Számításelmélet

2. előadás

előadó: Kolonits Gábor
kolomax@inf.elte.hu

Hossz-nemcsökkentő grammatika

Definíció

Egy $G = \langle N, T, P, S \rangle$ grammatikát **hossz-nemcsökkentőnek** mondunk, ha minden szabályának alakjára az alábbiak valamelyike teljesül

Hossz-nemcsökkentő grammatika

Definíció

Egy $G = \langle N, T, P, S \rangle$ grammatikát **hossz-nemcsökkentőnek** mondunk, ha minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.

Hossz-nemcsökkentő grammatika

Definíció

Egy $G = \langle N, T, P, S\rangle$ grammatikát **hossz-nemcsökkentőnek** mondunk, ha minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $u \to v$, ahol $u, v \in (T \cup N)^+$ és $|u| \leqslant |v|$.

Hossz-nemcsökkentő grammatika

Definíció

Egy $G = \langle N, T, P, S \rangle$ grammatikát **hossz-nemcsökkentőnek** mondunk, ha minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $u \to v$, ahol $u, v \in (T \cup N)^+$ és $|u| \leqslant |v|$.

A környezetfüggő grammatikák nyilván hossz-nemcsökkentőek.

Hossz-nemcsökkentő grammatika

Tétel

Minden hossz-nemcsökkentő grammatika környezetfüggő nyelvet generál.

Hossz-nemcsökkentő grammatika

Tétel

Minden hossz-nemcsökkentő grammatika környezetfüggő nyelvet generál.

Bizonyítás: (vázlat) Minden hossz-nemcsökkentő $G = \langle N, T, P, S\rangle$ grammatikához megadható egy vele ekvivalens $G' = \langle N', T, P', S\rangle$ környezetfüggő grammatika.

Hossz-nemcsökkentő grammatika

Tétel

Minden hossz-nemcsökkentő grammatika környezetfüggő nyelvet generál.

Bizonyítás: (vázlat) Minden hossz-nemcsökkentő $G = \langle N, T, P, S \rangle$ grammatikához megadható egy vele ekvivalens $G' = \langle N', T, P', S \rangle$ környezetfüggő grammatika.

1. lépés: **Álterminálisok bevezetése**
A szokásos módon feltehető, terminálisok csak $A \rightarrow a$ $(A \in N, a \in T)$ alakú szabályok jobboldalán fordulnak elő.

Hossz-nemcsökkentő grammatika

2. lépés: Környezetfüggő szabályokkal való helyettesítés
Legyen $X_1 X_2 \cdots X_n \to Y_1 Y_2 \cdots Y_m \quad (m \geqslant n)$ egy hossz-nemcsökkető szabály.

Hossz-nemcsökkentő grammatika

2. lépés: Környezetfüggő szabályokkal való helyettesítés

Legyen $X_1X_2 \cdots X_n \to Y_1Y_2 \cdots Y_m \quad (m \geqslant n)$ egy hossz-nemcsökkető szabály. Ezt az alábbi csupa 1-típusú szabályokkal szimulálhatjuk:

$$
\begin{gathered}
X_1X_2 \cdots X_n \to Z_1X_2 \cdots X_n, \\
Z_1X_2 \cdots X_n \to Z_1Z_2X_3 \cdots X_n, \\
\vdots \\
Z_1Z_2 \cdots Z_{n-1}X_n \to Z_1Z_2 \cdots Z_nY_{n+1} \cdots Y_m \quad (n \leqslant m), \\
Z_1Z_2 \cdots Z_nY_{n+1} \cdots Y_m \to Y_1Z_2 \ldots Z_nY_{n+1} \cdots Y_m, \\
\vdots \\
Y_1 \cdots Y_{n-1}Z_nY_{n+1} \cdots Y_m \to Y_1Y_2 \cdots Y_m,
\end{gathered}
$$

ahol $Z_1, Z_2, \ldots, Z_n$ új nemterminálisok.

Hossz-nemcsökkentő grammatika

Meggondolható, hogy a $Z_1, \ldots, Z_n$ új volta miatt a szabályokat csak ebben a sorrendben lehet és kell végrehajtani, ezért az új grammatika is ugyanazt a nyelvet generálja. Csináljuk meg ezt a szabálytranszformációt az összes „rossz" szabályra. Az így kapott G' grammatika már 1-típusú és $L(G)$-t generálja.

Hossz-nemcsökkentő grammatika

Meggondolható, hogy a $Z_1, \ldots, Z_n$ új volta miatt a szabályokat csak ebben a sorrendben lehet és kell végrehajtani, ezért az új grammatika is ugyanazt a nyelvet generálja. Csináljuk meg ezt a szabálytranszformációt az összes „rossz” szabályra. Az így kapott G' grammatika már 1-típusú és $L(G)$-t generálja.

1. Példa (csak egyetlen hosszú szabály):

Az $ABC \rightarrow DEFGH$ szabály a következő környezetfüggő szabályokkal helyettesíthető:

Hossz-nemcsökkentő grammatika

Meggondolható, hogy a $Z_1, \ldots, Z_n$ új volta miatt a szabályokat csak ebben a sorrendben lehet és kell végrehajtani, ezért az új grammatika is ugyanazt a nyelvet generálja. Csináljuk meg ezt a szabálytranszformációt az összes „rossz'' szabályra. Az így kapott G' grammatika már 1-típusú és $L(G)$-t generálja.

1. Példa (csak egyetlen hosszú szabály):
Az $ABC \to DEFGH$ szabály a következő környezetfüggő szabályokkal helyettesíthető:

$ABC \to XBC$

$XBC \to XYC$

$XYC \to XYZGH$

$XYZGH \to DYZGH$

$DYZGH \to DEZGH$

$DEZGH \to DEFGH$

(X, Y, Z új nemterminálisok)

Hossz-nemcsökkentő grammatika

2. **Példa:** $G = \langle \{S, B\}, \{a, b, c\}, P, S \rangle$

Hossz-nemcsökkentő grammatika

2. Példa: $G = \langle \{S, B\}, \{a, b, c\}, P, S \rangle$

$P = \{S \to abc, S \to aSBc, cB \to Bc, bB \to abb\}$

Egy a G-vel ekvivalens 1-es típusú grammatika szabályai:

$S \to DEF$
$S \to DSBF$
$FB \to Z_1 B$
$Z_1 B \to Z_1 F$
$Z_1 F \to BF$

$EB \to Z_2 B$
$Z_2 B \to Z_2 EE$
$Z_2 EE \to DEE$
$D \to a$
$E \to b$
$F \to c$

(D, E, F, Z_1, Z_2 új nemterminálisok)

Kuroda normálforma

Definíció

Egy $G = \langle N, T, P, S \rangle$ grammatikát **Kuroda normálformájúnak** mondunk, ha minden szabályának alakjára az alábbiak valamelyike teljesül

Kuroda normálforma

Definíció

Egy $G = \langle N, T, P, S \rangle$ grammatikát **Kuroda normálformájúnak** mondunk, ha minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.

Kuroda normálforma

Definíció

Egy $G = \langle N, T, P, S\rangle$ grammatikát **Kuroda normálformájúnak** mondunk, ha minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $A \to a$, ahol $A \in N, a \in T$,

Kuroda normálforma

Definíció

Egy $G = \langle N, T, P, S\rangle$ grammatikát **Kuroda normálformájúnak** mondunk, ha minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $A \to a$, ahol $A \in N, a \in T$,
- $A \to BC$, ahol $A, B, C \in N$,

Kuroda normálforma

Definíció

Egy $G = \langle N, T, P, S\rangle$ grammatikát **Kuroda normálformájúnak** mondunk, ha minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $A \to a$, ahol $A \in N, a \in T$,
- $A \to BC$, ahol $A, B, C \in N$,
- $AB \to AC$, ahol $A, B, C \in N$,

Kuroda normálforma

Definíció

Egy $G = \langle N, T, P, S \rangle$ grammatikát **Kuroda normálformájúnak** mondunk, ha minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \rightarrow \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $A \rightarrow a$, ahol $A \in N, a \in T$,
- $A \rightarrow BC$, ahol $A, B, C \in N$,
- $AB \rightarrow AC$, ahol $A, B, C \in N$,
- $BA \rightarrow CA$, ahol $A, B, C \in N$.

Kuroda normálforma

Definíció

Egy $G = \langle N, T, P, S \rangle$ grammatikát **Kuroda normálformájúnak** mondunk, ha minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $A \to a$, ahol $A \in N, a \in T$,
- $A \to BC$, ahol $A, B, C \in N$,
- $AB \to AC$, ahol $A, B, C \in N$,
- $BA \to CA$, ahol $A, B, C \in N$.

A Kuroda normálformájú grammatikák nyilván környezetfüggőek.

Kuroda normálforma

Definíció

Egy $G = \langle N, T, P, S \rangle$ grammatikát **Kuroda normálformájúnak** mondunk, ha minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $A \to a$, ahol $A \in N, a \in T$,
- $A \to BC$, ahol $A, B, C \in N$,
- $AB \to AC$, ahol $A, B, C \in N$,
- $BA \to CA$, ahol $A, B, C \in N$.

A Kuroda normálformájú grammatikák nyilván környezetfüggőek.

Tétel

Minden környezetfüggő grammatika G grammatikához van vele ekvivalens Kuroda normálformájú G' grammatika.

Kuroda normálforma

Bizonyítás: (vázlat)

1. lépés: Álterminálisok bevezetése

A szokásos módon feltehető, terminálisok csak
$A \to a$ $(A \in N, a \in T)$ alakú szabályok jobboldalán fordulnak elő.

Kuroda normálforma

Bizonyítás: (vázlat)

1. lépés: Álterminálisok bevezetése

A szokásos módon feltehető, terminálisok csak
$A \to a$ $(A \in N, a \in T)$ alakú szabályok jobboldalán fordulnak elő.

2. lépés: Környezetfüggetlen szabályok hosszredukciója

Szintén a Chomsky normálformánál látott módon.

Kuroda normálforma

Bizonyítás: (vázlat)

1. lépés: Álterminálisok bevezetése

A szokásos módon feltehető, terminálisok csak
$A \to a$ $(A \in N, a \in T)$ alakú szabályok jobboldalán fordulnak elő.

2. lépés: Környezetfüggetlen szabályok hosszredukciója

Szintén a Chomsky normálformánál látott módon.

3. lépés: Környezetfüggő láncmentesítés

Az A-ból láncszabályokkal elérhető nemterminálisok
$H(A) = \{B \in N \mid A \Rightarrow^* B\}$ halmazának meghatározása a Chomsky normálformánál látott módon.

Kuroda normálforma

Bizonyítás: (vázlat)

1. lépés: Álterminálisok bevezetése

A szokásos módon feltehető, terminálisok csak
$A \to a$ $(A \in N, a \in T)$ alakú szabályok jobboldalán fordulnak elő.

2. lépés: Környezetfüggetlen szabályok hosszredukciója

Szintén a Chomsky normálformánál látott módon.

3. lépés: Környezetfüggő láncmentesítés

Az A-ból láncszabályokkal elérhető nemterminálisok
$H(A) = \{B \in N \mid A \Rightarrow^* B\}$ halmazának meghatározása a Chomsky normálformánál látott módon. A szabályrendszer módosítása:

$$P' := \{A_1 \cdots A_n \to w \mid w \notin N \wedge \exists B_1 \cdots B_n \to w \in P : B_i \in H(A_i)\,(\forall 1 \leqslant i \leqslant n)\}.$$

Kuroda normálforma

Bizonyítás: (vázlat)

1. lépés: Álterminálisok bevezetése

A szokásos módon feltehető, terminálisok csak $A \to a$ $(A \in N, a \in T)$ alakú szabályok jobboldalán fordulnak elő.

2. lépés: Környezetfüggetlen szabályok hosszredukciója

Szintén a Chomsky normálformánál látott módon.

3. lépés: Környezetfüggő láncmentesítés

Az A-ból láncszabályokkal elérhető nemterminálisok $H(A) = \{B \in N \mid A \Rightarrow^* B\}$ halmazának meghatározása a Chomsky normálformánál látott módon. A szabályrendszer módosítása:

$$P' := \{A_1 \cdots A_n \to w \mid w \notin N \wedge \exists B_1 \cdots B_n \to w \in P : B_i \in H(A_i)\ (\forall 1 \leqslant i \leqslant n)\}.$$

4. lépés: Környezetfüggő szabályok hosszredukciója

Az $X_1 \cdots X_m \to Y_1 \cdots Y_n$ alakú szabályok szimulációja, ahol $n \geqslant m \geqslant 2$.

