC$ or $A \rightarrow a$
- **Greibach Normal Form** Productions are of the form $A \rightarrow a\alpha$, where $\alpha \in V^*$

If ϵ is in the language, we allow the rule $S \rightarrow \epsilon$. We will require that S does not appear on the right hand side of any rules.

Normal Forms for Grammars

It is typically easier to work with a context free language if given a CFG in a **normal form**.

Normal Forms

A grammar is in a normal form if its production rules have a special structure:

- **Chomsky Normal Form:** Productions are of the form $A \rightarrow BC$ or $A \rightarrow a$
- **Greibach Normal Form** Productions are of the form $A \rightarrow a\alpha$, where $\alpha \in V^*$

If ϵ is in the language, we allow the rule $S \rightarrow \epsilon$. We will require that S does not appear on the right hand side of any rules.

- Today: How to convert *any context-free grammar* to an equivalent grammar in the Chomsky Normal Form

Normal Forms for Grammars

It is typically easier to work with a context free language if given a CFG in a **normal form**.

Normal Forms

A grammar is in a normal form if its production rules have a special structure:

- **Chomsky Normal Form:** Productions are of the form $A \rightarrow BC$ or $A \rightarrow a$
- **Greibach Normal Form** Productions are of the form $A \rightarrow a\alpha$, where $\alpha \in V^*$

If ϵ is in the language, we allow the rule $S \rightarrow \epsilon$. We will require that S does not appear on the right hand side of any rules.

- Today: How to convert *any context-free grammar* to an equivalent grammar in the Chomsky Normal Form
- We will start with a series of simplifications...

Eliminating ϵ -productions

- Often would like to ensure that the length of the intermediate strings in a derivation are not longer than the final string derived

Eliminating ϵ -productions

- Often would like to ensure that the length of the intermediate strings in a derivation are not longer than the final string derived
- But a long intermediate string can lead to a short final string if there are ϵ -productions (rules of the form $A \rightarrow \epsilon$).

Eliminating ϵ -productions

- Often would like to ensure that the length of the intermediate strings in a derivation are not longer than the final string derived
- But a long intermediate string can lead to a short final string if there are ϵ -productions (rules of the form $A \rightarrow \epsilon$).
- Can we rewrite the grammar not to have ϵ -productions?

Eliminating ϵ -productions

- Often would like to ensure that the length of the intermediate strings in a derivation are not longer than the final string derived
- But a long intermediate string can lead to a short final string if there are ϵ -productions (rules of the form $A \rightarrow \epsilon$).
- Can we rewrite the grammar not to have ϵ -productions?

Eliminating ϵ -productions

Given a grammar G produce an equivalent grammar G' (i.e., $L(G) = L(G')$) such that G' has no rules of the form $A \rightarrow \epsilon$, except possibly $S \rightarrow \epsilon$, and S does not appear on the right hand side of any rule.

Eliminating ϵ -productions

- Often would like to ensure that the length of the intermediate strings in a derivation are not longer than the final string derived
- But a long intermediate string can lead to a short final string if there are ϵ -productions (rules of the form $A \rightarrow \epsilon$).
- Can we rewrite the grammar not to have ϵ -productions?

Eliminating ϵ -productions

Given a grammar G produce an equivalent grammar G' (i.e., $L(G) = L(G')$) such that G' has no rules of the form $A \rightarrow \epsilon$, except possibly $S \rightarrow \epsilon$, and S does not appear on the right hand side of any rule.

Note: If S can appear on the RHS of a rule, say $S \rightarrow SS$, then when there is the rule $S \rightarrow \epsilon$, we can again have long intermediate strings yielding short final strings.

Eliminating ϵ -productions

Definition: Nullable Variables

A variable A (of grammar G) is nullable if $A \xRightarrow{*} \epsilon$.

Eliminating ϵ -productions

Definition: Nullable Variables

A variable A (of grammar G) is nullable if $A \xRightarrow{*} \epsilon$.

How do you determine if a variable is nullable?

Eliminating ϵ -productions

Definition: Nullable Variables

A variable A (of grammar G) is nullable if $A \xRightarrow{*} \epsilon$.

How do you determine if a variable is nullable?

- If $A \rightarrow \epsilon$ is a production in G then A is nullable

Definition: Nullable Variables

A variable A (of grammar G) is nullable if $A \xRightarrow{*} \epsilon$.

How do you determine if a variable is nullable?

- If $A \rightarrow \epsilon$ is a production in G then A is nullable
- If $A \rightarrow B_1 B_2 \cdots B_k$ is a production and each B_i is nullable, then A is nullable.

Definition: Nullable Variables

A variable A (of grammar G) is nullable if $A \xRightarrow{*} \epsilon$.

How do you determine if a variable is nullable?

- If $A \rightarrow \epsilon$ is a production in G then A is nullable
- If $A \rightarrow B_1 B_2 \cdots B_k$ is a production and each B_i is nullable, then A is nullable.

Fixed point algorithm: Propagate the label of nullable until there is no change.

Eliminating ϵ -productions given nullables

Intuition: For every variable A in G have a variable A in G' such that $A \xRightarrow{*}_{G'} w$ iff $A \xRightarrow{*}_G w$ and $w \neq \epsilon$.

Eliminating ϵ -productions given nullables

Intuition: For every variable A in G have a variable A in G' such that $A \xRightarrow{*}_{G'} w$ iff $A \xRightarrow{*}_G w$ and $w \neq \epsilon$. For every rule $B \rightarrow CAD$ in G , where A is nullable, add two rules in G' :

Eliminating ϵ -productions given nullable

Intuition: For every variable A in G have a variable A in G' such that $A \xRightarrow{*}_{G'} w$ iff $A \xRightarrow{*}_G w$ and $w \neq \epsilon$. For every rule $B \rightarrow CAD$ in G , where A is nullable, add two rules in G' : $B \rightarrow CD$ and $B \rightarrow CAD$.

Eliminating ϵ -productions given nullables

Intuition: For every variable A in G have a variable A in G' such that $A \xRightarrow{*}_{G'} w$ iff $A \xRightarrow{*}_G w$ and $w \neq \epsilon$. For every rule $B \rightarrow CAD$ in G , where A is nullable, add two rules in G' : $B \rightarrow CD$ and $B \rightarrow CAD$.

