Properties of Context-Free Grammars

Theorem 2.1

Let $G = (V, \Sigma, R, S)$ be a CFG.

Theorem 2.1

Let $G = (V, \Sigma, R, S)$ be a CFG.

- Suppose that $w = w_1 w_2 \ldots w_k$, $k \geq 1$, where $w_i \in (V \cup \Sigma)^*$, $1 \leq i \leq k$ and $w \stackrel{*}{\Rightarrow} x$.

Theorem 2.1

Let $G = (V, \Sigma, R, S)$ be a CFG.

- Suppose that $w = w_1 w_2 \ldots w_k$, $k \geq 1$, where $w_i \in (V \cup \Sigma)^*$, $1 \leq i \leq k$ and $w \stackrel{*}{\Rightarrow} x$.
- Then there exist $x_i \in (V \cup \Sigma)^*$, $1 \leq i \leq k$, so that $x = x_1 x_2 \ldots x_k$ and $w_i \stackrel{*}{\Rightarrow} x_i$

Theorem 2.1

Let $G = (V, \Sigma, R, S)$ be a CFG.

- Suppose that $w = w_1 w_2 \ldots w_k$, $k \geq 1$, where $w_i \in (V \cup \Sigma)^*$, $1 \leq i \leq k$ and $w \overset{*}{\Rightarrow} x$.
- Then there exist $x_i \in (V \cup \Sigma)^*$, $1 \leq i \leq k$, so that $x = x_1 x_2 \ldots x_k$ and $w_i \overset{*}{\Rightarrow} x_i$

Proof idea: By induction on the length of the derivation of x

Proof

Induction base: derivation length zero. In this case $w = x$ and $w_i = x_i$, $1 \leq i \leq k$

Proof

Induction base: derivation length zero. In this case $w = x$ and $w_i = x_i$, $1 \leq i \leq k$

Induction step: assume the result for all derivations of $n \geq 0$ steps and consider the $n+1$ step derivation $w_1 w_2 \ldots w_k \stackrel{*}{\Rightarrow} x$.

In this case we have:

1. Suppose this derivation first rewrites w_m, for $1 \leq m \leq k$, i.e.,
$w_1 w_2 \ldots w_m \ldots w_k \Rightarrow w_1 w_2 \ldots w_{m-1} y_1 y_2 \ldots y_p w_{m+1} \ldots w_k \stackrel{*}{\Rightarrow} x$
where $w_m \to y_1 y_2 \ldots y_p$ is a specification rule.

In this case we have:

1. Suppose this derivation first rewrites w_m, for $1 \leq m \leq k$, i.e.,
   $w_1 w_2 \ldots w_m \ldots w_k \Rightarrow w_1 w_2 \ldots w_{m-1} y_1 y_2 \ldots y_p w_{m+1} \ldots w_k \stackrel{*}{\Rightarrow} x$
   where $w_m \rightarrow y_1 y_2 \ldots y_p$ is a specification rule.
2. Applying the induction hypothesis to the last n steps of this derivation, there must exist $x_1, x_2, \ldots, x_{m-1}, x_{m+1}, \ldots, x_k$, $z_1, z_2, \ldots, z_p$ so that $w_i \stackrel{*}{\Rightarrow} x_i$, $i = 1, 2, \ldots, m-1, m+1, \ldots, k$ and $y_j \stackrel{*}{\Rightarrow} z_i$, $1 \leq j \leq p$, and $x = x_1 x_2 \ldots x_{m-1} z_1 z_2 \ldots z_p x_{m+1} \ldots x_k$.