Kuroda normálforma

Ha $n = m = 2$, akkor a következő lépésre ugorhatunk.

Kuroda normálforma

Ha $n = m = 2$, akkor a következő lépésre ugorhatunk. Különben a szabály szimulációja a $Z_1, Z_2, \ldots, Z_{n-2}$ új nemterminálisok bevezetésével:

$$\begin{gathered} X_1X_2 \to Y_1Z_1, \\ Z_1X_3 \to Y_2Z_2, \\ \vdots \\ Z_{m-3}X_{m-1} \to Y_{m-2}Z_{m-2}, \end{gathered}$$

Továbbá ha $n = m$, akkor

$$Z_{m-2}X_m \to Y_{m-1}Y_m,$$

Kuroda normálforma

Ha $n = m = 2$, akkor a következő lépésre ugorhatunk. Különben a szabály szimulációja a $Z_1, Z_2, \dots, Z_{n-2}$ új nemterminálisok bevezetésével:

$$\begin{gathered} X_1X_2 \to Y_1Z_1, \\ Z_1X_3 \to Y_2Z_2, \\ \vdots \\ Z_{m-3}X_{m-1} \to Y_{m-2}Z_{m-2}, \end{gathered}$$

Továbbá ha $n = m$, akkor

$$Z_{m-2}X_m \to Y_{m-1}Y_m,$$

egyébként $(n > m)$ esetén:

$$\begin{gathered} Z_{m-2}X_m \to Y_{m-1}Z_{m-1}, \\ Z_{m-1} \to Y_mZ_m, \\ \vdots \\ Z_{n-3} \to Y_{n-2}Z_{n-2}, \\ Z_{n-2} \to Y_{n-1}Y_n. \end{gathered}$$

Kuroda normálforma

5. lépés: Az $AB \to CD,\ A \neq C, B \neq D$ szabályok eliminációja

Kuroda normálforma

5. lépés: Az $AB \to CD$, $A \neq C, B \neq D$ szabályok eliminációja

Végül a nem Kuroda-normálformájú szabályok sémája ekkor $AB \to CD$ $(A, B, C, D \in N)$. Átalakításukhoz szabályonként egyedi W új nemterminálisokat vezetünk be és a fenti szabályt az alábbi szabályokkal szimuláljuk:

$$AB \to AW,$$
$$AW \to CW,$$
$$CW \to CD.$$

Kuroda normálforma

5. lépés: Az $AB \to CD$, $A \neq C, B \neq D$ szabályok eliminációja

Végül a nem Kuroda-normálformájú szabályok sémája ekkor $AB \to CD$ $(A, B, C, D \in N)$. Átalakításukhoz szabályonként egyedi W új nemterminálisokat vezetünk be és a fenti szabályt az alábbi szabályokkal szimuláljuk:

$$AB \to AW,$$
$$AW \to CW,$$
$$CW \to CD.$$

A kapott G' grammatika ekvivalens G-vel. Ugyanis az átalakított grammatikában ezen 3 szabály bármelyikének alkalmazása implikálja a másik 2 alkalmazását ebben a sorrendben.

Kuroda normálforma

5. lépés: Az $AB \to CD$, $A \neq C, B \neq D$ szabályok eliminációja

Végül a nem Kuroda-normálformájú szabályok sémája ekkor $AB \to CD$ $(A, B, C, D \in N)$. Átalakításukhoz szabályonként egyedi W új nemterminálisokat vezetünk be és a fenti szabályt az alábbi szabályokkal szimuláljuk:

$$AB \to AW,$$
$$AW \to CW,$$
$$CW \to CD.$$

A kapott G' grammatika ekvivalens G-vel. Ugyanis az átalakított grammatikában ezen 3 szabály bármelyikének alkalmazása implikálja a másik 2 alkalmazását ebben a sorrendben. Meggondolható, hogy az $AB \to AW$ szabályalkalmazás hátratolható közvetlenül az $AW \to CW$ szabályalkalmazás elé, míg a $CW \to CD$ szabályalkalmazás előrehozható közvetlenül az $AW \to CW$ szabályalkalmazás utánra.

Kuroda normálforma – példa

Példa:

$S \to C \mid AABC$

$A \to ABC \mid a$

$B \to b$

$C \to B \mid bA$

$ABC \to ABaC$

Kuroda normálforma – példa

Példa:

$S \rightarrow C \mid AABC$
$A \rightarrow ABC \mid a$
$B \rightarrow b$
$C \rightarrow B \mid bA$
$ABC \rightarrow ABaC$

1-2. lépés után:

$S \rightarrow C \mid AD$
$A \rightarrow AF \mid a$
$B \rightarrow b$
$C \rightarrow B \mid YA$
$ABC \rightarrow ABXC$
$D \rightarrow AE$
$E \rightarrow BC$
$F \rightarrow BC$
$X \rightarrow a$
$Y \rightarrow b$

Kuroda normálforma – példa

3. lépés: $H(S) = \{S, C, B\}$, $H(C) = \{C, B\}$, minden más Z nemterminálisra $H(Z) = \{Z\}$.

Kuroda normálforma – példa

3. lépés: $H(S) = \{S, C, B\}$, $H(C) = \{C, B\}$, minden más Z nemterminálisra $H(Z) = \{Z\}$.

$S \to AD \mid YA \mid b$
$A \to AF \mid a$
$B \to b$
$C \to YA \mid b$
$ABC \to ABXC$
$ACC \to ABXC$
$ASC \to ABXC$
$ABS \to ABXC$
$ACS \to ABXC$
$ASS \to ABXC$
$D \to AE$
$E \to BC$
$F \to BC$
$X \to a$
$Y \to b$

Kuroda normálforma – példa

4. lépés:

$S \to AD \mid YA \mid b$

$A \to AF \mid a$

$B \to b$

$C \to YA \mid b$

$AB \to AZ_1 \quad Z_1C \to BZ_2 \quad Z_2 \to XC$

$AC \to AZ_3 \quad Z_3C \to BZ_4 \quad Z_4 \to XC$

$AS \to AZ_5 \quad Z_5C \to BZ_6 \quad Z_6 \to XC$

$AB \to AZ_7 \quad Z_7S \to BZ_8 \quad Z_8 \to XC$

$AC \to AZ_9 \quad Z_9S \to BZ_{10} \quad Z_{10} \to XC$

$AS \to AZ_{11} \quad Z_{11}S \to BZ_{12} \quad Z_{12} \to XC$

$D \to AE$

$E \to BC$

$F \to BC$

$X \to a$

$Y \to b$

Kuroda normálforma – példa

5. lépés:

$S \to AD \mid YA \mid b$

$A \to AF \mid a$

$B \to b$

$C \to YA \mid b$

$AB \to AZ_1 \quad Z_1C \to Z_1W_1 \quad Z_1W_1 \to BW_1 \quad BW_1 \to BZ_2 \quad Z_2 \to XC$

$AC \to AZ_3 \quad Z_3C \to Z_3W_2 \quad Z_3W_2 \to BW_2 \quad BW_2 \to BZ_4 \quad Z_4 \to XC$

$AS \to AZ_5 \quad Z_5C \to Z_5W_3 \quad Z_5W_3 \to BW_3 \quad BW_3 \to BZ_6 \quad Z_6 \to XC$

$AB \to AZ_7 \quad Z_7S \to Z_7W_4 \quad Z_7W_4 \to BW_4 \quad BW_4 \to BZ_8 \quad Z_8 \to XC$

$AC \to AZ_9 \quad Z_9S \to Z_9W_5 \quad Z_9W_1 \to BW_5 \quad BW_5 \to BZ_{10} \quad Z_{10} \to XC$

$AS \to AZ_{11} \quad Z_{11}S \to Z_{11}W_6 \quad Z_{11}W_6 \to BW_6 \quad BW_6 \to BZ_{12}$

$D \to AE \qquad Z_{12} \to XC$

$E \to BC$

$F \to BC$

$X \to a$

$Y \to b$

Egy 0-típusú normálforma

Tétel

Bármely $G = \langle N, T, P, S \rangle$ 0-típusú grammatikához van vele ekvivalens G' grammatika, ahol G' minden szabályának alakjára az alábbiak valamelyike teljesül

Egy 0-típusú normálforma

Tétel

Bármely $G = \langle N, T, P, S \rangle$ 0-típusú grammatikához van vele ekvivalens G' grammatika, ahol G' minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.

Egy 0-típusú normálforma

Tétel

Bármely $G = \langle N, T, P, S\rangle$ 0-típusú grammatikához van vele ekvivalens G' grammatika, ahol G' minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $A \to a$, ahol $A \in N, a \in T$,

Egy 0-típusú normálforma

Tétel

Bármely $G = \langle N, T, P, S \rangle$ 0-típusú grammatikához van vele ekvivalens G' grammatika, ahol G' minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \rightarrow \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $A \rightarrow a$, ahol $A \in N, a \in T$,
- $A \rightarrow B$, ahol $A, B \in N$,

Egy 0-típusú normálforma

Tétel

Bármely $G = \langle N, T, P, S \rangle$ 0-típusú grammatikához van vele ekvivalens G' grammatika, ahol G' minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \rightarrow \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $A \rightarrow a$, ahol $A \in N, a \in T$,
- $A \rightarrow B$, ahol $A, B \in N$,
- $A \rightarrow BC$, ahol $A, B, C \in N$,

Egy 0-típusú normálforma

Tétel

Bármely $G = \langle N, T, P, S \rangle$ 0-típusú grammatikához van vele ekvivalens G' grammatika, ahol G' minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $A \to a$, ahol $A \in N, a \in T$,
- $A \to B$, ahol $A, B \in N$,
- $A \to BC$, ahol $A, B, C \in N$,
- $AB \to B$, ahol $A, B \in N$,

Egy 0-típusú normálforma

Tétel

Bármely $G = \langle N, T, P, S \rangle$ 0-típusú grammatikához van vele ekvivalens G' grammatika, ahol G' minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $A \to a$, ahol $A \in N, a \in T$,
- $A \to B$, ahol $A, B \in N$,
- $A \to BC$, ahol $A, B, C \in N$,
- $AB \to B$, ahol $A, B \in N$,
- $AB \to AC$, ahol $A, B, C \in N$,

Egy 0-típusú normálforma

Tétel

Bármely $G = \langle N, T, P, S \rangle$ 0-típusú grammatikához van vele ekvivalens G' grammatika, ahol G' minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $A \to a$, ahol $A \in N, a \in T$,
- $A \to B$, ahol $A, B \in N$,
- $A \to BC$, ahol $A, B, C \in N$,
- $AB \to B$, ahol $A, B \in N$,
- $AB \to AC$, ahol $A, B, C \in N$,
- $BA \to CA$, ahol $A, B, C \in N$.

Egy 0-típusú normálforma

Tétel

Bármely $G = \langle N, T, P, S \rangle$ 0-típusú grammatikához van vele ekvivalens G' grammatika, ahol G' minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $A \to a$, ahol $A \in N, a \in T$,
- $A \to B$, ahol $A, B \in N$,
- $A \to BC$, ahol $A, B, C \in N$,
- $AB \to B$, ahol $A, B \in N$,
- $AB \to AC$, ahol $A, B, C \in N$,
- $BA \to CA$, ahol $A, B, C \in N$.

A tételt nem bizonyítjuk.

Egy 0-típusú normálforma

Tétel

Bármely $G = \langle N, T, P, S \rangle$ 0-típusú grammatikához van vele ekvivalens G' grammatika, ahol G' minden szabályának alakjára az alábbiak valamelyike teljesül

- $S \to \varepsilon$, de ha van ilyen szabály, akkor más szabály nem tartalmazhatja jobboldalán S-et.
- $A \to a$, ahol $A \in N, a \in T$,
- $A \to B$, ahol $A, B \in N$,
- $A \to BC$, ahol $A, B, C \in N$,
- $AB \to B$, ahol $A, B \in N$,
- $AB \to AC$, ahol $A, B, C \in N$,
- $BA \to CA$, ahol $A, B, C \in N$.