Algorithm

- G' has same variables, except for a new start symbol S' .

Eliminating ϵ -productions given nullables

Intuition: For every variable A in G have a variable A in G' such that $A \xRightarrow{*}_{G'} w$ iff $A \xRightarrow{*}_G w$ and $w \neq \epsilon$. For every rule $B \rightarrow CAD$ in G , where A is nullable, add two rules in G' : $B \rightarrow CD$ and $B \rightarrow CAD$.

Algorithm

- G' has same variables, except for a new start symbol S' .
- For each rule $A \rightarrow X_1X_2 \cdots X_k$ in G , create rules $A \rightarrow \alpha_1\alpha_2 \cdots \alpha_k$ where

Eliminating ϵ -productions given nullables

Intuition: For every variable A in G have a variable A in G' such that $A \xRightarrow{*}_{G'} w$ iff $A \xRightarrow{*}_G w$ and $w \neq \epsilon$. For every rule $B \rightarrow CAD$ in G , where A is nullable, add two rules in G' : $B \rightarrow CD$ and $B \rightarrow CAD$.

Algorithm

- G' has same variables, except for a new start symbol S' .
- For each rule $A \rightarrow X_1 X_2 \cdots X_k$ in G , create rules $A \rightarrow \alpha_1 \alpha_2 \cdots \alpha_k$ where

$$\alpha_j = \begin{cases} X_j & \text{if } X_j \text{ is a non-nullable variable/terminal in } G \\ X_j \text{ or } \epsilon & \text{if } X_j \text{ is nullable in } G \end{cases}$$

and not all α_j are ϵ

Eliminating ϵ -productions given nullables

Intuition: For every variable A in G have a variable A in G' such that $A \xRightarrow{*}_{G'} w$ iff $A \xRightarrow{*}_G w$ and $w \neq \epsilon$. For every rule $B \rightarrow CAD$ in G , where A is nullable, add two rules in G' : $B \rightarrow CD$ and $B \rightarrow CAD$.

Algorithm

- G' has same variables, except for a new start symbol S' .
- For each rule $A \rightarrow X_1 X_2 \cdots X_k$ in G , create rules $A \rightarrow \alpha_1 \alpha_2 \cdots \alpha_k$ where

$$\alpha_i = \begin{cases} X_i & \text{if } X_i \text{ is a non-nullable variable/terminal in } G \\ X_i \text{ or } \epsilon & \text{if } X_i \text{ is nullable in } G \end{cases}$$

and not all α_i are ϵ

- Add rule $S' \rightarrow S$. If S nullable in G , add $S' \rightarrow \epsilon$ also.

Eliminating ϵ -productions given nullables

Correctness of the Algorithm

Eliminating ϵ -productions given nullables

Correctness of the Algorithm

- By construction, there are no rules of the form $A \rightarrow \epsilon$ in G' (except possibly $S' \rightarrow \epsilon$), and S' does not appear in the RHS of any rule.

Eliminating ϵ -productions given nullables

Correctness of the Algorithm

- By construction, there are no rules of the form $A \rightarrow \epsilon$ in G' (except possibly $S' \rightarrow \epsilon$), and S' does not appear in the RHS of any rule.
- $L(G) = L(G')$

Eliminating ϵ -productions given nullables

Correctness of the Algorithm

- By construction, there are no rules of the form $A \rightarrow \epsilon$ in G' (except possibly $S' \rightarrow \epsilon$), and S' does not appear in the RHS of any rule.
- $L(G) = L(G')$
 - $L(G') \subseteq L(G)$: For every rule $A \rightarrow w$ in G' , we have $A \xRightarrow{*}_G w$ (by expanding zero or more nullable variables in w to ϵ)

Eliminating ϵ -productions given nullables

Correctness of the Algorithm

- By construction, there are no rules of the form $A \rightarrow \epsilon$ in G' (except possibly $S' \rightarrow \epsilon$), and S' does not appear in the RHS of any rule.
- $L(G) = L(G')$
 - $L(G') \subseteq L(G)$: For every rule $A \rightarrow w$ in G' , we have $A \xRightarrow{*}_G w$ (by expanding zero or more nullable variables in w to ϵ)
 - $L(G) \subseteq L(G')$: If $\epsilon \in L(G)$, then $\epsilon \in L(G')$.

Eliminating ϵ -productions given nullables

Correctness of the Algorithm

- By construction, there are no rules of the form $A \rightarrow \epsilon$ in G' (except possibly $S' \rightarrow \epsilon$), and S' does not appear in the RHS of any rule.
- $L(G) = L(G')$
 - $L(G') \subseteq L(G)$: For every rule $A \rightarrow w$ in G' , we have $A \xRightarrow{*}_G w$ (by expanding zero or more nullable variables in w to ϵ)
 - $L(G) \subseteq L(G')$: If $\epsilon \in L(G)$, then $\epsilon \in L(G')$. If $A \xRightarrow{*}_G w \in \Sigma^+$, then by induction on the number of steps in the derivation, $A \xRightarrow{*}_{G'} w$. Base case: if $A \rightarrow w \in \Sigma^+$, then $A \rightarrow w$.

(Proof details skipped.)

Eliminating ϵ -productions

An Example

Example

Rules of grammar G be $S \rightarrow AB$; $A \rightarrow AaA|\epsilon$; and $B \rightarrow BbB|\epsilon$.

- Nullables in G are

Eliminating ϵ -productions

An Example

Example

Rules of grammar G be $S \rightarrow AB$; $A \rightarrow AaA|\epsilon$; and $B \rightarrow BbB|\epsilon$.

- Nullables in G are A, B and S
- Rules for grammar G' :
 - $S \rightarrow$

Eliminating ϵ -productions

An Example

Example

Rules of grammar G be $S \rightarrow AB$; $A \rightarrow AaA|\epsilon$; and $B \rightarrow BbB|\epsilon$.

- Nullables in G are A, B and S
- Rules for grammar G' :
 - $S \rightarrow AB|A|B$

Eliminating ϵ -productions

An Example

Example

Rules of grammar G be $S \rightarrow AB$; $A \rightarrow AaA|\epsilon$; and $B \rightarrow BbB|\epsilon$.