In this case we have:

1. Suppose this derivation first rewrites w_m, for $1 \leq m \leq k$, i.e., $w_1 w_2 \ldots w_m \ldots w_k \Rightarrow w_1 w_2 \ldots w_{m-1} y_1 y_2 \ldots y_p w_{m+1} \ldots w_k \stackrel{*}{\Rightarrow} x$ where $w_m \rightarrow y_1 y_2 \ldots y_p$ is a specification rule.
2. Applying the induction hypothesis to the last n steps of this derivation, there must exist $x_1, x_2, \ldots, x_{m-1}, x_{m+1}, \ldots, x_k$, $z_1, z_2, \ldots, z_p$ so that $w_i \stackrel{*}{\Rightarrow} x_i$, $i = 1, 2, \ldots, m-1, m+1, \ldots, k$ and $y_j \stackrel{*}{\Rightarrow} z_i$, $1 \leq j \leq p$, and $x = x_1 x_2 \ldots x_{m-1} z_1 z_2 \ldots z_p x_{m+1} \ldots x_k$.
3. Taking $x_m = z_1 z_2 \ldots z_p$ and $w_m \Rightarrow y_1 y_2 \ldots y_p \stackrel{*}{\Rightarrow} x_m$ the induction is extended to $n+1$ length derivation

Example application

Consider the CFG $G = (\{S, B\}, \{a, b\}, \{S \to aSB|\epsilon, B \to bB|\epsilon\}, S)$.

Example application

Consider the CFG $G = (\{S, B\}, \{a, b\}, \{S \rightarrow aSB|\epsilon, B \rightarrow bB|\epsilon\}, S)$.

- The following are derivations with G:

  $S \Rightarrow aSB \rightarrow aaSBB \Rightarrow aaSbBB,$

  $S \Rightarrow aSB \rightarrow aaSBB \Rightarrow aaSBbB,$

  $S \Rightarrow aSB \rightarrow aaSBB \Rightarrow aaSB,$

  $S \Rightarrow aSB \rightarrow aaSBB \Rightarrow aaBB$

  which show that derivations with this grammar are quite complex

Note

- According to Theorem 2.1, when rewriting the string $aaSBB$ we can consider further derivations of each of its symbols in isolation

Note

- According to Theorem 2.1, when rewriting the string $aaSBB$ we can consider further derivations of each of its symbols in isolation
- Derivations from B are $B \Rightarrow bB \Rightarrow bbB \Rightarrow b^kB$, $k \geq 0$

Note

- According to Theorem 2.1, when rewriting the string $aaSBB$ we can consider further derivations of each of its symbols in isolation
- Derivations from B are $B \Rightarrow bB \Rightarrow bbB \Rightarrow b^kB$, $k \geq 0$
- Therefore $aaSBB \stackrel{*}{\Rightarrow} aaSb^pb^q$, $p, q \geq 0$

Reachable symbols

Let $G = (V, \Sigma, R, S)$ be a CFG. A symbol:

Reachable symbols

Let $G = (V, \Sigma, R, S)$ be a CFG. A symbol:

1. $X \in (V \cup \Sigma)$ is reachable if $S \stackrel{*}{\Rightarrow} \alpha X \beta$ for some $\alpha, \beta \in (S \cup \Sigma)^*$;

Reachable symbols

Let $G = (V, \Sigma, R, S)$ be a CFG. A symbol:

1. $X \in (V \cup \Sigma)$ is reachable if $S \stackrel{*}{\Rightarrow} \alpha X \beta$ for some $\alpha, \beta \in (S \cup \Sigma)^*$;
2. $X \in (V \cup \Sigma)$ is unreachable otherwise

Reachable symbols

Let $G = (V, \Sigma, R, S)$ be a CFG. A symbol:

1. $X \in (V \cup \Sigma)$ is reachable if $S \stackrel{*}{\Rightarrow} \alpha X \beta$ for some $\alpha, \beta \in (S \cup \Sigma)^*$;
2. $X \in (V \cup \Sigma)$ is unreachable otherwise
3. $X \in V$ is live if $X \stackrel{*}{\Rightarrow} x$ for some $x \in \Sigma^*$

Reachable symbols

Let $G = (V, \Sigma, R, S)$ be a CFG. A symbol:

1. $X \in (V \cup \Sigma)$ is reachable if $S \overset{*}{\Rightarrow} \alpha X \beta$ for some $\alpha, \beta \in (S \cup \Sigma)^*$;
2. $X \in (V \cup \Sigma)$ is unreachable otherwise
3. $X \in V$ is live if $X \overset{*}{\Rightarrow} x$ for some $x \in \Sigma^*$
4. $X \in V$ is dead if there is no $x \in \Sigma^*$ such that $X \overset{*}{\Rightarrow} x$

Reachable symbols

Let $G = (V, \Sigma, R, S)$ be a CFG. A symbol:

1. $X \in (V \cup \Sigma)$ is reachable if $S \stackrel{*}{\Rightarrow} \alpha X \beta$ for some $\alpha, \beta \in (S \cup \Sigma)^*$;
2. $X \in (V \cup \Sigma)$ is unreachable otherwise
3. $X \in V$ is live if $X \stackrel{*}{\Rightarrow} x$ for some $x \in \Sigma^*$
4. $X \in V$ is dead if there is no $x \in \Sigma^*$ such that $X \stackrel{*}{\Rightarrow} x$
5. $X \in (X \cup \Sigma)$ is useless if it is either unreachable or dead

Note

- Unreachable symbols and their productions can contribute nothing to the language specifi ed by a CFG

Note

- Unreachable symbols and their productions can contribute nothing to the language specifi ed by a CFG
- The occurrence of a dead variable in a derivation insures that that derivation contribute no string in the language, even if that variable is reachable

Note

- Unreachable symbols and their productions can contribute nothing to the language specifi ed by a CFG
- The occurrence of a dead variable in a derivation insures that that derivation contribute no string in the language, even if that variable is reachable
- While useless symbols are no value to the description of a language, they are not prohibited.

Example

Consider the CFG $G = (\{S, A, B, C\}, \{a, b\}, R, S)$ where R is the set

$$\begin{aligned} S &\rightarrow bb \mid aB \\ A &\rightarrow a \mid Aa \\ B &\rightarrow bB \mid Ba \mid AB \\ C &\rightarrow ba \mid aA \mid Bb \mid aCb \end{aligned}$$

Example

Consider the CFG $G = (\{S, A, B, C\}, \{a, b\}, R, S)$ where R is the set

$$\begin{array}{rcl} S & \rightarrow & bb \mid aB \\ A & \rightarrow & a \mid Aa \\ B & \rightarrow & bB \mid Ba \mid AB \\ C & \rightarrow & ba \mid aA \mid Bb \mid aCb \end{array}$$

1. C is unreachable and B is dead

Example

Consider the CFG $G = (\{S, A, B, C\}, \{a, b\}, R, S)$ where R is the set

$$\begin{array}{lcl} S & \rightarrow & bb \mid aB \\ A & \rightarrow & a \mid Aa \\ B & \rightarrow & bB \mid Ba \mid AB \\ C & \rightarrow & ba \mid aA \mid Bb \mid aCb \end{array}$$

1. C is unreachable and B is dead
2. A is live and reachable but contributes nothing to the language

Cleaning up a grammar

Cleaning up a grammar is the process of eliminating useless symbols and their productions

Dead symbol elimination

For each CFG $G = (V, \Sigma, R, S)$ with $L(G) \neq \emptyset$, there is a CFG $G' = (V', \Sigma, R', S)$ so that $L(G') = L(G)$, $V' \subseteq V$, $R' \subseteq R$, and G' has no dead symbols.

Dead symbol elimination

For each CFG $G = (V, \Sigma, R, S)$ with $L(G) \neq \emptyset$, there is a CFG $G' = (V', \Sigma, R', S)$ so that $L(G') = L(G)$, $V' \subseteq V$, $R' \subseteq R$, and G' has no dead symbols.

Proof idea: partition inductively the variables of G as follows:

Dead symbol elimination

For each CFG $G = (V, \Sigma, R, S)$ with $L(G) \neq \emptyset$, there is a CFG $G' = (V', \Sigma, R', S)$ so that $L(G') = L(G)$, $V' \subseteq V$, $R' \subseteq R$, and G' has no dead symbols.