A tételt nem bizonyítjuk.

Megjegyzés: A 0-típusú grammatikáknak többfajta normálformája ismeretes. Ez egy ε-szabály mentes változat.

Algoritmikus problémák

Akkor beszélünk algoritmikus eldöntési problémáról, ha a bemenet egy olyan objektum, mely egy adott végtelen halmazból kerülhet ki (például egy természetes szám vagy adott ábécé feletti szó), kimenete pedig igen/nem.

Algoritmikus problémák

Akkor beszélünk algoritmikus eldöntési problémáról, ha a bemenet egy olyan objektum, mely egy adott végtelen halmazból kerülhet ki (például egy természetes szám vagy adott ábécé feletti szó), kimenete pedig igen/nem.

Olyan algoritmust keresünk, amely kellően általános ahhoz, hogy a végtelen lehetséges bemenet bármelyike esetén alkalmazható legyen.

Algoritmikus problémák

Akkor beszélünk algoritmikus eldöntési problémáról, ha a bemenet egy olyan objektum, mely egy adott végtelen halmazból kerülhet ki (például egy természetes szám vagy adott ábécé feletti szó), kimenete pedig igen/nem.

Olyan algoritmust keresünk, amely kellően általános ahhoz, hogy a végtelen lehetséges bemenet bármelyike esetén alkalmazható legyen.

Egy probléma **eldönthető**, ha létezik olyan minden lehetséges bemenet esetén termináló algoritmus, amelyik a helyes választ adja.

Algoritmikus problémák

Akkor beszélünk algoritmikus eldöntési problémáról, ha a bemenet egy olyan objektum, mely egy adott végtelen halmazból kerülhet ki (például egy természetes szám vagy adott ábécé feletti szó), kimenete pedig igen/nem.

Olyan algoritmust keresünk, amely kellően általános ahhoz, hogy a végtelen lehetséges bemenet bármelyike esetén alkalmazható legyen.

Egy probléma **eldönthető**, ha létezik olyan minden lehetséges bemenet esetén termináló algoritmus, amelyik a helyes választ adja.

Megjegyzés: Az eldönthetőség fogalmát a későbbiekben precízebben definiáljuk.

Algoritmikus problémák

Akkor beszélünk algoritmikus eldöntési problémáról, ha a bemenet egy olyan objektum, mely egy adott végtelen halmazból kerülhet ki (például egy természetes szám vagy adott ábécé feletti szó), kimenete pedig igen/nem.

Olyan algoritmust keresünk, amely kellően általános ahhoz, hogy a végtelen lehetséges bemenet bármelyike esetén alkalmazható legyen.

Egy probléma **eldönthető**, ha létezik olyan minden lehetséges bemenet esetén termináló algoritmus, amelyik a helyes választ adja.

Megjegyzés: Az eldönthetőség fogalmát a későbbiekben precízebben definiáljuk.
Előző szemeszterben tanultuk:

Tétel

$\mathcal{L}_i$ zárt a reguláris bűveletekre (unió, konkatenáció, Kleene-lezárt) $(i = 0, 1, 2, 3)$

$\mathcal{L}_3$ további zártsági tulajdonságai

Tétel

$\mathcal{L}_3$ zárt a komplementerre, a metszetre és a különbségre.

$\mathcal{L}_3$ további zártsági tulajdonságai

Tétel

$\mathcal{L}_3$ zárt a komplementerre, a metszetre és a különbségre.

Bizonyítás:

- Legyen $L \in \mathcal{L}_3$, ekkor az előző tételek alapján L felismerhető egy $A = \langle Q, T, \delta, q_0, F \rangle$ véges determinisztikus automatával.

$\mathcal{L}_3$ további zártsági tulajdonságai

Tétel

$\mathcal{L}_3$ zárt a komplementerre, a metszetre és a különbségre.

Bizonyítás:

- Legyen $L \in \mathcal{L}_3$, ekkor az előző tételek alapján L felismerhető egy $A = \langle Q, T, \delta, q_0, F \rangle$ véges determinisztikus automatával. Nyilván az $A' = \langle Q, T, \delta, q_0, Q \backslash F \rangle$ véges determinisztikus automata által felismert nyelv $\overline{L}$.

$\mathcal{L}_3$ további zártsági tulajdonságai

Tétel

$\mathcal{L}_3$ zárt a komplementerre, a metszetre és a különbségre.

Bizonyítás:

- Legyen $L \in \mathcal{L}_3$, ekkor az előző tételek alapján L felismerhető egy $A = \langle Q, T, \delta, q_0, F \rangle$ véges determinisztikus automatával.

  Nyilván az $A' = \langle Q, T, \delta, q_0, Q \setminus F \rangle$ véges determinisztikus automata által felismert nyelv $\overline{L}$.

  Mivel $\overline{L}$ felismerhető véges determinisztikus automatával, ezért korábbi tételünk miatt $\overline{L} \in \mathcal{L}_3$.

$\mathcal{L}_3$ további zártsági tulajdonságai

Tétel

$\mathcal{L}_3$ zárt a komplementerre, a metszetre és a különbségre.

Bizonyítás:

- Legyen $L \in \mathcal{L}_3$, ekkor az előző tételek alapján L felismerhető egy $A = \langle Q, T, \delta, q_0, F \rangle$ véges determinisztikus automatával.

  Nyilván az $A' = \langle Q, T, \delta, q_0, Q \backslash F \rangle$ véges determinisztikus automata által felismert nyelv $\overline{L}$.

  Mivel $\overline{L}$ felismerhető véges determinisztikus automatával, ezért korábbi tételünk miatt $\overline{L} \in \mathcal{L}_3$.
- Mivel $L_1 \cap L_2 = \overline{\overline{L}_1 \cup \overline{L}_2}$, ezért a metszetre való zártság következik az unióra és a komplementerre való zártságból.

$\mathcal{L}_3$ további zártsági tulajdonságai

Tétel

$\mathcal{L}_3$ zárt a komplementerre, a metszetre és a különbségre.

Bizonyítás:

- Legyen $L \in \mathcal{L}_3$, ekkor az előző tételek alapján L felismerhető egy $A = \langle Q, T, \delta, q_0, F \rangle$ véges determinisztikus automatával.

  Nyilván az $A' = \langle Q, T, \delta, q_0, Q \backslash F \rangle$ véges determinisztikus automata által felismert nyelv $\overline{L}$.

  Mivel $\overline{L}$ felismerhető véges determinisztikus automatával, ezért korábbi tételünk miatt $\overline{L} \in \mathcal{L}_3$.
- Mivel $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$, ezért a metszetre való zártság következik az unióra és a komplementerre való zártságból.
- Mivel $L_1 \backslash L_2 = L_1 \cap \overline{L_2}$, ezért a különbségre való zártság következik a metszetre és a komplementerre való zártságból.

$\mathcal{L}_3$ további zártsági tulajdonságai

Tétel

$\mathcal{L}_3$ zárt a komplementerre, a metszetre és a különbségre.

Bizonyítás:

- Legyen $L \in \mathcal{L}_3$, ekkor az előző tételek alapján L felismerhető egy $A = \langle Q, T, \delta, q_0, F\rangle$ véges determinisztikus automatával.

  Nyilván az $A' = \langle Q, T, \delta, q_0, Q \backslash F\rangle$ véges determinisztikus automata által felismert nyelv $\overline{L}$.

  Mivel $\overline{L}$ felismerhető véges determinisztikus automatával, ezért korábbi tételünk miatt $\overline{L} \in \mathcal{L}_3$.
- Mivel $L_1 \cap L_2 = \overline{\overline{L}_1 \cup \overline{L}_2}$, ezért a metszetre való zártság következik az unióra és a komplementerre való zártságból.
- Mivel $L_1 \backslash L_2 = L_1 \cap \overline{L}_2$, ezért a különbségre való zártság következik a metszetre és a komplementerre való zártságból.

Megjegyzés: $\mathcal{L}_2$ nem zárt a metszetre. Annyi viszont igaz, hogy egy 3-típusú és egy 2-típusú nyelv metszete 2-típusú.

3-as típusú grammatikák algoritmikus problémái

Állítás

Eldönthető, hogy egy G reguláris grammatika az üres nyelvet generálja-e.

3-as típusú grammatikák algoritmikus problémái

Állítás

Eldönthető, hogy egy G reguláris grammatika az üres nyelvet generálja-e.

Bizonyítás: Az FNyF-ben tanultak szerint G-hez algoritmikusan megadható egy $L(G)$-t felismerő A VDA. A pontosan akkor nem ismer fel egyetlen szót sem, ha a kezdőállapotából minden végállapota elérhetetlen. A átmeneti gráfja véges, így ez például A gráfján egy szélességi kereséssel algoritmikusan eldönthető.

3-as típusú grammatikák algoritmikus problémái

Állítás

Eldönthető, hogy egy G reguláris grammatika az üres nyelvet generálja-e.

Bizonyítás: Az FNyF-ben tanultak szerint G-hez algoritmikusan megadható egy $L(G)$-t felismerő A VDA. A pontosan akkor nem ismer fel egyetlen szót sem, ha a kezdőállapotából minden végállapota elérhetetlen. A átmeneti gráfja véges, így ez például A gráfján egy szélességi kereséssel algoritmikusan eldönthető.

Állítás

Eldönthető, hogy két reguláris grammatika által generált nyelv diszjunkt-e.

3-as típusú grammatikák algoritmikus problémái

Állítás

Eldönthető, hogy egy G reguláris grammatika az üres nyelvet generálja-e.

Bizonyítás: Az FNyF-ben tanultak szerint G-hez algoritmikusan megadható egy $L(G)$-t felismerő A VDA. A pontosan akkor nem ismer fel egyetlen szót sem, ha a kezdőállapotából minden végállapota elérhetetlen. A átmeneti gráfja véges, így ez például A gráfján egy szélességi kereséssel algoritmikusan eldönthető.

Állítás

Eldönthető, hogy két reguláris grammatika által generált nyelv diszjunkt-e.

Bizonyítás: Generálják a G_1 és G_2 reguláris grammatikák rendre az L_1 és L_2 nyelveket. Az előbbi tétel szerint $L_1 \cap L_2$ is reguláris, így az előző állítás szerint ennek üressége eldönthető.

3-as típusú grammatikák algoritmikus problémái

Állítás

Eldönthető, hogy két reguláris grammatika ugyanazt a nyelvet generálja-e vagy sem.

3-as típusú grammatikák algoritmikus problémái

Állítás

Eldönthető, hogy két reguláris grammatika ugyanazt a nyelvet generálja-e vagy sem.

Bizonyítás:

Generálják a G_1 és G_2 reguláris grammatikák rendre az L_1 és L_2 nyelveket. Az $L_3 = (L_1 \cap \bar{L}_2) \cup (\bar{L}_1 \cap L_2)$ nyelv szintén reguláris, így van olyan G_3 reguláris grammatika, amely L_3-at generálja. Ekkor azonban $L_1 = L_2$ akkor és csak akkor, ha $L_3 = \emptyset$, amely a fentiek szerint eldönthető.

3-as típusú grammatikák algoritmikus problémái

Állítás

Eldönthető, hogy két reguláris grammatika ugyanazt a nyelvet generálja-e vagy sem.

Bizonyítás:

Generálják a G_1 és G_2 reguláris grammatikák rendre az L_1 és L_2 nyelveket. Az $L_3 = (L_1 \cap \overline{L}_2) \cup (\overline{L}_1 \cap L_2)$ nyelv szintén reguláris, így van olyan G_3 reguláris grammatika, amely L_3-at generálja. Ekkor azonban $L_1 = L_2$ akkor és csak akkor, ha $L_3 = \emptyset$, amely a fentiek szerint eldönthető.

Állítás

Eldönthető, hogy egy reguláris grammatika bővebb vagy egyenlő nyelvet generál-e mint egy másik.

3-as típusú grammatikák algoritmikus problémái

Állítás

Eldönthető, hogy két reguláris grammatika ugyanazt a nyelvet generálja-e vagy sem.