- Nullables in G are A, B and S
- Rules for grammar G' :
 - $S \rightarrow AB|A|B$
 - $A \rightarrow$

Eliminating ϵ -productions

An Example

Example

Rules of grammar G be $S \rightarrow AB$; $A \rightarrow AaA|\epsilon$; and $B \rightarrow BbB|\epsilon$.

- Nullables in G are A, B and S
- Rules for grammar G' :
 - $S \rightarrow AB|A|B$
 - $A \rightarrow AaA|aA|Aa|a$

Eliminating ϵ -productions

An Example

Example

Rules of grammar G be $S \rightarrow AB$; $A \rightarrow AaA|\epsilon$; and $B \rightarrow BbB|\epsilon$.

- Nullables in G are A, B and S
- Rules for grammar G' :
 - $S \rightarrow AB|A|B$
 - $A \rightarrow AaA|aA|Aa|a$
 - $B \rightarrow$

Eliminating ϵ -productions

An Example

Example

Rules of grammar G be $S \rightarrow AB$; $A \rightarrow AaA|\epsilon$; and $B \rightarrow BbB|\epsilon$.

- Nullables in G are A , B and S
- Rules for grammar G' :
 - $S \rightarrow AB|A|B$
 - $A \rightarrow AaA|aA|Aa|a$
 - $B \rightarrow BbB|bB|Bb|b$

Eliminating ϵ -productions

An Example

Example

Rules of grammar G be $S \rightarrow AB$; $A \rightarrow AaA|\epsilon$; and $B \rightarrow BbB|\epsilon$.

- Nullables in G are A, B and S
- Rules for grammar G' :
 - $S \rightarrow AB|A|B$
 - $A \rightarrow AaA|aA|Aa|a$
 - $B \rightarrow BbB|bB|Bb|b$
 - $S' \rightarrow$

Eliminating ϵ -productions

An Example

Example

Rules of grammar G be $S \rightarrow AB$; $A \rightarrow AaA|\epsilon$; and $B \rightarrow BbB|\epsilon$.

- Nullables in G are A , B and S
- Rules for grammar G' :
 - $S \rightarrow AB|A|B$
 - $A \rightarrow AaA|aA|Aa|a$
 - $B \rightarrow BbB|bB|Bb|b$
 - $S' \rightarrow S|\epsilon$

Eliminating Unit Productions

- Often would like to ensure that the number of steps in a derivation are not much more than the length of the string derived

Eliminating Unit Productions

- Often would like to ensure that the number of steps in a derivation are not much more than the length of the string derived
- But can have a long chain of derivation steps that make little or no “progress,” if the grammar has **unit productions** (rules of the form $A \rightarrow B$, where B is a non-terminal).

Eliminating Unit Productions

- Often would like to ensure that the number of steps in a derivation are not much more than the length of the string derived
- But can have a long chain of derivation steps that make little or no “progress,” if the grammar has **unit productions** (rules of the form $A \rightarrow B$, where B is a non-terminal).
 - Note: $A \rightarrow a$ is not a unit production

Eliminating Unit Productions

- Often would like to ensure that the number of steps in a derivation are not much more than the length of the string derived
- But can have a long chain of derivation steps that make little or no “progress,” if the grammar has **unit productions** (rules of the form $A \rightarrow B$, where B is a non-terminal).
 - Note: $A \rightarrow a$ is not a unit production
- Can we rewrite the grammar not to have unit-productions?

Eliminating unit-productions

Given a grammar G produce an equivalent grammar G' (i.e., $L(G) = L(G')$) such that G' has no rules of the form $A \rightarrow B$ where $B \in V'$.

Eliminating Unit Productions

Unit Productions

Unit productions can play an important role in designing grammars:

Eliminating Unit Productions

Unit Productions

Unit productions can play an important role in designing grammars:

- While eliminating ϵ -productions we added a rule $S' \rightarrow S$. This is a unit production.

Eliminating Unit Productions

Unit Productions

Unit productions can play an important role in designing grammars:

- While eliminating ϵ -productions we added a rule $S' \rightarrow S$. This is a unit production.
- We have used unit productions in building an unambiguous grammar:

$$I \rightarrow a \mid b \mid la \mid lb$$

$$N \rightarrow 0 \mid 1 \mid N0 \mid N1 \mid -N \mid +N$$

$$F \rightarrow I \mid N \mid (E)$$

$$T \rightarrow F \mid T * F$$

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

Eliminating Unit Productions

Unit Productions

Unit productions can play an important role in designing grammars:

- While eliminating ϵ -productions we added a rule $S' \rightarrow S$. This is a unit production.
- We have used unit productions in building an unambiguous grammar:

$$I \rightarrow a \mid b \mid la \mid lb$$

$$N \rightarrow 0 \mid 1 \mid N0 \mid N1 \mid -N \mid +N$$

$$F \rightarrow I \mid N \mid (E)$$

$$T \rightarrow F \mid T * F$$

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

But as we shall see now, they can be (safely) eliminated

Eliminating Unit Productions

Basic Idea

Introduce new “look-ahead” productions to replace unit productions: look ahead to see where the unit production (or a chain of unit productions) leads to and add a rule to directly go there.

Eliminating Unit Productions

Basic Idea

Introduce new “look-ahead” productions to replace unit productions: look ahead to see where the unit production (or a chain of unit productions) leads to and add a rule to directly go there.

Example

$$E \rightarrow T \rightarrow F \rightarrow I \rightarrow a|b|Ia|Ib.$$

Eliminating Unit Productions

Basic Idea

Introduce new “look-ahead” productions to replace unit productions: look ahead to see where the unit production (or a chain of unit productions) leads to and add a rule to directly go there.

Example

$E \rightarrow T \rightarrow F \rightarrow I \rightarrow a|b|la|lb$. So introduce new rules

$E \rightarrow a|b|la|lb$

Eliminating Unit Productions

Basic Idea

Introduce new “look-ahead” productions to replace unit productions: look ahead to see where the unit production (or a chain of unit productions) leads to and add a rule to directly go there.

Example

$E \rightarrow T \rightarrow F \rightarrow I \rightarrow a|b|la|lb$. So introduce new rules

$E \rightarrow a|b|la|lb$

But what if the grammar has **cycles of unit productions**?