Proof idea: partition inductively the variables of G as follows:

1. $V_0 = \{X \in V \mid X \to w \in R \wedge w \in \Sigma^*\}$

Dead symbol elimination

For each CFG $G = (V, \Sigma, R, S)$ with $L(G) \neq \emptyset$, there is a CFG $G' = (V', \Sigma, R', S)$ so that $L(G') = L(G)$, $V' \subseteq V$, $R' \subseteq R$, and G' has no dead symbols.

Proof idea: partition inductively the variables of G as follows:

1. $V_0 = \{X \in V \mid X \to w \in R \wedge w \in \Sigma^*\}$
2. For $i \geq 0$ define $V_{i+1} = V_i \cup \{X \in V \mid X \to w \in R \wedge w \in (V_i \cup \Sigma)^*\}$

Dead symbol elimination

For each CFG $G = (V, \Sigma, R, S)$ with $L(G) \neq \emptyset$, there is a CFG $G' = (V', \Sigma, R', S)$ so that $L(G') = L(G)$, $V' \subseteq V$, $R' \subseteq R$, and G' has no dead symbols.

Proof idea: partition inductively the variables of G as follows:

1. $V_0 = \{X \in V \mid X \rightarrow w \in R \wedge w \in \Sigma^*\}$
2. For $i \geq 0$ define $V_{i+1} = V_i \cup \{X \in V \mid X \rightarrow w \in R \wedge w \in (V_i \cup \Sigma)^*\}$
3. Define $V' = \cup_{i=0}^{\infty} V_i$ and $R' = \{X \rightarrow w \in R \mid \wedge w \in (V' \cup \Sigma)^*\}$

Claim

We show now that V' is the set of all live symbols of G and for each $X \in V$, $X \overset{R}{\Rightarrow} y \in \Sigma^*$ iff $X \overset{R'}{\Rightarrow} y \in \Sigma^*$

Claim

We show now that V' is the set of all live symbols of G and for each $X \in V$, $X \overset{R}{\Rightarrow} y \in \Sigma^*$ iff $X \overset{R'}{\Rightarrow} y \in \Sigma^*$

Every variable in V' is live

Claim

We show now that V' is the set of all live symbols of G and for each $X \in V$, $X \overset{R}{\Rightarrow} y \in \Sigma^*$ iff $X \overset{R'}{\Rightarrow} y \in \Sigma^*$

Every variable in V' is live

1. Each $X \in V'$ must belong to V_i for some i

Claim

We show now that V' is the set of all live symbols of G and for each $X \in V$, $X \overset{R}{\Rightarrow} y \in \Sigma^*$ iff $X \overset{R'}{\Rightarrow} y \in \Sigma^*$

Every variable in V' is live

1. Each $X \in V'$ must belong to V_i for some i
2. By definition, variables in V_0 are live

Claim

We show now that V' is the set of all live symbols of G and for each $X \in V$, $X \stackrel{R}{\Rightarrow} y \in \Sigma^*$ iff $X \stackrel{R'}{\Rightarrow} y \in \Sigma^*$

Every variable in V' is live

1. Each $X \in V'$ must belong to V_i for some i
2. By definition, variables in V_0 are live
3. If variables in V_i are live then variables in V_{i+1} are live by construction

Proof, continuation

By ind. on the length of a shortest derivation from $X \in V$ we show that every live variable X of G belongs to V'

Proof, continuation

By ind. on the length of a shortest derivation from $X \in V$ we show that every live variable X of G belongs to V'

1. If X has a one step derivation then by construction $X \in V_0$ and hence $X \in V'$

Proof, continuation

By ind. on the length of a shortest derivation from $X \in V$ we show that every live variable X of G belongs to V'

1. If X has a one step derivation then by construction $X \in V_0$ and hence $X \in V'$
2. Assume that all variables with n or fewer step derivations to a terminal string belong to V' and consider a variable X with $n+1$ step derivation, say $X \Rightarrow w_1 w_2 \ldots w_k \overset{*}{\Rightarrow} y \in \Sigma^*$.