Bizonyítás:
Generálják a G_1 és G_2 reguláris grammatikák rendre az L_1 és L_2 nyelveket. Az $L_3 = (L_1 \cap \overline{L}_2) \cup (\overline{L}_1 \cap L_2)$ nyelv szintén reguláris, így van olyan G_3 reguláris grammatika, amely L_3-at generálja. Ekkor azonban $L_1 = L_2$ akkor és csak akkor, ha $L_3 = \emptyset$, amely a fentiek szerint eldönthető.

Állítás

Eldönthető, hogy egy reguláris grammatika bővebb vagy egyenlő nyelvet generál-e mint egy másik.

Bizonyítás: Generálják a G_1,G_2 reguláris grammatikák rendre az L_1,L_2 nyelveket. Az $L_3 = (L_1 \cap \overline{L}_2) = L_1 \setminus L_2$ szintén reguláris. Ennek üressége eldönthető, ami ekvivalens azzal, hogy $L_1 \subseteq L_2$.

Lineáris algoritmus a 3-as típusú szóproblémára

Szóprobléma (membership problem):

Adott a G grammatika és $u \in T^*$. $u \stackrel{?}{\in} L(G)$.

Tétel

A 3-as típusú grammatikák szóproblémája (hatékonyan) eldönthető.

Lineáris algoritmus a 3-as típusú szóproblémára

Szóprobléma (membership problem):

Adott a G grammatika és $u \in T^*$. $u \overset{?}{\in} L(G)$.

Tétel

A 3-as típusú grammatikák szóproblémája (hatékonyan) eldönthető.

Bizonyítás: Legyen $G = \langle N, T, P, S \rangle$ egy 3-as típusú grammatika normálformában adva és $u = t_1 \cdots t_n$ a levezetendő szó.

Lineáris algoritmus a 3-as típusú szóproblémára

Szóprobléma (membership problem):

Adott a G grammatika és $u \in T^*$. $u \stackrel{?}{\in} L(G)$.

Tétel

A 3-as típusú grammatikák szóproblémája (hatékonyan) eldönthető.

Bizonyítás: Legyen $G = \langle N, T, P, S \rangle$ egy 3-as típusú grammatika normálformában adva és $u = t_1 \cdots t_n$ a levezetendő szó.

Az algoritmus rekurzívan kiszámol egy N részhalmazaiból álló H_i $(0 \leqslant i \leqslant n)$ sorozatot. H_i-ból H_{i+1} (csak G-től függő) konstans időben számolható.

Lineáris algoritmus a 3-as típusú szóproblémára

Szóprobléma (membership problem):

Adott a G grammatika és $u \in T^*$. $u \stackrel{?}{\in} L(G)$.

Tétel

A 3-as típusú grammatikák szóproblémája (hatékonyan) eldönthető.

Bizonyítás: Legyen $G = \langle N, T, P, S \rangle$ egy 3-as típusú grammatika normálformában adva és $u = t_1 \cdots t_n$ a levezetendő szó.

Az algoritmus rekurzívan kiszámol egy N részhalmazaiból álló H_i $(0 \leqslant i \leqslant n)$ sorozatot. H_i-ból H_{i+1} (csak G-től függő) konstans időben számolható.

$H_0 := \{S\}$

Lineáris algoritmus a 3-as típusú szóproblémára

Szóprobléma (membership problem):

Adott a G grammatika és $u \in T^*$. $u \overset{?}{\in} L(G)$.

Tétel

A 3-as típusú grammatikák szóproblémája (hatékonyan) eldönthető.

Bizonyítás: Legyen $G = \langle N, T, P, S\rangle$ egy 3-as típusú grammatika normálformában adva és $u = t_1 \cdots t_n$ a levezetendő szó.

Az algoritmus rekurzívan kiszámol egy N részhalmazaiból álló H_i $(0 \leqslant i \leqslant n)$ sorozatot. H_i-ból H_{i+1} (csak G-től függő) konstans időben számolható.

$H_0 := \{S\} \quad H_{i+1} := \{A \in N \mid \exists\, B \in H_i \wedge B \to t_{i+1}A \in P\}$.

Lineáris algoritmus a 3-as típusú szóproblémára

Szóprobléma (membership problem):

Adott a G grammatika és $u \in T^*$. $u \stackrel{?}{\in} L(G)$.

Tétel

A 3-as típusú grammatikák szóproblémája (hatékonyan) eldönthető.

Bizonyítás: Legyen $G = \langle N, T, P, S \rangle$ egy 3-as típusú grammatika normálformában adva és $u = t_1 \cdots t_n$ a levezetendő szó.

Az algoritmus rekurzívan kiszámol egy N részhalmazaiból álló H_i $(0 \leqslant i \leqslant n)$ sorozatot. H_i-ból H_{i+1} (csak G-től függő) konstans időben számolható.

$$H_0 := \{S\} \quad H_{i+1} := \{A \in N \mid \exists\, B \in H_i \wedge B \to t_{i+1}A \in P\}.$$

Könnyen látható, hogy H_i azon nemterminálosok halmaza, melyek pontosan i levezetési lépés után a mondatforma végén állhatnak.

Lineáris algoritmus a 3-as típusú szóproblémára

Szóprobléma (membership problem):

Adott a G grammatika és $u \in T^*$. $u \stackrel{?}{\in} L(G)$.

Tétel

A 3-as típusú grammatikák szóproblémája (hatékonyan) eldönthető.

Bizonyítás: Legyen $G = \langle N, T, P, S \rangle$ egy 3-as típusú grammatika normálformában adva és $u = t_1 \cdots t_n$ a levezetendő szó.

Az algoritmus rekurzívan kiszámol egy N részhalmazaiból álló H_i $(0 \leqslant i \leqslant n)$ sorozatot. H_i-ból H_{i+1} (csak G-től függő) konstans időben számolható.

$H_0 := \{S\} \quad H_{i+1} := \{A \in N \mid \exists\, B \in H_i \wedge B \to t_{i+1}A \in P\}$.

Könnyen látható, hogy H_i azon nemterminálosok halmaza, melyek pontosan i levezetési lépés után a mondatforma végén állhatnak.

Tehát ha $F = \{A \in N \mid A \to \varepsilon \in P\}$, akkor nyilván

Lineáris algoritmus a 3-as típusú szóproblémára

Szóprobléma (membership problem):

Adott a G grammatika és $u \in T^*$. $u \overset{?}{\in} L(G)$.

Tétel

A 3-as típusú grammatikák szóproblémája (hatékonyan) eldönthető.

Bizonyítás: Legyen $G = \langle N, T, P, S \rangle$ egy 3-as típusú grammatika normálformában adva és $u = t_1 \cdots t_n$ a levezetendő szó.

Az algoritmus rekurzívan kiszámol egy N részhalmazaiból álló H_i $(0 \leqslant i \leqslant n)$ sorozatot. H_i-ból H_{i+1} (csak G-től függő) konstans időben számolható.

$H_0 := \{S\} \quad H_{i+1} := \{A \in N \mid \exists\, B \in H_i \wedge B \to t_{i+1}A \in P\}$.

Könnyen látható, hogy H_i azon nemterminálosok halmaza, melyek pontosan i levezetési lépés után a mondatforma végén állhatnak.

Tehát ha $F = \{A \in N \mid A \to \varepsilon \in P\}$, akkor nyilván

$$u \in L(G) \Leftrightarrow H_n \cap F \neq \emptyset.$$

Lineáris algoritmus a 3-as típusú szóproblémára

Példa:

$S \to aA \mid bS \mid \varepsilon$
$A \to aA \mid aS$

Lineáris algoritmus a 3-as típusú szóproblémára

Példa:

$S \to aA \mid bS \mid \varepsilon$
$A \to aA \mid aS$

abb-re $H_0 = \{S\}, H_1 = \{A\}, H_2 = \emptyset, H_3 = \emptyset,$

Lineáris algoritmus a 3-as típusú szóproblémára

Példa:

$S \rightarrow aA \mid bS \mid \varepsilon$
$A \rightarrow aA \mid aS$

abb-re $H_0 = \{S\}, H_1 = \{A\}, H_2 = \emptyset, H_3 = \emptyset$,

míg aab-re $H_0 = \{S\}$, $H_1 = \{A\}, H_2 = \{A, S\}, H_3 = \{S\}$

lesz a H_i-k sorozata.

Lineáris algoritmus a 3-as típusú szóproblémára

Példa:

$S \to aA \mid bS \mid \varepsilon$
$A \to aA \mid aS$

abb-re $H_0 = \{S\}, H_1 = \{A\}, H_2 = \emptyset, H_3 = \emptyset,$

míg *aab*-re $H_0 = \{S\}$, $H_1 = \{A\}, H_2 = \{A, S\}, H_3 = \{S\}$

lesz a H_i-k sorozata. Mivel most $F = \{S\}$, ezért *aab* generálható, *abb* viszont nem.

Lineáris algoritmus a 3-as típusú szóproblémára

Példa:

$S \to aA \mid bS \mid \varepsilon$
$A \to aA \mid aS$

abb-re $H_0 = \{S\}, H_1 = \{A\}, H_2 = \emptyset, H_3 = \emptyset$,

míg *aab*-re $H_0 = \{S\}$, $H_1 = \{A\}, H_2 = \{A, S\}, H_3 = \{S\}$

lesz a H_i-k sorozata. Mivel most $F = \{S\}$, ezért *aab* generálható, *abb* viszont nem.

Vegyük észre, hogy $H_i \in \mathcal{P}(N)$ és minden $1 \leqslant i \leqslant n-1$-re H_{i+1} csak H_i-től és t_{i+1}-től függ.

Lineáris algoritmus a 3-as típusú szóproblémára

Példa:

$S \to aA \mid bS \mid \varepsilon$
$A \to aA \mid aS$

abb-re $H_0 = \{S\}, H_1 = \{A\}, H_2 = \emptyset, H_3 = \emptyset,$

míg aab-re $H_0 = \{S\}$, $H_1 = \{A\}, H_2 = \{A, S\}, H_3 = \{S\}$

lesz a H_i-k sorozata. Mivel most $F = \{S\}$, ezért aab generálható, abb viszont nem.

Vegyük észre, hogy $H_i \in \mathcal{P}(N)$ és minden $1 \leqslant i \leqslant n-1$-re H_{i+1} csak H_i-től és t_{i+1}-től függ.Azaz a $\{H_i\}_{0 \leqslant i \leqslant n}$ halmazok meghatározása egy VDA segítségével automatizálható.

Lineáris algoritmus a 3-as típusú szóproblémára

Példa:

$S \to aA \mid bS \mid \varepsilon$
$A \to aA \mid aS$

abb-re $H_0 = \{S\}, H_1 = \{A\}, H_2 = \emptyset, H_3 = \emptyset,$

míg aab-re $H_0 = \{S\},\ H_1 = \{A\}, H_2 = \{A, S\}, H_3 = \{S\}$

lesz a H_i-k sorozata. Mivel most $F = \{S\}$, ezért aab generálható, abb viszont nem.

Vegyük észre, hogy $H_i \in \mathcal{P}(N)$ és minden $1 \leqslant i \leqslant n-1$-re H_{i+1} csak H_i-től és t_{i+1}-től függ.Azaz a $\{H_i\}_{0 \leqslant i \leqslant n}$ halmazok meghatározása egy VDA segítségével automatizálható.

	a	b
$\leftrightarrows \{S\}$	$\{A\}$	$\{S\}$
$\{A\}$	$\{S, A\}$	$\{\}$
$\leftarrow \{S, A\}$	$\{S, A\}$	$\{S\}$
$\{\}$	$\{\}$	$\{\}$

Lineáris algoritmus a 3-as típusú szóproblémára

Megjegyzések:

- Ez a VDA nem más, mint a G-hez készített VNDA determinizáltja. $\mathcal{P}(N)$ állapothalmazzal, $H_0 = \{S\}$ kezdőállapottal, $\delta(H, a) = \{B \in N \mid \exists\, A \in H : A \rightarrow aB \in P\}$ állapotátmenet-függvénnyel $(H \in \mathcal{P}(N)$ és $a \in T)$ és $F' = \{H \subseteq N \mid H \cap F \neq \emptyset\}$ elfogadó állapothalmazzal, ahol $F = \{A \in N \mid A \rightarrow \varepsilon \in P\}$.