Eliminating Unit Productions

Basic Idea

Introduce new “look-ahead” productions to replace unit productions: look ahead to see where the unit production (or a chain of unit productions) leads to and add a rule to directly go there.

Example

$E \rightarrow T \rightarrow F \rightarrow I \rightarrow a|b|la|lb$. So introduce new rules

$E \rightarrow a|b|la|lb$

But what if the grammar has **cycles of unit productions**? For example, $A \rightarrow B|a$, $B \rightarrow C|b$ and $C \rightarrow A|c$. You cannot use the “look-ahead” approach, because then you will get into an infinite loop.

Eliminating Unit Productions

Basic Idea: Fixed

Algorithm

Eliminating Unit Productions

Basic Idea: Fixed

Algorithm

- 1 Determine pairs $\langle A, B \rangle$ such that $A \xRightarrow{*}_u B$, i.e., A derives B using only unit rules. Such pairs are called *unit pairs*.

Eliminating Unit Productions

Basic Idea: Fixed

Algorithm

- 1 Determine pairs $\langle A, B \rangle$ such that $A \xRightarrow{*}_u B$, i.e., A derives B using only unit rules. Such pairs are called *unit pairs*.
 - Easy to determine unit pairs:

Eliminating Unit Productions

Basic Idea: Fixed

Algorithm

- 1 Determine pairs $\langle A, B \rangle$ such that $A \xRightarrow{*}_u B$, i.e., A derives B using only unit rules. Such pairs are called *unit pairs*.
 - Easy to determine unit pairs: Make a directed graph with vertices = V , and edges = unit productions. $\langle A, B \rangle$ is a unit pair, if there is a directed path from A to B in the graph.

Eliminating Unit Productions

Basic Idea: Fixed

Algorithm

- 1 Determine pairs $\langle A, B \rangle$ such that $A \xRightarrow{*}_u B$, i.e., A derives B using only unit rules. Such pairs are called *unit pairs*.
 - Easy to determine unit pairs: Make a directed graph with vertices = V , and edges = unit productions. $\langle A, B \rangle$ is a unit pair, if there is a directed path from A to B in the graph.
- 2 If $\langle A, B \rangle$ is a unit pair, then add production rules $A \rightarrow \beta_1 | \beta_2 | \cdots | \beta_k$, where $B \rightarrow \beta_1 | \beta_2 | \cdots | \beta_k$ are *all the non-unit production rules of B*

Eliminating Unit Productions

Basic Idea: Fixed

Algorithm

- 1 Determine pairs $\langle A, B \rangle$ such that $A \xRightarrow{*}_u B$, i.e., A derives B using only unit rules. Such pairs are called *unit pairs*.
 - Easy to determine unit pairs: Make a directed graph with vertices = V , and edges = unit productions. $\langle A, B \rangle$ is a unit pair, if there is a directed path from A to B in the graph.
- 2 If $\langle A, B \rangle$ is a unit pair, then add production rules $A \rightarrow \beta_1 | \beta_2 | \cdots | \beta_k$, where $B \rightarrow \beta_1 | \beta_2 | \cdots | \beta_k$ are *all the non-unit production rules of B*
- 3 Remove all unit production rules.

Let G' be the grammar obtained from G using this algorithm.
Then $L(G') = L(G)$

Eliminating Unit Productions

$L(G) = L(G')$: Proof

Eliminating Unit Productions

$L(G) = L(G')$: Proof

- $L(G') \subseteq L(G)$: For every rule $A \rightarrow w$ in G' , we have $A \xRightarrow{*}_G w$ (by a sequence of zero or more unit productions followed by a nonunit production of G)

Eliminating Unit Productions

$L(G) = L(G')$: Proof

- $L(G') \subseteq L(G)$: For every rule $A \rightarrow w$ in G' , we have $A \xRightarrow{*}_G w$ (by a sequence of zero or more unit productions followed by a nonunit production of G)
- $L(G) \subseteq L(G')$: For $w \in L(G)$ consider a **leftmost derivation** $S \xRightarrow{*}_{\text{lm}} w$ in G .

Eliminating Unit Productions

$L(G) = L(G')$: Proof

- $L(G') \subseteq L(G)$: For every rule $A \rightarrow w$ in G' , we have $A \xRightarrow{*}_G w$ (by a sequence of zero or more unit productions followed by a nonunit production of G)
- $L(G) \subseteq L(G')$: For $w \in L(G)$ consider a **leftmost derivation** $S \xRightarrow{*}_{\text{lm}} w$ in G .
 - All these derivation steps are possible in G' also, except the ones using the unit productions of G .

Eliminating Unit Productions

$L(G) = L(G')$: Proof

- $L(G') \subseteq L(G)$: For every rule $A \rightarrow w$ in G' , we have $A \xRightarrow{*}_G w$ (by a sequence of zero or more unit productions followed by a nonunit production of G)
- $L(G) \subseteq L(G')$: For $w \in L(G)$ consider a **leftmost derivation** $S \xRightarrow{*}_{\text{lm}} w$ in G .
 - All these derivation steps are possible in G' also, except the ones using the unit productions of G .
 - Suppose $S \xRightarrow{*} xA\alpha \Rightarrow_1 xB\alpha \Rightarrow_2 \dots$, where \Rightarrow_1 corresponds to a unit rule.

Eliminating Unit Productions

$L(G) = L(G')$: Proof

- $L(G') \subseteq L(G)$: For every rule $A \rightarrow w$ in G' , we have $A \xRightarrow{*}_G w$ (by a sequence of zero or more unit productions followed by a nonunit production of G)
- $L(G) \subseteq L(G')$: For $w \in L(G)$ consider a **leftmost derivation** $S \xRightarrow{*}_{\text{lm}} w$ in G .
 - All these derivation steps are possible in G' also, except the ones using the unit productions of G .
 - Suppose $S \xRightarrow{*} xA\alpha \Rightarrow_1 xB\alpha \Rightarrow_2 \dots$, where \Rightarrow_1 corresponds to a unit rule. Then (in a leftmost derivation) \Rightarrow_2 must correspond to using a rule for B .