Proof, continuation

By ind. on the length of a shortest derivation from $X \in V$ we show that every live variable X of G belongs to V'

1. If X has a one step derivation then by construction $X \in V_0$ and hence $X \in V'$
2. Assume that all variables with n or fewer step derivations to a terminal string belong to V' and consider a variable X with $n+1$ step derivation, say $X \Rightarrow w_1 w_2 \dots w_k \overset{*}{\Rightarrow} y \in \Sigma^*$.
3. By Theorem 2.1, $y = y_1 y_2 \dots y_k$ and $w_i \overset{*}{\Rightarrow} y_i$, $1 \leq i \leq k$. Hence, by induction hypothesis $w_i \in V'$. But then each w_i, $1 \leq i \leq k$ belongs to some set $V_m \subseteq V'$.

Note

If p is the maximum index m in the above conclusion then by construction $\{w_1, w_2, \ldots, w_k\} \subseteq V_p \subseteq V'$. Thus the derivation $X \Rightarrow w_1 w_2 \ldots w_k$ sets X in V_{p+1} and therefore $X \in V'$.

Proof, continuation

$L(G') = L(G)$

1. Since $L(G) \neq \emptyset$ it result that S is live and hence $S \in V'$

Proof, continuation

$L(G') = L(G)$

1. Since $L(G) \neq \emptyset$ it result that S is live and hence $S \in V'$
2. $L(G') \subseteq L(G)$ because $R' \subseteq R$ and so any rewriting in G' can also be done in G

Proof, continuation

$L(G') = L(G)$

1. Since $L(G) \neq \emptyset$ it result that S is live and hence $S \in V'$
2. $L(G') \subseteq L(G)$ because $R' \subseteq R$ and so any rewriting in G' can also be done in G
3. $L(G) \subseteq L(G')$: each step in the derivation of a string of terminals in G can introduce no dead symbols and so productions used belong to R'.

Unreachable symbol elimination

For each CFG $G = (V, \Sigma, R, S)$ with $L(G) \neq \emptyset$ there is $G' = (V', \Sigma', R', S)$ so that $L(G') = L(G)$, $V' \subseteq V$, $\Sigma' \subseteq \Sigma$, $R' \subseteq R$, and G' has no dead or unreachable symbols.

Proof idea

- By the previous result, assume that G has no dead variables.

Proof idea

- By the previous result, assume that G has no dead variables.
- Split the symbols in $V \cup \Sigma$ inductively by the procedure:

Proof idea

- By the previous result, assume that G has no dead variables.
- Split the symbols in $V \cup \Sigma$ inductively by the procedure:
  1. $S_0 = \{S\}$

Proof idea

- By the previous result, assume that G has no dead variables.
- Split the symbols in $V \cup \Sigma$ inductively by the procedure:
  1. $S_0 = \{S\}$
  2. For $i \geq 0$, $S_{i+1} = S_i \cup \{\sigma \in (V \cup \Sigma) \mid \exists X \in S_i \wedge \exists \alpha, \beta \in (V \cup \Sigma)^* \wedge X \to \alpha\sigma\beta \in R\}$

Proof idea

- By the previous result, assume that G has no dead variables.
- Split the symbols in $V \cup \Sigma$ inductively by the procedure:
  1. $S_0 = \{S\}$
  2. For $i \geq 0$, $S_{i+1} = S_i \cup \{\sigma \in (V \cup \Sigma) \mid \exists X \in S_i \wedge \exists \alpha, \beta \in (V \cup \Sigma)^* \wedge X \to \alpha\sigma\beta \in R\}$
  3. Set $V' = V \cap (\cup_{i=0}^{\infty} S_i)$, $\Sigma' = \Sigma \cap (\cup_{i=0}^{\infty} S_i)$, $R' = \{r \in R \mid lhs(r) \in V' \wedge rhs(r) \in (V' \cup \Sigma')^*\}$

Proof, continuation

Claim 1: $V' \cup \Sigma'$ is exactly the collection of reachable symbols of G

Proof: show that each $\sigma \in V' \cup \Sigma'$ is reachable by induction on the smallest index i so that $\sigma \in S_i$.