Lineáris algoritmus a 3-as típusú szóproblémára

Megjegyzések:

- Ez a VDA nem más, mint a G-hez készített VNDA determinizáltja. $\mathcal{P}(N)$ állapothalmazzal, $H_0 = \{S\}$ kezdőállapottal, $\delta(H, a) = \{B \in N \mid \exists\, A \in H : A \to aB \in P\}$ állapotátmenet-függvénnyel $(H \in \mathcal{P}(N)$ és $a \in T)$ és $F' = \{H \subseteq N \mid H \cap F \neq \emptyset\}$ elfogadó állapothalmazzal, ahol $F = \{A \in N \mid A \to \varepsilon \in P\}$.
- Az algoritmus **lineáris** időben eldönti, $u \overset{?}{\in} L(G)$, hiszen minden egyes H_i halmaz kiszámításának ideje csak G-től ($|u|$-tól nem) függő konstans.

Lineáris algoritmus a 3-as típusú szóproblémára

Megjegyzések:

- Ez a VDA nem más, mint a G-hez készített VNDA determinizáltja. $\mathcal{P}(N)$ állapothalmazzal, $H_0 = \{S\}$ kezdőállapottal, $\delta(H, a) = \{B \in N \mid \exists\, A \in H : A \to aB \in P\}$ állapotátmenet-függvénnyel $(H \in \mathcal{P}(N)$ és $a \in T)$ és $F' = \{H \subseteq N \mid H \cap F \neq \varnothing\}$ elfogadó állapothalmazzal, ahol $F = \{A \in N \mid A \to \varepsilon \in P\}$.
- Az algoritmus **lineáris** időben eldönti, $u \overset{?}{\in} L(G)$, hiszen minden egyes H_i halmaz kiszámításának ideje csak G-től ($|u|$-tól nem) függő konstans.
- Amennyiben G méretéhez képest hosszú szóra vagy több szóra szeretnénk a kérdést eldönteni érdemes lehet előfeldolgozó lépésként a fenti automatát elkészíteni.

Lineáris algoritmus a 3-as típusú szóproblémára

Megjegyzések:

- Ez a VDA nem más, mint a G-hez készített VNDA determinizáltja. $\mathcal{P}(N)$ állapothalmazzal, $H_0 = \{S\}$ kezdőállapottal, $\delta(H, a) = \{B \in N \mid \exists\, A \in H : A \to aB \in P\}$ állapotátmenet-függvénnyel $(H \in \mathcal{P}(N)$ és $a \in T)$ és $F' = \{H \subseteq N \mid H \cap F \neq \emptyset\}$ elfogadó állapothalmazzal, ahol $F = \{A \in N \mid A \to \varepsilon \in P\}$.
- Az algoritmus **lineáris** időben eldönti, $u \overset{?}{\in} L(G)$, hiszen minden egyes H_i halmaz kiszámításának ideje csak G-től ($|u|$-tól nem) függő konstans.
- Amennyiben G méretéhez képest hosszú szóra vagy több szóra szeretnénk a kérdést eldönteni érdemes lehet előfeldolgozó lépésként a fenti automatát elkészíteni.
- Másrészt, kevés, kisméretű szó és nagyméretű grammatika esetén hatékonyabb lehet a H_i halmazok közvetlen kiszámítása.

A környezetfüggetlen grammatikák szóproblémája

Állítás: Legyen $G = \langle T, N, P, S \rangle$ környezetfüggetlen grammatika és egy $u \in T^*$ egy szó. Eldönthető, hogy $u \overset{?}{\in} L(G)$.

A környezetfüggetlen grammatikák szóproblémája

Állítás: Legyen $G = \langle T, N, P, S \rangle$ környezetfüggetlen grammatika és egy $u \in T^*$ egy szó. Eldönthető, hogy $u \stackrel{?}{\in} L(G)$.

Bizonyítás: Tanultuk, hogy algoritmikusan előállítható egy G-vel ekvivalens Chomsky normálformájú grammatika. Legyen $n = |u|$.

A környezetfüggetlen grammatikák szóproblémája

Állítás: Legyen $G = \langle T, N, P, S \rangle$ környezetfüggetlen grammatika és egy $u \in T^*$ egy szó. Eldönthető, hogy $u \overset{?}{\in} L(G)$.

Bizonyítás: Tanultuk, hogy algoritmikusan előállítható egy G-vel ekvivalens Chomsky normálformájú grammatika. Legyen $n = |u|$.

$n = 0$ esetén $u \in L(G) \Leftrightarrow S \rightarrow \varepsilon \in P$.

A környezetfüggetlen grammatikák szóproblémája

Állítás: Legyen $G = \langle T, N, P, S \rangle$ környezetfüggetlen grammatika és egy $u \in T^*$ egy szó. Eldönthető, hogy $u \overset{?}{\in} L(G)$.

Bizonyítás: Tanultuk, hogy algoritmikusan előállítható egy G-vel ekvivalens Chomsky normálformájú grammatika. Legyen $n = |u|$.

$n = 0$ esetén $u \in L(G) \Leftrightarrow S \to \varepsilon \in P$.

Teljes indukcióval könnyen látható, hogy $n \geqslant 1$ esetén u levezetése $n - 1$ darab „$A \to BC$" típusú és n darab „$A \to a$" típusú szabály alkalmazásából kell álljon. Mivel adott grammatika és n esetén véges sok ilyen levezetés van ezek megvizsgálása után el tudjuk dönteni, hogy $u \overset{?}{\in} L(G)$.

A környezetfüggetlen grammatikák szóproblémája

Az előző algoritmus alapján csak exponenciális felső becslést adhatunk a szóprobléma eldöntésének hatékonyságára.

A környezetfüggetlen grammatikák szóproblémája

Az előző algoritmus alapján csak exponenciális felső becslést adhatunk a szóprobléma eldöntésének hatékonyságára.

u levezetéseinek keresését azonban hatékonyabban is megszervezhetjük egy „bottom-up" ún. „dinamikus programozás" segítségével, az így kapott algoritmus már polinomiális (azaz hatékony) lesz.

A környezetfüggetlen grammatikák szóproblémája

Az előző algoritmus alapján csak exponenciális felső becslést adhatunk a szóprobléma eldöntésének hatékonyságára.

u levezetéseinek keresését azonban hatékonyabban is megszervezhetjük egy „bottom-up” ún. „dinamikus programozás” segítségével, az így kapott algoritmus már polinomiális (azaz hatékony) lesz.

John Cocke, Daniel Younger és Tadao Kasami algoritmusa (CYK algoritmus) egy Chomsky normálformában adott grammatika esetében $|G| \cdot n^3$ nagyságrendű lépésben eldönti a szóproblémát.

A környezetfüggetlen grammatikák szóproblémája

Az előző algoritmus alapján csak exponenciális felső becslést adhatunk a szóprobléma eldöntésének hatékonyságára.

u levezetéseinek keresését azonban hatékonyabban is megszervezhetjük egy „bottom-up” ún. „dinamikus programozás” segítségével, az így kapott algoritmus már polinomiális (azaz hatékony) lesz.

John Cocke, Daniel Younger és Tadao Kasami algoritmusa (CYK algoritmus) egy Chomsky normálformában adott grammatika esetében $|G| \cdot n^3$ nagyságrendű lépésben eldönti a szóproblémát.

Tetszőleges környezetfüggetlen grammatika esetén tehát a Chomsky normálformára hozással kombinálva a CYK algoritmus hatékonyságára $|G|^2 \cdot n^3$ nagyságrendű felső becslés adható.

CYK algoritmus 2-es típusú szóproblémára

Cocke-Younger-Kasami (CYK) algoritmus

CYK algoritmus 2-es típusú szóproblémára

Cocke-Younger-Kasami (CYK) algoritmus

Input: Egy környezetfüggetlen $G = \langle T, N, P, S \rangle$ grammatika **Chomsky-normálformában** adva és egy $u \in T^*$ szó.

CYK algoritmus 2-es típusú szóproblémára

Cocke-Younger-Kasami (CYK) algoritmus

Input: Egy környezetfüggetlen $G = \langle T, N, P, S \rangle$ grammatika **Chomsky-normálformában** adva és egy $u \in T^*$ szó.

Output: $u \overset{?}{\in} L(G)$

CYK algoritmus 2-es típusú szóproblémára

Cocke-Younger-Kasami (CYK) algoritmus

Input: Egy környezetfüggetlen $G = \langle T, N, P, S \rangle$ grammatika **Chomsky-normálformában** adva és egy $u \in T^*$ szó.

Output: $u \overset{?}{\in} L(G)$

Legyen $u = t_1 \ldots t_n,\ t_i \in T, n \geqslant 1$. Legyen A_i a $P_i \in P$ szabály bal-, β_i pedig a jobboldala. ($1 \leqslant i \leqslant |P|$, $A_i \in N$, $\beta_i \in T \cup N^2$.)

CYK algoritmus 2-es típusú szóproblémára

Cocke-Younger-Kasami (CYK) algoritmus

Input: Egy környezetfüggetlen $G = \langle T, N, P, S \rangle$ grammatika Chomsky-normálformában adva és egy $u \in T^*$ szó.

Output: $u \overset{?}{\in} L(G)$

Legyen $u = t_1 \ldots t_n$, $t_i \in T, n \geqslant 1$. Legyen A_i a $P_i \in P$ szabály bal-, β_i pedig a jobboldala. ($1 \leqslant i \leqslant |P|$, $A_i \in N$, $\beta_i \in T \cup N^2$.)

A CYK algoritmus rekurzíven definiál $H_{i,j}$, $1 \leqslant i \leqslant j \leqslant n$ halmazokat $(j-i)$ szerint növekvő sorrendben.

CYK algoritmus 2-es típusú szóproblémára

Cocke-Younger-Kasami (CYK) algoritmus

Input: Egy környezetfüggetlen $G = \langle T, N, P, S\rangle$ grammatika Chomsky-normálformában adva és egy $u \in T^*$ szó.

Output: $u \overset{?}{\in} L(G)$

Legyen $u = t_1 \dots t_n$, $t_i \in T, n \geqslant 1$. Legyen A_i a $P_i \in P$ szabály bal-, β_i pedig a jobboldala. ($1 \leqslant i \leqslant |P|$, $A_i \in N$, $\beta_i \in T \cup N^2$.)

A CYK algoritmus rekurzíven definiál $H_{i,j}$, $1 \leqslant i \leqslant j \leqslant n$ halmazokat $(j-i)$ szerint növekvő sorrendben.

$$H_{i,i} := \{A_j \mid \beta_j = t_i\}$$

CYK algoritmus 2-es típusú szóproblémára

Cocke-Younger-Kasami (CYK) algoritmus

Input: Egy környezetfüggetlen $G = \langle T, N, P, S\rangle$ grammatika Chomsky-normálformában adva és egy $u \in T^*$ szó.

Output: $u \overset{?}{\in} L(G)$

Legyen $u = t_1 \dots t_n$, $t_i \in T, n \geqslant 1$. Legyen A_i a $P_i \in P$ szabály bal-, β_i pedig a jobboldala. ($1 \leqslant i \leqslant |P|$, $A_i \in N$, $\beta_i \in T \cup N^2$.)

A CYK algoritmus rekurzíven definiál $H_{i,j}$, $1 \leqslant i \leqslant j \leqslant n$ halmazokat $(j-i)$ szerint növekvő sorrendben.

$$H_{i,i} := \{A_j \mid \beta_j = t_i\}$$

$$H_{i,j} := \{A_k \mid \beta_k \in \bigcup_{h=i}^{j-1} H_{i,h}H_{h+1,j}\} \quad (i < j)$$

CYK algoritmus 2-es típusú szóproblémára

Cocke-Younger-Kasami (CYK) algoritmus

Input: Egy környezetfüggetlen $G = \langle T, N, P, S \rangle$ grammatika **Chomsky-normálformában** adva és egy $u \in T^*$ szó.

Output: $u \stackrel{?}{\in} L(G)$

Legyen $u = t_1 \dots t_n$, $t_i \in T, n \geqslant 1$. Legyen A_i a $P_i \in P$ szabály bal-, β_i pedig a jobboldala. ($1 \leqslant i \leqslant |P|$, $A_i \in N$, $\beta_i \in T \cup N^2$.)