Eliminating Unit Productions

$L(G) = L(G')$: Proof

- $L(G') \subseteq L(G)$: For every rule $A \rightarrow w$ in G' , we have $A \xRightarrow{*}_G w$ (by a sequence of zero or more unit productions followed by a nonunit production of G)
- $L(G) \subseteq L(G')$: For $w \in L(G)$ consider a **leftmost derivation** $S \xRightarrow{*}_{\text{lm}} w$ in G .
 - All these derivation steps are possible in G' also, except the ones using the unit productions of G .
 - Suppose $S \xRightarrow{*} xA\alpha \Rightarrow_1 xB\alpha \Rightarrow_2 \dots$, where \Rightarrow_1 corresponds to a unit rule. Then (in a leftmost derivation) \Rightarrow_2 must correspond to using a rule for B .
 - So a leftmost derivation of w in G can be broken up into “big-steps” each consisting of zero or more unit productions on the leftmost variable, followed by a non-unit production.

Eliminating Unit Productions

$L(G) = L(G')$: Proof

- $L(G') \subseteq L(G)$: For every rule $A \rightarrow w$ in G' , we have $A \xRightarrow{*}_G w$ (by a sequence of zero or more unit productions followed by a nonunit production of G)
- $L(G) \subseteq L(G')$: For $w \in L(G)$ consider a **leftmost derivation** $S \xRightarrow{*}_{\text{lm}} w$ in G .
 - All these derivation steps are possible in G' also, except the ones using the unit productions of G .
 - Suppose $S \xRightarrow{*} xA\alpha \Rightarrow_1 xB\alpha \Rightarrow_2 \dots$, where \Rightarrow_1 corresponds to a unit rule. Then (in a leftmost derivation) \Rightarrow_2 must correspond to using a rule for B .
 - So a leftmost derivation of w in G can be broken up into “big-steps” each consisting of zero or more unit productions on the leftmost variable, followed by a non-unit production.
 - For each such “big-step” there is a single production rule in G' that yields the same result.

Eliminating Useless Symbols

Eliminating Useless Symbols

- Ideally one would like to use a compact grammar, with the fewest possible variables

Eliminating Useless Symbols

- Ideally one would like to use a compact grammar, with the fewest possible variables
- But a grammar may have “useless” variables which do not appear in any valid derivation

Eliminating Useless Symbols

- Ideally one would like to use a compact grammar, with the fewest possible variables
- But a grammar may have “useless” variables which do not appear in any valid derivation
- Can we identify all the useless variables and remove them from the grammar? (Note: there may still be other redundancies in the grammar.)

Eliminating Useless Symbols

- Ideally one would like to use a compact grammar, with the fewest possible variables
- But a grammar may have “useless” variables which do not appear in any valid derivation
- Can we identify all the useless variables and remove them from the grammar? (Note: there may still be other redundancies in the grammar.)

Definition

A symbol $X \in V \cup \Sigma$ is *useless* in a grammar $G = (V, \Sigma, S, P)$ if there is no derivation of the form $S \xRightarrow{*} \alpha X \beta \xRightarrow{*} w$ where $w \in \Sigma^*$ and $\alpha, \beta \in (V \cup \Sigma)^*$.

Eliminating Useless Symbols

- Ideally one would like to use a compact grammar, with the fewest possible variables
- But a grammar may have “useless” variables which do not appear in any valid derivation
- Can we identify all the useless variables and remove them from the grammar? (Note: there may still be other redundancies in the grammar.)

Definition

A symbol $X \in V \cup \Sigma$ is *useless* in a grammar $G = (V, \Sigma, S, P)$ if there is no derivation of the form $S \xRightarrow{*} \alpha X \beta \xRightarrow{*} w$ where $w \in \Sigma^*$ and $\alpha, \beta \in (V \cup \Sigma)^*$.

Removing useless symbols (and rules involving them) from a grammar does not change the language of the grammar

Eliminating Useless Symbols

Definition

A symbol $X \in V \cup \Sigma$ is *useless* in a grammar $G = (V, \Sigma, S, P)$ if there is no derivation of the form $S \xRightarrow{*} \alpha X \beta \xRightarrow{*} w$ where $w \in \Sigma^*$ and $\alpha, \beta \in (V \cup \Sigma)^*$.

Eliminating Useless Symbols

Definition

A symbol $X \in V \cup \Sigma$ is *useless* in a grammar $G = (V, \Sigma, S, P)$ if there is no derivation of the form $S \xRightarrow{*} \alpha X \beta \xRightarrow{*} w$ where $w \in \Sigma^*$ and $\alpha, \beta \in (V \cup \Sigma)^*$.

i.e., X is useless iff either

Eliminating Useless Symbols

Definition

A symbol $X \in V \cup \Sigma$ is *useless* in a grammar $G = (V, \Sigma, S, P)$ if there is no derivation of the form $S \xRightarrow{*} \alpha X \beta \xRightarrow{*} w$ where $w \in \Sigma^*$ and $\alpha, \beta \in (V \cup \Sigma)^*$.

i.e., X is useless iff either

Type 1: X is not “reachable” from S (i.e., no α, β such that $S \xRightarrow{*} \alpha X \beta$),

Eliminating Useless Symbols

Definition

A symbol $X \in V \cup \Sigma$ is *useless* in a grammar $G = (V, \Sigma, S, P)$ if there is no derivation of the form $S \xRightarrow{*} \alpha X \beta \xRightarrow{*} w$ where $w \in \Sigma^*$ and $\alpha, \beta \in (V \cup \Sigma)^*$.

i.e., X is useless iff either

- Type 1:** X is not “reachable” from S (i.e., no α, β such that $S \xRightarrow{*} \alpha X \beta$), or
- Type 2:** for all α, β such that $S \xRightarrow{*} \alpha X \beta$, either α , X or β cannot yield a string in Σ^* .