Induction basis: For $i = 0$ this is trivial

Induction step: assume that all symbols in S_n are reachable and $\sigma \in S_{n+1}$. If $\sigma \in S_n$ then it is reachable by induction hypothesis; if $\sigma \notin S_n$ then there exist $X \in S_n$, $\alpha, \beta \in (V \cup \Sigma)^*$ with $X \to \alpha\sigma\beta \in R$. Since $X \in S_n$, X is reachable and so $S \stackrel{*}{\Rightarrow} w_1 X w_2 \Rightarrow w_1 \alpha\sigma\beta w_2$ and thus σ is reachable

Note

The claim that every reachable symbol σ of G is in $V' \cup \Sigma'$ is proved by induction on the length of the shortest derivation producing σ.

Proof, continuation

Claim 2: deleting the unreachable symbols and their rules create no dead variables in G'

Proof: if X is a reachable variable and $X \stackrel{*}{\Rightarrow} w \in \Sigma^*$ in G, then every symbol introduced in this derivation is also reachable, and so it is not dead. Consequently X is still live in G'

Proof, continuation

Claim 3: $L(G') = L(G)$

Proof: $L(G') \subseteq L(G) \wedge L(G) \subseteq L(G')$

1. $L(G') \subseteq L(G)$: since $R' \subseteq R$, each derivation in G' is a derivation in G.
2. $L(G) \subseteq L(G')$: If $S \stackrel{*}{\Rightarrow} w \in \Sigma^*$ in G then every symbol introduced in this derivation is reachable. Consequently all productions used are retained in G' so this is also a derivation in G'

Application

Clean $G = (\{S, A, B, C\}, \{a, b\}, R, S)$ where R is:

$$\begin{array}{lcl} S & \rightarrow & bb|aB \\ A & \rightarrow & a|Aa \\ B & \rightarrow & bB|Ba|AB \\ C & \rightarrow & ba|aA|Bb|aCb \end{array}$$

1. $V_0 = \{S, A, C\} = V_1 = V'$;
   $G' = (\{S, A, C\}, \{a, b\}, \{S \rightarrow bb, A \rightarrow a|Aa, C \rightarrow ba|aA|aCb\}, S)$
2. $S_0 = \{S\}$, $S_1 = \{S, b\}$, $S_2 = S_1$;$G'' = (\{S\}, \{b\}, \{S \rightarrow bb\}, S)$
3. $L(G'') = L(G) = \{bb\}$

Language elements

Language element interpretation:

- Sometimes a CFG can generate the same string in several different ways
- Such a string will have several different derivation trees
- Since each derivation tree represents an interpretation of the string, each derivation tree defi nes a meaning of the string.
- Different derivation trees for a string means different meanings for the same language element

Observations

1. Multiple meanings of the same language element are undesirable for some applications
2. For example, multiple meanings of a program are unacceptable in a programming language
3. Each language element in a programming language should have a unique interpretation

Note: multiple derivations for a sentence is a common situation in natural languages

Ambiguity

- If a CFG G generates the same string x in several different ways, we say that x is derived *ambiguously* in G.

Ambiguity

- If a CFG G generates the same string x in several different ways, we say that x is derived *ambiguously* in G.
- If a CFG G generates some string ambiguously we say that the grammar G is *ambiguous*

Example

Consider the grammar G_4 whose rules are:

$E \rightarrow E + T|T, T \rightarrow T * F|F, F \rightarrow (E)|a$

and the grammar G_5, whose rules are:

$E \longrightarrow E + E|E * E|(E)|a$

- $L(G_4) = L(G_5)$

  Note: one can easily show this by showing the inclusions $L(G_4) \subseteq L(G_5)$ and $L(G_5 \subseteq L(G_4)$
- G_5 generates ambiguously some arithmetic expressions while G_4 doesn't.