A CYK algoritmus rekurzíven definiál $H_{i,j}$, $1 \leqslant i \leqslant j \leqslant n$ halmazokat $(j-i)$ szerint növekvő sorrendben.

$$H_{i,i} := \{A_j \mid \beta_j = t_i\}$$

$$H_{i,j} := \{A_k \mid \beta_k \in \bigcup_{h=i}^{j-1} H_{i,h} H_{h+1,j}\} \quad (i < j)$$

$u \in L(G) \iff S \in H_{1,n}$.

A CYK algoritmus helyessége

Az algoritmus helyességéhez elég belátni, hogy minden $1 \leqslant i \leqslant j \leqslant n$ esetén $H_{i,j} = \{X \in N \mid X \Rightarrow_G^* t_i \cdots t_j\}$.

A CYK algoritmus helyessége

Az algoritmus helyességéhez elég belátni, hogy minden $1 \leqslant i \leqslant j \leqslant n$ esetén $H_{i,j} = \{X \in N \mid X \Rightarrow_G^* t_i \cdots t_j\}$.

Ezt $(j - i)$-re vonatkozó teljes indukcióval bizonyítjuk.

A CYK algoritmus helyessége

Az algoritmus helyességéhez elég belátni, hogy minden $1 \leqslant i \leqslant j \leqslant n$ esetén $H_{i,j} = \{X \in N \mid X \Rightarrow_G^* t_i \cdots t_j\}$.

Ezt $(j-i)$-re vonatkozó teljes indukcióval bizonyítjuk.

$j - i = 0$ esetén világos, hogy t_i éppen $H_{i,i}$ nemterminálisaiból vezethető le.

A CYK algoritmus helyessége

Az algoritmus helyességéhez elég belátni, hogy minden $1 \leqslant i \leqslant j \leqslant n$ esetén $H_{i,j} = \{X \in N \mid X \Rightarrow_G^* t_i \cdots t_j\}$.

Ezt $(j-i)$-re vonatkozó teljes indukcióval bizonyítjuk.

$j - i = 0$ esetén világos, hogy t_i éppen $H_{i,i}$ nemterminálisaiból vezethető le.

Tegyük fel, hogy az állítás igaz minden olyan $1 \leqslant k \leqslant \ell \leqslant n$-re, melyre $\ell - k < j - i$ és legyen $1 \leqslant i < j \leqslant n$.

A CYK algoritmus helyessége

Az algoritmus helyességéhez elég belátni, hogy minden $1 \leqslant i \leqslant j \leqslant n$ esetén $H_{i,j} = \{X \in N \mid X \Rightarrow_G^* t_i \cdots t_j\}$.

Ezt $(j-i)$-re vonatkozó teljes indukcióval bizonyítjuk.

$j - i = 0$ esetén világos, hogy t_i éppen $H_{i,i}$ nemterminálisaiból vezethető le.

Tegyük fel, hogy az állítás igaz minden olyan $1 \leqslant k \leqslant \ell \leqslant n$-re, melyre $\ell - k < j - i$ és legyen $1 \leqslant i < j \leqslant n$.

Tekintsük egy $X \Rightarrow^* t_i \cdots t_j$ levezetést.

A CYK algoritmus helyessége

Az algoritmus helyességéhez elég belátni, hogy minden $1 \leqslant i \leqslant j \leqslant n$ esetén $H_{i,j} = \{X \in N \mid X \Rightarrow_G^* t_i \cdots t_j\}$.

Ezt $(j - i)$-re vonatkozó teljes indukcióval bizonyítjuk.

$j - i = 0$ esetén világos, hogy t_i éppen $H_{i,i}$ nemterminálisaiból vezethető le.

Tegyük fel, hogy az állítás igaz minden olyan $1 \leqslant k \leqslant \ell \leqslant n$-re, melyre $\ell - k < j - i$ és legyen $1 \leqslant i < j \leqslant n$.

Tekintsük egy $X \Rightarrow^* t_i \cdots t_j$ levezetést. Mivel a levezetendő szó legalább 2 hosszú ezért a levezetés első lépése $X \Rightarrow YZ$ valamely $Y, Z \in N$-re.

A CYK algoritmus helyessége

Az algoritmus helyességéhez elég belátni, hogy minden $1 \leqslant i \leqslant j \leqslant n$ esetén $H_{i,j} = \{X \in N \,|\, X \Rightarrow_G^* t_i \cdots t_j\}$.

Ezt $(j-i)$-re vonatkozó teljes indukcióval bizonyítjuk.

$j - i = 0$ esetén világos, hogy t_i éppen $H_{i,i}$ nemterminálisaiból vezethető le.

Tegyük fel, hogy az állítás igaz minden olyan $1 \leqslant k \leqslant \ell \leqslant n$-re, melyre $\ell - k < j - i$ és legyen $1 \leqslant i < j \leqslant n$.

Tekintsük egy $X \Rightarrow^* t_i \cdots t_j$ levezetést. Mivel a levezetendő szó legalább 2 hosszú ezért a levezetés első lépése $X \Rightarrow YZ$ valamely $Y, Z \in N$-re. Ekkor létezik egy olyan $i \leqslant h < j$, melyre $Y \Rightarrow^* t_i \cdots t_h$ és $Z \Rightarrow^* t_{h+1} \cdots t_j$.

A CYK algoritmus helyessége

Az algoritmus helyességéhez elég belátni, hogy minden $1 \leqslant i \leqslant j \leqslant n$ esetén $H_{i,j} = \{X \in N \mid X \Rightarrow_G^* t_i \cdots t_j\}$.

Ezt $(j-i)$-re vonatkozó teljes indukcióval bizonyítjuk.

$j - i = 0$ esetén világos, hogy t_i éppen $H_{i,i}$ nemterminálisaiból vezethető le.

Tegyük fel, hogy az állítás igaz minden olyan $1 \leqslant k \leqslant \ell \leqslant n$-re, melyre $\ell - k < j - i$ és legyen $1 \leqslant i < j \leqslant n$.

Tekintsük egy $X \Rightarrow^* t_i \cdots t_j$ levezetést. Mivel a levezetendő szó legalább 2 hosszú ezért a levezetés első lépése $X \Rightarrow YZ$ valamely $Y, Z \in N$-re. Ekkor létezik egy olyan $i \leqslant h < j$, melyre $Y \Rightarrow^* t_i \cdots t_h$ és $Z \Rightarrow^* t_{h+1} \cdots t_j$. Mivel $h - i < j - i$ és $j - (h+1) < j - i$ ezért az indukciós feltevés szerint $Y \in H_{i,h}, Z \in H_{h+1,j}$ és így $X \in H_{i,j}$.

A CYK algoritmus helyessége

Az algoritmus helyességéhez elég belátni, hogy minden $1 \leqslant i \leqslant j \leqslant n$ esetén $H_{i,j} = \{X \in N \mid X \Rightarrow_G^* t_i \cdots t_j\}$.

Ezt $(j-i)$-re vonatkozó teljes indukcióval bizonyítjuk.

$j - i = 0$ esetén világos, hogy t_i éppen $H_{i,i}$ nemterminálisaiból vezethető le.

Tegyük fel, hogy az állítás igaz minden olyan $1 \leqslant k \leqslant \ell \leqslant n$-re, melyre $\ell - k < j - i$ és legyen $1 \leqslant i < j \leqslant n$.

Tekintsük egy $X \Rightarrow^* t_i \cdots t_j$ levezetést. Mivel a levezetendő szó legalább 2 hosszú ezért a levezetés első lépése $X \Rightarrow YZ$ valamely $Y, Z \in N$-re. Ekkor létezik egy olyan $i \leqslant h < j$, melyre $Y \Rightarrow^* t_i \cdots t_h$ és $Z \Rightarrow^* t_{h+1} \cdots t_j$. Mivel $h - i < j - i$ és $j - (h+1) < j - i$ ezért az indukciós feltevés szerint $Y \in H_{i,h}, Z \in H_{h+1,j}$ és így $X \in H_{i,j}$.

Fordítva, ha $X \in H_{i,j}$ $(j > i)$, akkor van olyan $i \leqslant h < j$ és $Y \in H_{i,h}, Z \in H_{h+1,j}$, melyre $X \rightarrow YZ \in P$,

A CYK algoritmus helyessége

Az algoritmus helyességéhez elég belátni, hogy minden $1 \leqslant i \leqslant j \leqslant n$ esetén $H_{i,j} = \{X \in N \mid X \Rightarrow_G^* t_i \cdots t_j\}$.

Ezt $(j-i)$-re vonatkozó teljes indukcióval bizonyítjuk.

$j - i = 0$ esetén világos, hogy t_i éppen $H_{i,i}$ nemterminálisaiból vezethető le.

Tegyük fel, hogy az állítás igaz minden olyan $1 \leqslant k \leqslant \ell \leqslant n$-re, melyre $\ell - k < j - i$ és legyen $1 \leqslant i < j \leqslant n$.

Tekintsük egy $X \Rightarrow^* t_i \cdots t_j$ levezetést. Mivel a levezetendő szó legalább 2 hosszú ezért a levezetés első lépése $X \Rightarrow YZ$ valamely $Y, Z \in N$-re. Ekkor létezik egy olyan $i \leqslant h < j$, melyre $Y \Rightarrow^* t_i \cdots t_h$ és $Z \Rightarrow^* t_{h+1} \cdots t_j$. Mivel $h - i < j - i$ és $j - (h+1) < j - i$ ezért az indukciós feltevés szerint $Y \in H_{i,h}, Z \in H_{h+1,j}$ és így $X \in H_{i,j}$.

Fordítva, ha $X \in H_{i,j}$ $(j > i)$, akkor van olyan $i \leqslant h < j$ és $Y \in H_{i,h}, Z \in H_{h+1,j}$, melyre $X \rightarrow YZ \in P$, azaz $Y \Rightarrow^* t_i \cdots t_h$ és $Z \Rightarrow^* t_{h+1} \cdots t_j$.

A CYK algoritmus helyessége

Az algoritmus helyességéhez elég belátni, hogy minden $1 \leqslant i \leqslant j \leqslant n$ esetén $H_{i,j} = \{X \in N \mid X \Rightarrow_G^* t_i \cdots t_j\}$.

Ezt $(j-i)$-re vonatkozó teljes indukcióval bizonyítjuk.

$j - i = 0$ esetén világos, hogy t_i éppen $H_{i,i}$ nemterminálisaiból vezethető le.

Tegyük fel, hogy az állítás igaz minden olyan $1 \leqslant k \leqslant \ell \leqslant n$-re, melyre $\ell - k < j - i$ és legyen $1 \leqslant i < j \leqslant n$.

Tekintsük egy $X \Rightarrow^* t_i \cdots t_j$ levezetést. Mivel a levezetendő szó legalább 2 hosszú ezért a levezetés első lépése $X \Rightarrow YZ$ valamely $Y, Z \in N$-re. Ekkor létezik egy olyan $i \leqslant h < j$, melyre $Y \Rightarrow^* t_i \cdots t_h$ és $Z \Rightarrow^* t_{h+1} \cdots t_j$. Mivel $h - i < j - i$ és $j - (h+1) < j - i$ ezért az indukciós feltevés szerint $Y \in H_{i,h}, Z \in H_{h+1,j}$ és így $X \in H_{i,j}$.

Fordítva, ha $X \in H_{i,j}$ $(j > i)$, akkor van olyan $i \leqslant h < j$ és $Y \in H_{i,h}, Z \in H_{h+1,j}$, melyre $X \to YZ \in P$, azaz $Y \Rightarrow^* t_i \cdots t_h$ és $Z \Rightarrow^* t_{h+1} \cdots t_j$. Ekkor $X \Rightarrow YZ \Rightarrow^* t_i \cdots t_h Z \Rightarrow^* t_i \cdots t_j$.