Eliminating Useless Symbols

Definition

A symbol $X \in V \cup \Sigma$ is *useless* in a grammar $G = (V, \Sigma, S, P)$ if there is no derivation of the form $S \xRightarrow{*} \alpha X \beta \xRightarrow{*} w$ where $w \in \Sigma^*$ and $\alpha, \beta \in (V \cup \Sigma)^*$.

i.e., X is useless iff either

Type 1: X is **not “reachable”** from S (i.e., no α, β such that $S \xRightarrow{*} \alpha X \beta$), or

Type 2: for all α, β such that $S \xRightarrow{*} \alpha X \beta$, either α, X or β cannot yield a string in Σ^* . i.e., either

Type 2a: X is **not “generating”** (i.e., no $w \in \Sigma^*$ such that $X \xRightarrow{*} w$),

Eliminating Useless Symbols

Definition

A symbol $X \in V \cup \Sigma$ is *useless* in a grammar $G = (V, \Sigma, S, P)$ if there is no derivation of the form $S \xRightarrow{*} \alpha X \beta \xRightarrow{*} w$ where $w \in \Sigma^*$ and $\alpha, \beta \in (V \cup \Sigma)^*$.

i.e., X is useless iff either

Type 1: X is **not “reachable”** from S (i.e., no α, β such that $S \xRightarrow{*} \alpha X \beta$), or

Type 2: for all α, β such that $S \xRightarrow{*} \alpha X \beta$, either α, X or β cannot yield a string in Σ^* . i.e., either

Type 2a: X is **not “generating”** (i.e., no $w \in \Sigma^*$ such that $X \xRightarrow{*} w$), or

Type 2b: α or β contains a non-generating symbol

Eliminating Useless Symbols

Algorithm

So, in order to remove useless symbols,

Eliminating Useless Symbols

Algorithm

So, in order to remove useless symbols,

- 1 First remove all symbols that are not generating (Type 2a)

Eliminating Useless Symbols

Algorithm

So, in order to remove useless symbols,

- 1 First remove all symbols that are not generating (Type 2a)
 - If X was useless, but reachable and generating (i.e., Type 2b) then X becomes unreachable after this step

Eliminating Useless Symbols

Algorithm

So, in order to remove useless symbols,

- 1 First remove all symbols that are not generating (Type 2a)
 - If X was useless, but reachable and generating (i.e., Type 2b) then X becomes unreachable after this step
 - Type 2b: for all α, β such that $S \xRightarrow{*} \alpha X \beta$, α or β contains a non-generating symbol.

Eliminating Useless Symbols

Algorithm

So, in order to remove useless symbols,

- 1 First remove all symbols that are not generating (Type 2a)
 - If X was useless, but reachable and generating (i.e., Type 2b) then X becomes unreachable after this step
 - Type 2b: for all α, β such that $S \xRightarrow{*} \alpha X \beta$, α or β contains a non-generating symbol. Then in the new grammar all such derivations disappear (because some variable in α or β is removed).

Eliminating Useless Symbols

Algorithm

So, in order to remove useless symbols,

- 1 First remove all symbols that are not generating (Type 2a)
 - If X was useless, but reachable and generating (i.e., Type 2b) then X becomes unreachable after this step
 - Type 2b: for all α, β such that $S \xRightarrow{*} \alpha X \beta$, α or β contains a non-generating symbol. Then in the new grammar all such derivations disappear (because some variable in α or β is removed).
- 2 Next remove all unreachable symbols in the new grammar.

Eliminating Useless Symbols

Algorithm

So, in order to remove useless symbols,

- 1 First remove all symbols that are not generating (Type 2a)
 - If X was useless, but reachable and generating (i.e., Type 2b) then X becomes unreachable after this step
 - Type 2b: for all α, β such that $S \xRightarrow{*} \alpha X \beta$, α or β contains a non-generating symbol. Then in the new grammar all such derivations disappear (because some variable in α or β is removed).
- 2 Next remove all unreachable symbols in the new grammar.
 - Removes Type 1 (originally unreachable) and Type 2b useless symbols now

Eliminating Useless Symbols

Algorithm

So, in order to remove useless symbols,

- 1 First remove all symbols that are not generating (Type 2a)
 - If X was useless, but reachable and generating (i.e., Type 2b) then X becomes unreachable after this step
 - Type 2b: for all α, β such that $S \xRightarrow{*} \alpha X \beta$, α or β contains a non-generating symbol. Then in the new grammar all such derivations disappear (because some variable in α or β is removed).
- 2 Next remove all unreachable symbols in the new grammar.
 - Removes Type 1 (originally unreachable) and Type 2b useless symbols now

Doesn't remove any useful symbol in either step (Why?)

Eliminating Useless Symbols

Algorithm

So, in order to remove useless symbols,

- 1 First remove all symbols that are not generating (Type 2a)
 - If X was useless, but reachable and generating (i.e., Type 2b) then X becomes unreachable after this step
 - Type 2b: for all α, β such that $S \xRightarrow{*} \alpha X \beta$, α or β contains a non-generating symbol. Then in the new grammar all such derivations disappear (because some variable in α or β is removed).
- 2 Next remove all unreachable symbols in the new grammar.
 - Removes Type 1 (originally unreachable) and Type 2b useless symbols now

Doesn't remove any useful symbol in either step (Why?)

Only remains to show how to do the two steps in this algorithm

Eliminating Useless Symbols

Generating and Reachable Symbols

The set of generating symbols

Eliminating Useless Symbols

Generating and Reachable Symbols

The set of generating symbols

- If $A \rightarrow x$, where $x \in \Sigma^*$, is a production then A is generating

Eliminating Useless Symbols

Generating and Reachable Symbols

The set of generating symbols

- If $A \rightarrow x$, where $x \in \Sigma^*$, is a production then A is generating
- If $A \rightarrow \gamma$ is a production and all variables in γ are generating, then A is generating.

Eliminating Useless Symbols

Generating and Reachable Symbols

The set of generating symbols

- If $A \rightarrow x$, where $x \in \Sigma^*$, is a production then A is generating
- If $A \rightarrow \gamma$ is a production and all variables in γ are generating, then A is generating.

The set of reachable symbols

Eliminating Useless Symbols

Generating and Reachable Symbols

The set of generating symbols

- If $A \rightarrow x$, where $x \in \Sigma^*$, is a production then A is generating
- If $A \rightarrow \gamma$ is a production and all variables in γ are generating, then A is generating.

The set of reachable symbols

- S is reachable

Eliminating Useless Symbols

Generating and Reachable Symbols

The set of generating symbols

- If $A \rightarrow x$, where $x \in \Sigma^*$, is a production then A is generating
- If $A \rightarrow \gamma$ is a production and all variables in γ are generating, then A is generating.