Example

Consider the grammar G_4 whose rules are:

$E \rightarrow E+T|T, T \rightarrow T*F|F, F \rightarrow (E)|a$

and the grammar G_5, whose rules are:

$E \longrightarrow E+E|E*E|(E)|a$

- $L(G_4) = L(G_5)$

  Note: one can easily show this by showing the inclusions $L(G_4) \subseteq L(G_5)$ and $L(G_5 \subseteq L(G_4)$

- G_5 generates ambiguously some arithmetic expressions while G_4 doesn't.

Example

Consider the grammar G_4 whose rules are:

$E \to E + T|T, T \to T * F|F, F \to (E)|a$

and the grammar G_5, whose rules are:

$E \longrightarrow E + E|E * E|(E)|a$

- $L(G_4) = L(G_5)$

  Note: one can easily show this by showing the inclusions $L(G_4) \subseteq L(G_5)$ and $L(G_5 \subseteq L(G_4)$

- G_5 generates ambiguously some arithmetic expressions while G_4 doesn't.

Ambiguous expressions

Figure 1 shows two different derivation trees for `a+a*a`

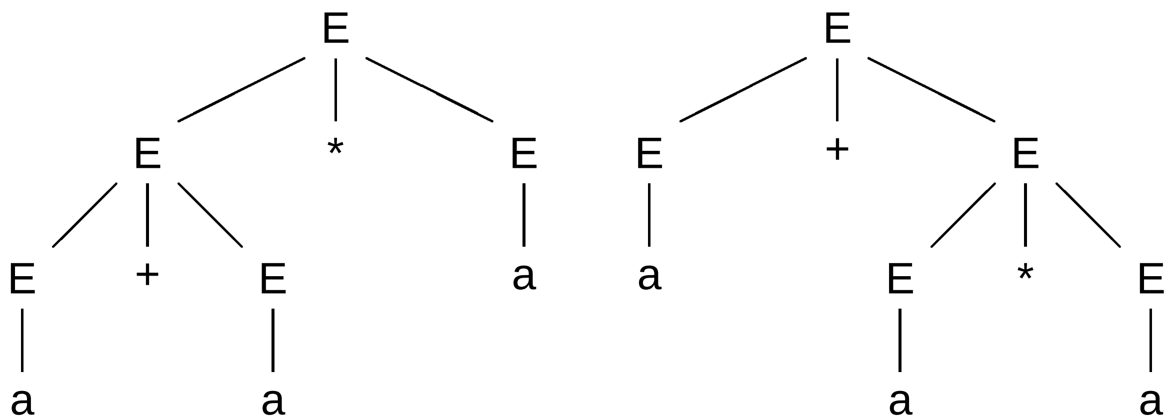

Figure 1: Two derivation trees for `a+a*a`

Note

- The grammar G_5 does not capture the usual precedence relations and so groups '+' before '*' and vice versa

Note

- The grammar G_5 does not capture the usual precedence relations and so groups '+' before '*' and vice versa
- In contrast, the grammar G_4 generates the same language, but every generated string has a unique derivation tree

Note

- The grammar G_5 does not capture the usual precedence relations and so groups '+' before '*' and vice versa
- In contrast, the grammar G_4 generates the same language, but every generated string has a unique derivation tree
- Hence, G_5 is ambiguous and G_4 is not, i.e., G_4 is *unambiguous*