CYK algoritmus

Példa: A $G = \langle\{S, A, B, C, U, V, W, X, Y, Z\}, \{a, b, c\}, P, S\rangle$ Chomsky normálformájú grammatika esetén CYK algoritmussal döntsük el, hogy az *aabbcc* szót generálja-e G, ahol P:

$S \rightarrow AB \mid BC$
$A \rightarrow XA \mid a$
$X \rightarrow a$
$C \rightarrow YC \mid c$
$Y \rightarrow c$
$B \rightarrow UV \mid VW \mid XS$
$U \rightarrow XX$
$W \rightarrow YY \mid XS$
$V \rightarrow ZZ$
$Z \rightarrow b$

CYK algoritmus

Példa: A $G = \langle \{S, A, B, C, U, V, W, X, Y, Z\}, \{a, b, c\}, P, S\rangle$ Chomsky normálformájú grammatika esetén CYK algoritmussal döntsük el, hogy az *aabbcc* szót generálja-e G, ahol P:

$S \to AB \mid BC$
$A \to XA \mid a$
$X \to a$
$C \to YC \mid c$
$Y \to c$
$B \to UV \mid VW \mid XS$
$U \to XX$
$W \to YY \mid XS$
$V \to ZZ$
$Z \to b$

t_1	t_2	$\cdots$	t_n

CYK algoritmus

Példa: A $G = \langle \{S, A, B, C, U, V, W, X, Y, Z\}, \{a, b, c\}, P, S \rangle$ Chomsky normálformájú grammatika esetén CYK algoritmussal döntsük el, hogy az *aabbcc* szót generálja-e G, ahol P:

$S \rightarrow AB \mid BC$
$A \rightarrow XA \mid a$
$X \rightarrow a$
$C \rightarrow YC \mid c$
$Y \rightarrow c$
$B \rightarrow UV \mid VW \mid XS$
$U \rightarrow XX$
$W \rightarrow YY \mid XS$
$V \rightarrow ZZ$
$Z \rightarrow b$

$H_{1,1}$	$H_{2,2}$		$\cdots$	$H_{n,n}$
t_1	t_2	$\cdots$		t_n

CYK algoritmus

Példa: A $G = \langle \{S, A, B, C, U, V, W, X, Y, Z\}, \{a, b, c\}, P, S \rangle$ Chomsky normálformájú grammatika esetén CYK algoritmussal döntsük el, hogy az *aabbcc* szót generálja-e G, ahol P:

$S \to AB \mid BC$
$A \to XA \mid a$
$X \to a$
$C \to YC \mid c$
$Y \to c$
$B \to UV \mid VW \mid XS$
$U \to XX$
$W \to YY \mid XS$
$V \to ZZ$
$Z \to b$

$$\begin{array}{ccccccc} & H_{1,2} & & H_{2,3} & & \cdots & H_{n-1,n} & \\ H_{1,1} & & H_{2,2} & & & \cdots & & H_{n,n} \\ \hline t_1 & & t_2 & & \cdots & & & t_n \end{array}$$

CYK algoritmus

Példa: A $G = \langle \{S, A, B, C, U, V, W, X, Y, Z\}, \{a, b, c\}, P, S \rangle$ Chomsky normálformájú grammatika esetén CYK algoritmussal döntsük el, hogy az *aabbcc* szót generálja-e G, ahol P:

$S \to AB \mid BC$
$A \to XA \mid a$
$X \to a$
$C \to YC \mid c$
$Y \to c$
$B \to UV \mid VW \mid XS$
$U \to XX$
$W \to YY \mid XS$
$V \to ZZ$
$Z \to b$

$$
\begin{array}{cccccc}
 & & H_{1,3} & H_{2,4} & & \\
 & H_{1,2} & H_{2,3} & & \cdots & H_{n-1,n} \\
H_{1,1} & H_{2,2} & & & \cdots & H_{n,n} \\
\hline
t_1 & t_2 & & \cdots & & t_n
\end{array}
$$

CYK algoritmus

Példa: A $G = \langle \{S, A, B, C, U, V, W, X, Y, Z\}, \{a, b, c\}, P, S \rangle$ Chomsky normálformájú grammatika esetén CYK algoritmussal döntsük el, hogy az *aabbcc* szót generálja-e G, ahol P:

$S \to AB \mid BC$
$A \to XA \mid a$
$X \to a$
$C \to YC \mid c$
$Y \to c$
$B \to UV \mid VW \mid XS$
$U \to XX$
$W \to YY \mid XS$
$V \to ZZ$
$Z \to b$

$$
\begin{array}{ccccccccc}
 & & & & & \vdots & & & \\
 & & H_{1,3} & & H_{2,4} & & & & \\
 & H_{1,2} & & H_{2,3} & & & \cdots & & H_{n-1,n} & \\
H_{1,1} & & H_{2,2} & & & & & \cdots & & H_{n,n} \\
\hline
t_1 & & t_2 & & & \cdots & & & & t_n
\end{array}
$$

CYK algoritmus

Példa: A $G = \langle \{S, A, B, C, U, V, W, X, Y, Z\}, \{a, b, c\}, P, S\rangle$
Chomsky normálformájú grammatika esetén CYK algoritmussal döntsük el, hogy az *aabbcc* szót generálja-e G, ahol P:

$S \to AB \mid BC$
$A \to XA \mid a$
$X \to a$
$C \to YC \mid c$
$Y \to c$
$B \to UV \mid VW \mid XS$
$U \to XX$
$W \to YY \mid XS$
$V \to ZZ$
$Z \to b$

$$
\begin{array}{cccccc}
 & & & H_{1,n-1} & H_{2,n} & \\
 & & & \vdots & & \\
 & H_{1,3} & H_{2,4} & & & \\
H_{1,2} & H_{2,3} & & \cdots & H_{n-1,n} & \\
H_{1,1} & H_{2,2} & & & \cdots & H_{n,n} \\
\hline
t_1 & t_2 & & \cdots & & t_n
\end{array}
$$

CYK algoritmus

Példa: A $G = \langle \{S, A, B, C, U, V, W, X, Y, Z\}, \{a, b, c\}, P, S\rangle$ Chomsky normálformájú grammatika esetén CYK algoritmussal döntsük el, hogy az *aabbcc* szót generálja-e G, ahol P:

$S \rightarrow AB \mid BC$
$A \rightarrow XA \mid a$
$X \rightarrow a$
$C \rightarrow YC \mid c$
$Y \rightarrow c$
$B \rightarrow UV \mid VW \mid XS$
$U \rightarrow XX$
$W \rightarrow YY \mid XS$
$V \rightarrow ZZ$
$Z \rightarrow b$

$$
\begin{array}{cccccccccc}
 & & & & & H_{1,n} & & & & \\
 & & & & H_{1,n-1} & & H_{2,n} & & & \\
 & & & & & \vdots & & & & \\
 & & H_{1,3} & & H_{2,4} & & & & & \\
 & H_{1,2} & & H_{2,3} & & & & \cdots & & H_{n-1,n} \\
H_{1,1} & & H_{2,2} & & & & & & \cdots & & H_{n,n} \\
\hline
t_1 & & t_2 & & & \cdots & & & & & t_n
\end{array}
$$

CYK algoritmus

$$S \to AB \mid BC$$
$$A \to XA \mid a$$
$$X \to a$$
$$C \to YC \mid c$$
$$Y \to c$$
$$B \to UV \mid VW \mid XS$$
$$U \to XX$$
$$W \to YY \mid XS$$
$$V \to ZZ$$
$$Z \to b$$

				$\{Y, C\}$	
a	a	b	b	c	c

CYK algoritmus

$$S \rightarrow AB \mid BC$$
$$A \rightarrow XA \mid a$$
$$X \rightarrow a$$
$$C \rightarrow YC \mid c$$
$$Y \rightarrow c$$
$$B \rightarrow UV \mid VW \mid XS$$
$$U \rightarrow XX$$
$$W \rightarrow YY \mid XS$$
$$V \rightarrow ZZ$$
$$Z \rightarrow b$$

$\{A, U\}$

$\{A, X\}$	$\{A, X\}$	$\{Z\}$	$\{Z\}$	$\{Y, C\}$	$\{Y, C\}$
a	a	b	b	c	c

$$\{A, X\}\{A, X\} = \{AA, AX, XA, XX\}$$

CYK algoritmus

$$
\begin{aligned}
S &\to AB \mid BC \\
A &\to XA \mid a \\
X &\to a \\
C &\to YC \mid c \\
Y &\to c \\
B &\to UV \mid VW \mid XS \\
U &\to XX \\
W &\to YY \mid XS \\
V &\to ZZ \\
Z &\to b
\end{aligned}
$$

	$\{\ \}$				
$\{A,U\}$	$\{\ \}$	$\{V\}$	$\{\ \}$	$\{C,W\}$	
$\{A,X\}$	$\{A,X\}$	$\{Z\}$	$\{Z\}$	$\{Y,C\}$	$\{Y,C\}$
a	a	b	b	c	c

$\{A,X\}\{\ \} = \{\ \}$ $\{A,U\}\{Z\} = \{AZ, UZ\}$

CYK algoritmus

$S \to AB \mid BC$
$A \to XA \mid a$
$X \to a$
$C \to YC \mid c$
$Y \to c$
$B \to UV \mid VW \mid XS$
$U \to XX$
$W \to YY \mid XS$
$V \to ZZ$
$Z \to b$

$$\{B\}$$

$$\{\ \} \quad \{\ \} \quad \{\ \} \quad \{\ \}$$

$$\{A, U\} \quad \{\ \} \quad \{V\} \quad \{\ \} \quad \{C, W\}$$

$$\{A, X\} \quad \{A, X\} \quad \{Z\} \quad \{Z\} \quad \{Y, C\} \quad \{Y, C\}$$

$$a \quad a \quad b \quad b \quad c \quad c$$

$$\{Z\}\{\ \} = \{\ \} \quad \{V\}\{C, W\} = \{VC, VW\} \quad \{\ \}\{Y, C\} = \{\ \}$$

CYK algoritmus

$$
\begin{aligned}
S &\rightarrow \boxed{AB} \mid BC \\
A &\rightarrow XA \mid a \\
X &\rightarrow a \\
C &\rightarrow YC \mid c \\
Y &\rightarrow c \\
B &\rightarrow UV \mid VW \mid XS \\
U &\rightarrow XX \\
W &\rightarrow YY \mid XS \\
V &\rightarrow ZZ \\
Z &\rightarrow b
\end{aligned}
$$

$$\{S\}$$

$$\{B\} \quad \{\,\} \quad \{B\}$$

$$\{\,\} \quad \{\,\} \quad \{\,\} \quad \{\,\}$$

$$\{A, U\} \quad \{\,\} \quad \{V\} \quad \{\,\} \quad \{C, W\}$$

$$\{A, X\} \quad \{A, X\} \quad \{Z\} \quad \{Z\} \quad \{Y, C\} \quad \{Y, C\}$$

$$a \quad a \quad b \quad b \quad c \quad c$$

$$\{\ AB, XB\ \} \quad \{\,\} \quad \{\,\} \quad \{\,\}$$

CYK algoritmus

$$
\begin{aligned}
S &\to \boxed{AB} \mid \boxed{BC} \\
A &\to XA \mid a \\
X &\to a \\
C &\to YC \mid c \\
Y &\to c \\
B &\to UV \mid VW \mid \boxed{XS} \\
U &\to XX \\
W &\to YY \mid \boxed{XS} \\
V &\to ZZ \\
Z &\to b
\end{aligned}
$$

$$\{\, S, B, W \,\}$$

$$\{S\} \quad \{S\}$$

$$\{B\} \quad \{\ \} \quad \{B\}$$

$$\{\ \} \quad \{\ \} \quad \{\ \} \quad \{\ \}$$

$$\{A, U\} \quad \{\ \} \quad \{V\} \quad \{\ \} \quad \{C, W\}$$

$$\{A, X\} \quad \{A, X\} \quad \{Z\} \quad \{Z\} \quad \{Y, C\} \quad \{Y, C\}$$

$$a \quad a \quad b \quad b \quad c \quad c$$

$$\{\ AS, XS\} \quad \{\ AB, UB\ \} \quad \{\ \} \quad \{\ BC, BW\ \} \quad \{\ SY, SC\ \}$$

CYK algoritmus

$$
\begin{aligned}
S &\to AB \mid BC \\
A &\to XA \mid a \\
X &\to a \\
C &\to YC \mid c \\
Y &\to c \\
B &\to UV \mid VW \mid XS \\
U &\to XX \\
W &\to YY \mid XS \\
V &\to ZZ \\
Z &\to b
\end{aligned}
$$

$\{S, B, W\}$					
$\{S\}$	$\{S\}$				
$\{B\}$	$\{\,\}$	$\{B\}$			
$\{\,\}$	$\{\,\}$	$\{\,\}$	$\{\,\}$		
$\{A, U\}$	$\{\,\}$	$\{V\}$	$\{\,\}$	$\{C, W\}$	
$\{A, X\}$	$\{A, X\}$	$\{Z\}$	$\{Z\}$	$\{Y, C\}$	$\{Y, C\}$
a	a	b	b	c	c

CYK algoritmus

$$
\begin{aligned}
S &\to AB \mid BC \\
A &\to XA \mid a \\
X &\to a \\
C &\to YC \mid c \\
Y &\to c \\
B &\to UV \mid VW \mid XS \\
U &\to XX \\
W &\to YY \mid XS \\
V &\to ZZ \\
Z &\to b
\end{aligned}
$$

$$
\begin{array}{ccccccccccc}
 & & & & & \{S, B, W\} & & & & & \\
 & & & & \{S\} & & \{S\} & & & & \\
 & & & \{B\} & & \{\,\} & & \{B\} & & & \\
 & & \{\,\} & & \{\,\} & & \{\,\} & & \{\,\} & & \\
 & \{A, U\} & & \{\,\} & & \{V\} & & \{\,\} & & \{C, W\} & \\
\{A, X\} & & \{A, X\} & & \{Z\} & & \{Z\} & & \{Y, C\} & & \{Y, C\} \\
\hline
a & & a & & b & & b & & c & & c
\end{array}
$$

Mivel $S \in H_{1,6}$, ezért $aabbcc \in L(G)$.