The set of reachable symbols

- S is reachable
- If A is reachable and $A \rightarrow \alpha B \beta$ is a production, then B is reachable

Eliminating Useless Symbols

Generating and Reachable Symbols

The set of generating symbols

- If $A \rightarrow x$, where $x \in \Sigma^*$, is a production then A is generating
- If $A \rightarrow \gamma$ is a production and all variables in γ are generating, then A is generating.

The set of reachable symbols

- S is reachable
- If A is reachable and $A \rightarrow \alpha B \beta$ is a production, then B is reachable

Fixed point algorithm: Propagate the label (generating or reachable) until no change.

The Three Simplifications, Together

The Three Simplifications, Together

Given a grammar G , such that $L(G) \neq \emptyset$, we can find a grammar G' such that $L(G') = L(G)$ and G' has no ϵ -productions (except possibly $S \rightarrow \epsilon$), unit productions, or useless symbols, and S does not appear in the RHS of any rule.

The Three Simplifications, Together

Given a grammar G , such that $L(G) \neq \emptyset$, we can find a grammar G' such that $L(G') = L(G)$ and G' has no ϵ -productions (except possibly $S \rightarrow \epsilon$), unit productions, or useless symbols, and S does not appear in the RHS of any rule.

Proof.

Apply the following 3 steps **in order**:

- 1 Eliminate ϵ -productions
- 2 Eliminate unit productions
- 3 Eliminate useless symbols. □

The Three Simplifications, Together

Given a grammar G , such that $L(G) \neq \emptyset$, we can find a grammar G' such that $L(G') = L(G)$ and G' has no ϵ -productions (except possibly $S \rightarrow \epsilon$), unit productions, or useless symbols, and S does not appear in the RHS of any rule.

Proof.

Apply the following 3 steps **in order**:

- 1 Eliminate ϵ -productions
- 2 Eliminate unit productions
- 3 Eliminate useless symbols.

Note: Applying the steps in a different order may result in a grammar not having all the desired properties.

Chomsky Normal Form

Proposition

For any non-empty context-free language L , there is a grammar G , such that $L(G) = L$ and each rule in G is of the form

- 1 $A \rightarrow a$ where $a \in \Sigma$,

Proposition

For any non-empty context-free language L , there is a grammar G , such that $L(G) = L$ and each rule in G is of the form

- 1 $A \rightarrow a$ where $a \in \Sigma$, or
- 2 $A \rightarrow BC$ where neither B nor C is the start symbol,

Proposition

For any non-empty context-free language L , there is a grammar G , such that $L(G) = L$ and each rule in G is of the form

- 1 $A \rightarrow a$ where $a \in \Sigma$, or
- 2 $A \rightarrow BC$ where neither B nor C is the start symbol, or
- 3 $S \rightarrow \epsilon$ where S is the start symbol (iff $\epsilon \in L$)

Proposition

For any non-empty context-free language L , there is a grammar G , such that $L(G) = L$ and each rule in G is of the form

- 1 $A \rightarrow a$ where $a \in \Sigma$, or
- 2 $A \rightarrow BC$ where neither B nor C is the start symbol, or
- 3 $S \rightarrow \epsilon$ where S is the start symbol (iff $\epsilon \in L$)

Furthermore, G has no useless symbols.

Chomsky Normal Form

Outline of Normalization

Given $G = (V, \Sigma, S, P)$, convert to CNF

- Let $G' = (V', \Sigma, S, P')$ be the grammar obtained after eliminating ϵ -productions, unit productions, and useless symbols from G .

Chomsky Normal Form

Outline of Normalization

Given $G = (V, \Sigma, S, P)$, convert to CNF

- Let $G' = (V', \Sigma, S, P')$ be the grammar obtained after eliminating ϵ -productions, unit productions, and useless symbols from G .
- If $A \rightarrow x$ is a rule of G' , where $|x| = 0$, then A must be S (because G' has no other ϵ -productions). If $A \rightarrow x$ is a rule of G' , where $|x| = 1$, then $x \in \Sigma$ (because G' has no unit productions). In either case $A \rightarrow x$ is in a valid form.

Chomsky Normal Form

Outline of Normalization

Given $G = (V, \Sigma, S, P)$, convert to CNF

- Let $G' = (V', \Sigma, S, P')$ be the grammar obtained after eliminating ϵ -productions, unit productions, and useless symbols from G .
- If $A \rightarrow x$ is a rule of G' , where $|x| = 0$, then A must be S (because G' has no other ϵ -productions). If $A \rightarrow x$ is a rule of G' , where $|x| = 1$, then $x \in \Sigma$ (because G' has no unit productions). In either case $A \rightarrow x$ is in a valid form.
- All remaining productions are of form $A \rightarrow X_1 X_2 \cdots X_n$ where $X_i \in V' \cup \Sigma$, $n \geq 2$ (and S does not occur in the RHS).

Chomsky Normal Form

Outline of Normalization

Given $G = (V, \Sigma, S, P)$, convert to CNF

- Let $G' = (V', \Sigma, S, P')$ be the grammar obtained after eliminating ϵ -productions, unit productions, and useless symbols from G .
- If $A \rightarrow x$ is a rule of G' , where $|x| = 0$, then A must be S (because G' has no other ϵ -productions). If $A \rightarrow x$ is a rule of G' , where $|x| = 1$, then $x \in \Sigma$ (because G' has no unit productions). In either case $A \rightarrow x$ is in a valid form.
- All remaining productions are of form $A \rightarrow X_1 X_2 \cdots X_n$ where $X_i \in V' \cup \Sigma$, $n \geq 2$ (and S does not occur in the RHS). We will put these rules in the right form by applying the following two transformations:
 - 1 Make the RHS consist only of variables
 - 2 Make the RHS be of length 2.

Chomsky Normal Form

Make the RHS consist only of variables

Let $A \rightarrow X_1 X_2 \cdots X_n$, with X_i being either a variable or a terminal.
We want rules where all the X_i are variables.

Chomsky Normal Form

Make the RHS consist only of variables

Let $A \rightarrow X_1 X_2 \cdots X_n$, with X_i being either a variable or a terminal.
We want rules where all the X_i are variables.

Example

Consider $A \rightarrow BbCdefG$. How do you remove the terminals?