Another example

G_2 below is another ambiguous grammar

$$
\begin{array}{rcl}
\langle SENTENCE \rangle & \longrightarrow & \langle NounPhrase \rangle\ \langle VerbPhrase \rangle \\
\langle NounPhrase \rangle & \longrightarrow & \langle CpNoun \rangle | \langle CpNoun \rangle\ \langle PrepPhrase \rangle \\
\langle VerbPhrase \rangle & \longrightarrow & \langle CpVerb \rangle | \langle CpVerb \rangle\ \langle PrepPhrase \rangle \\
\langle PrepPhrase \rangle & \longrightarrow & \langle Prep \rangle\ \langle CpNoun \rangle \\
\langle CpNoun \rangle & \longrightarrow & \langle Article \rangle\ \langle Noun \rangle \\
\langle CpVerb \rangle & \longrightarrow & \langle Verb \rangle | \langle Verb \rangle\ \langle NounPhrase \rangle \\
\langle Article \rangle & \longrightarrow & a | the \\
\langle Noun \rangle & \longrightarrow & boy | girl | flower \\
\langle Verb \rangle & \longrightarrow & touches | likes | sees \\
\langle Prep \rangle & \longrightarrow & with
\end{array}
$$

Example ambiguous string

The sentence:

`the girl touches the boy with the flower`

has two different derivations, so it is ambiguous

The two derivations correspond to the two readings:

(the girl touches the boy) (with the flower)

(the girl touches) (the boy with the flower)

Example ambiguous string

The sentence:

`the girl touches the boy with the flower`

has two different derivations, so it is ambiguous

The two derivations correspond to the two readings:

(the girl touches the boy) (with the flower)

(the girl touches) (the boy with the flower)

Example ambiguous string

The sentence:

`the girl touches the boy with the flower`

has two different derivations, so it is ambiguous

The two derivations correspond to the two readings:

(the girl touches the boy) (with the flower)

(the girl touches) (the boy with the flower)

Note

- When a grammar generates a string ambiguously it means that the string has two different derivation trees.

Note

- When a grammar generates a string ambiguously it means that the string has two different derivation trees.
- Two different derivations however, may produce the same derivation tree because they may differ in the order in which they replace nonterminals not in the rules they use

Note

- When a grammar generates a string ambiguously it means that the string has two different derivation trees.
- Two different derivations however, may produce the same derivation tree because they may differ in the order in which they replace nonterminals not in the rules they use
- To concentrate on the structure of derivations we need to fi x the order of rule application

Fixing rule application order

Leftmost derivation: a derivation of a string w in a grammar G is a *leftmost derivation* if at every step the leftmost nonterminal is replaced

Fixing rule application order

Leftmost derivation: a derivation of a string w in a grammar G is a *leftmost derivation* if at every step the leftmost nonterminal is replaced

Rightmost derivation: a derivation of a string w in a grammar G is a *rightmost derivation* if at every step the rightmost nonterminal is replaced

Note

The leftmost and rightmost derivations of a string w are unique, so they are equivalent to the derivation trees

Ambiguity again

- A string w is derived ambiguously in the CFG G if it has two or more different leftmost (or rightmost) derivations.

Ambiguity again

- A string w is derived ambiguously in the CFG G if it has two or more different leftmost (or rightmost) derivations.
- A CFG G is ambiguous if it generates some string ambiguously

Ambiguity again

- A string w is derived ambiguously in the CFG G if it has two or more different leftmost (or rightmost) derivations.
- A CFG G is ambiguous if it generates some string ambiguously

Note

Sometimes when we have an ambiguous grammar (such as G_5) we can find an unambiguous grammar (such as G_4) that generates the same language

Inherent ambiguity

- Some CFL, however, can be generated only by ambiguous grammar.

Inherent ambiguity

- Some CFL, however, can be generated only by ambiguous grammar.
- A CFL that can be generated only by ambiguous grammars is called *inherently ambiguous*

Inherent ambiguity

- Some CFL, however, can be generated only by ambiguous grammar.
- A CFL that can be generated only by ambiguous grammars is called *inherently ambiguous*
- **Example of inherently ambiguous language:**

$$\{0^i 1^j 2^k | i = j \vee j = k\}$$