Chomsky Normal Form

Make the RHS consist only of variables

Let $A \rightarrow X_1 X_2 \cdots X_n$, with X_i being either a variable or a terminal.
We want rules where all the X_i are variables.

Example

Consider $A \rightarrow BbCdefG$. How do you remove the terminals?
For each $a, b, c \dots \in \Sigma$ add variables X_a, X_b, X_c, \dots with productions $X_a \rightarrow a, X_b \rightarrow b, \dots$. Then replace the production $A \rightarrow BbCdefG$ by $A \rightarrow BX_bCX_dX_eX_fG$

Chomsky Normal Form

Make the RHS consist only of variables

Let $A \rightarrow X_1 X_2 \cdots X_n$, with X_i being either a variable or a terminal.
We want rules where all the X_i are variables.

Example

Consider $A \rightarrow BbCdefG$. How do you remove the terminals?
For each $a, b, c \dots \in \Sigma$ add variables X_a, X_b, X_c, \dots with productions $X_a \rightarrow a, X_b \rightarrow b, \dots$. Then replace the production $A \rightarrow BbCdefG$ by $A \rightarrow BX_bCX_dX_eX_fG$

For every $a \in \Sigma$

- 1 Add a new variable X_a
- 2 In every rule, if a occurs in the RHS, replace it by X_a
- 3 Add a new rule $X_a \rightarrow a$

Chomsky Normal Form

Make the RHS be of length 2

- Now all productions are of the form $A \rightarrow a$ or $A \rightarrow B_1 B_2 \cdots B_n$, where $n \geq 2$ and each B_i is a variable.

Chomsky Normal Form

Make the RHS be of length 2

- Now all productions are of the form $A \rightarrow a$ or $A \rightarrow B_1 B_2 \cdots B_n$, where $n \geq 2$ and each B_i is a variable.
- How do you eliminate rules of the form $A \rightarrow B_1 B_2 \cdots B_n$ where $n > 2$?

Chomsky Normal Form

Make the RHS be of length 2

- Now all productions are of the form $A \rightarrow a$ or $A \rightarrow B_1 B_2 \cdots B_n$, where $n \geq 2$ and each B_i is a variable.
- How do you eliminate rules of the form $A \rightarrow B_1 B_2 \cdots B_n$ where $n > 2$?
- Replace the rule by the following set of rules

$$\begin{aligned} A &\rightarrow B_1 B_{(2,n)} \\ B_{(2,n)} &\rightarrow B_2 B_{(3,n)} \\ B_{(3,n)} &\rightarrow B_3 B_{(4,n)} \\ &\vdots \\ B_{(n-1,n)} &\rightarrow B_{n-1} B_n \end{aligned}$$

where $B_{(i,n)}$ are “new” variables.

Chomsky Normal Form

An Example

Example

Convert: $S \rightarrow aA|bB|b$, $A \rightarrow Baa|ba$, $B \rightarrow bAAb|ab$, into Chomsky Normal Form.

Chomsky Normal Form

An Example

Example

Convert: $S \rightarrow aA|bB|b$, $A \rightarrow Baa|ba$, $B \rightarrow bAAb|ab$, into Chomsky Normal Form.

- 1 Eliminate ϵ -productions, unit productions, and useless symbols.

Chomsky Normal Form

An Example

Example

Convert: $S \rightarrow aA|bB|b$, $A \rightarrow Baa|ba$, $B \rightarrow bAAb|ab$, into Chomsky Normal Form.

- 1 Eliminate ϵ -productions, unit productions, and useless symbols. This grammar is already in the right form.

Chomsky Normal Form

An Example

Example

Convert: $S \rightarrow aA|bB|b$, $A \rightarrow Baa|ba$, $B \rightarrow bAAb|ab$, into Chomsky Normal Form.

- 1 Eliminate ϵ -productions, unit productions, and useless symbols. This grammar is already in the right form.
- 2 Remove terminals from the RHS of long rules.

Chomsky Normal Form

An Example

Example

Convert: $S \rightarrow aA|bB|b$, $A \rightarrow Baa|ba$, $B \rightarrow bAAb|ab$, into Chomsky Normal Form.

- 1 Eliminate ϵ -productions, unit productions, and useless symbols. This grammar is already in the right form.
- 2 Remove terminals from the RHS of long rules. New grammar is: $X_a \rightarrow a$, $X_b \rightarrow b$, $S \rightarrow X_aA|X_bB|b$, $A \rightarrow BX_aX_a|X_bX_a$, and $B \rightarrow X_bAAX_b|X_aX_b$

Chomsky Normal Form

An Example

Example

Convert: $S \rightarrow aA|bB|b$, $A \rightarrow Baa|ba$, $B \rightarrow bAAb|ab$, into Chomsky Normal Form.

- 1 Eliminate ϵ -productions, unit productions, and useless symbols. This grammar is already in the right form.
- 2 Remove terminals from the RHS of long rules. New grammar is: $X_a \rightarrow a$, $X_b \rightarrow b$, $S \rightarrow X_aA|X_bB|b$, $A \rightarrow BX_aX_a|X_bX_a$, and $B \rightarrow X_bAAX_b|X_aX_b$
- 3 Reduce the RHS of rules to be of length at most two.

Chomsky Normal Form

An Example

Example

Convert: $S \rightarrow aA|bB|b$, $A \rightarrow Baa|ba$, $B \rightarrow bAAb|ab$, into Chomsky Normal Form.

- 1 Eliminate ϵ -productions, unit productions, and useless symbols. This grammar is already in the right form.
- 2 Remove terminals from the RHS of long rules. New grammar is: $X_a \rightarrow a$, $X_b \rightarrow b$, $S \rightarrow X_aA|X_bB|b$, $A \rightarrow BX_aX_a|X_bX_a$, and $B \rightarrow X_bAAX_b|X_aX_b$
- 3 Reduce the RHS of rules to be of length at most two. New grammar replaces $A \rightarrow BX_aX_a$ by rules $A \rightarrow BX_{aa}$, $X_{aa} \rightarrow X_aX_a$, and $B \rightarrow X_bAAX_b$ by rules $B \rightarrow X_bX_{AAb}$, $X_{AAb} \rightarrow AX_{Ab}$, $X_{Ab} \rightarrow AX_b$