CYK algoritmus

$$
\begin{aligned}
S &\to AB \mid BC \\
A &\to XA \mid a \\
X &\to a \\
C &\to YC \mid c \\
Y &\to c \\
B &\to UV \mid VW \mid XS \\
U &\to XX \\
W &\to YY \mid XS \\
V &\to ZZ \\
Z &\to b
\end{aligned}
$$

$\{S, B, W\}$					
$\{S\}$	$\{S\}$				
$\{B\}$	$\{\ \}$	$\{B\}$			
$\{\ \}$	$\{\ \}$	$\{\ \}$	$\{\ \}$		
$\{A, U\}$	$\{\ \}$	$\{V\}$	$\{\ \}$	$\{C, W\}$	
$\{A, X\}$	$\{A, X\}$	$\{Z\}$	$\{Z\}$	$\{Y, C\}$	$\{Y, C\}$
a	a	b	b	c	c

Mivel $S \in H_{1,6}$, ezért $aabbcc \in L(G)$.
Visszafejthető például a következő levezetés:
$S \Rightarrow AB \Rightarrow XAB \Rightarrow XAVW \Rightarrow XAZZW \Rightarrow XAZZYY \Rightarrow^* aabbcc.$

2-típusú grammatikák algoritmikus kérdései

Következmény

Az alábbi algoritmikus kérdések eldönthetők

2-típusú grammatikák algoritmikus kérdései

Következmény

Az alábbi algoritmikus kérdések eldönthetők

1. Adott egy G 2-es típusú grammatika és egy L véges nyelv. $L \overset{?}{\subseteq} L(G)$.

2-típusú grammatikák algoritmikus kérdései

Következmény

Az alábbi algoritmikus kérdések eldönthetők

1. Adott egy G 2-es típusú grammatika és egy L véges nyelv. $L \overset{?}{\subseteq} L(G)$.
2. Adott egy G 2-es típusú grammatika és egy L véges nyelv. $L \cap L(G) \overset{?}{=} \emptyset$.

2-típusú grammatikák algoritmikus kérdései

Következmény

Az alábbi algoritmikus kérdések eldönthetők

1. Adott egy G 2-es típusú grammatika és egy L véges nyelv. $L \overset{?}{\subseteq} L(G)$.
2. Adott egy G 2-es típusú grammatika és egy L véges nyelv. $L \cap L(G) \overset{?}{=} \varnothing$.

FNyF-en tanultuk, hogy az alábbi algoritmikus kérdések szintén eldönthetők (nagy Bar-Hillel lemma következményei):

2-típusú grammatikák algoritmikus kérdései

Következmény

Az alábbi algoritmikus kérdések eldönthetők

1. Adott egy G 2-es típusú grammatika és egy L véges nyelv. $L \overset{?}{\subseteq} L(G)$.
2. Adott egy G 2-es típusú grammatika és egy L véges nyelv. $L \cap L(G) \overset{?}{=} \emptyset$.

FNyF-en tanultuk, hogy az alábbi algoritmikus kérdések szintén eldönthetők (nagy Bar-Hillel lemma következményei):

Tétel

1. Eldönthető, hogy egy környezetfüggetlen grammatika végtelen nyelvet generál-e vagy sem.

2-típusú grammatikák algoritmikus kérdései

Következmény

Az alábbi algoritmikus kérdések eldönthetők

1. Adott egy G 2-es típusú grammatika és egy L véges nyelv. $L \overset{?}{\subseteq} L(G)$.
2. Adott egy G 2-es típusú grammatika és egy L véges nyelv. $L \cap L(G) \overset{?}{=} \emptyset$.

FNyF-en tanultuk, hogy az alábbi algoritmikus kérdések szintén eldönthetők (nagy Bar-Hillel lemma következményei):

Tétel

1. Eldönthető, hogy egy környezetfüggetlen grammatika végtelen nyelvet generál-e vagy sem.
2. Eldönthető, hogy egy környezetfüggetlen grammatika az üres nyelvet generálja-e vagy sem.

Algoritmikus eldönthetetlen 2-típusú kérdések

A 2-es típusú grammatikák esetében **vannak algoritmikusan eldönthetetlen problémák**, azaz olyan eldöntési („igen”/„nem” kimenetű) problémák, amelyekre nem létezik olyan mindig termináló algoritmus, amelyik éppen a probléma „igen”-példányaira ad igen választ.

Algoritmikus eldönthetetlen 2-típusú kérdések

A 2-es típusú grammatikák esetében **vannak algoritmikusan eldönthetetlen problémák**, azaz olyan eldöntési („igen”/„nem” kimenetű) problémák, amelyekre nem létezik olyan mindig termináló algoritmus, amelyik éppen a probléma „igen”-példányaira ad igen választ. Ezekre még visszatérünk egy későbbi előadáson.

Algoritmikus eldönthetetlen 2-típusú kérdések

A 2-es típusú grammatikák esetében **vannak algoritmikusan eldönthetetlen problémák**, azaz olyan eldöntési („igen”/„nem” kimenetű) problémák, amelyekre nem létezik olyan mindig termináló algoritmus, amelyik éppen a probléma „igen”-példányaira ad igen választ. Ezekre még visszatérünk egy későbbi előadáson.

1. **Input:** G_1, G_2 2-típusú grammatikák

   Output: $L(G_1) \stackrel{?}{=} L(G_2)$

Algoritmikus eldönthetetlen 2-típusú kérdések

A 2-es típusú grammatikák esetében **vannak algoritmikusan eldönthetetlen problémák**, azaz olyan eldöntési („igen”/„nem” kimenetű) problémák, amelyekre nem létezik olyan mindig termináló algoritmus, amelyik éppen a probléma „igen”-példányaira ad igen választ. Ezekre még visszatérünk egy későbbi előadáson.

1. **Input:** G_1, G_2 2-típusú grammatikák
   Output: $L(G_1) \stackrel{?}{=} L(G_2)$
2. **Input:** G_1, G_2 2-típusú grammatikák
   Output: $L(G_1) \stackrel{?}{\subseteq} L(G_2)$

Algoritmikus eldönthetetlen 2-típusú kérdések

A 2-es típusú grammatikák esetében **vannak algoritmikusan eldönthetetlen problémák**, azaz olyan eldöntési („igen”/„nem” kimenetű) problémák, amelyekre nem létezik olyan mindig termináló algoritmus, amelyik éppen a probléma „igen”-példányaira ad igen választ. Ezekre még visszatérünk egy későbbi előadáson.

1. **Input:** G_1, G_2 2-típusú grammatikák
   Output: $L(G_1) \stackrel{?}{=} L(G_2)$
2. **Input:** G_1, G_2 2-típusú grammatikák
   Output: $L(G_1) \stackrel{?}{\subseteq} L(G_2)$
3. **Input:** G_1, G_2 2-típusú grammatikák
   Output: $L(G_1) \cap L(G_2) \stackrel{?}{=} \emptyset$

Algoritmikus eldönthetetlen 2-típusú kérdések

A 2-es típusú grammatikák esetében **vannak algoritmikusan eldönthetetlen problémák**, azaz olyan eldöntési („igen"/„nem" kimenetű) problémák, amelyekre nem létezik olyan mindig termináló algoritmus, amelyik éppen a probléma „igen"-példányaira ad igen választ. Ezekre még visszatérünk egy későbbi előadáson.

1. **Input:** G_1, G_2 2-típusú grammatikák
   Output: $L(G_1) \stackrel{?}{=} L(G_2)$
2. **Input:** G_1, G_2 2-típusú grammatikák
   Output: $L(G_1) \stackrel{?}{\subseteq} L(G_2)$
3. **Input:** G_1, G_2 2-típusú grammatikák
   Output: $L(G_1) \cap L(G_2) \stackrel{?}{=} \emptyset$
4. **Input:** $G = \langle N, T, P, S \rangle$ 2-típusú grammatika
   Output: $L(G) \stackrel{?}{=} T^*$

Algoritmikus eldönthetetlen 2-típusú kérdések

A 2-es típusú grammatikák esetében **vannak algoritmikusan eldönthetetlen problémák**, azaz olyan eldöntési („igen”/„nem” kimenetű) problémák, amelyekre nem létezik olyan mindig termináló algoritmus, amelyik éppen a probléma „igen”-példányaira ad igen választ. Ezekre még visszatérünk egy későbbi előadáson.

1. **Input:** G_1, G_2 2-típusú grammatikák
   Output: $L(G_1) \stackrel{?}{=} L(G_2)$
2. **Input:** G_1, G_2 2-típusú grammatikák
   Output: $L(G_1) \stackrel{?}{\subseteq} L(G_2)$
3. **Input:** G_1, G_2 2-típusú grammatikák
   Output: $L(G_1) \cap L(G_2) \stackrel{?}{=} \emptyset$
4. **Input:** $G = \langle N, T, P, S \rangle$ 2-típusú grammatika
   Output: $L(G) \stackrel{?}{=} T^*$
5. **Input:** G 2-típusú grammatika
   Output: G egyértelmű grammatika-e.

1-típusú grammatikák szóproblémája

Tétel

Eldönthető, hogy egy $G = \langle N, T, P, S\rangle$ hossz-nemcsökkentő grammatika és $u \in T^*$ szó esetén $u \in L(G)$ teljesül-e.

1-típusú grammatikák szóproblémája

Tétel

Eldönthető, hogy egy $G = \langle N, T, P, S \rangle$ hossz-nemcsökkentő grammatika és $u \in T^*$ szó esetén $u \in L(G)$ teljesül-e.

Bizonyítás: Ha $u = \varepsilon$, akkor $u \in L(G) \Leftrightarrow S \to \varepsilon \in P$.

1-típusú grammatikák szóproblémája

Tétel

Eldönthető, hogy egy $G = \langle N, T, P, S\rangle$ hossz-nemcsökkentő grammatika és $u \in T^*$ szó esetén $u \in L(G)$ teljesül-e.

Bizonyítás: Ha $u = \varepsilon$, akkor $u \in L(G) \Leftrightarrow S \to \varepsilon \in P$.

$n = |u| \geqslant 1$ esetén legyen $r = \sum_{i=1}^{n} |T \cup N|^i$. Ekkor r a $T \cup N$ halmaz legfeljebb n hosszú, nemüres szavainak száma.

1-típusú grammatikák szóproblémája

Tétel

Eldönthető, hogy egy $G = \langle N, T, P, S \rangle$ hossz-nemcsökkentő grammatika és $u \in T^*$ szó esetén $u \in L(G)$ teljesül-e.

Bizonyítás: Ha $u = \varepsilon$, akkor $u \in L(G) \Leftrightarrow S \to \varepsilon \in P$.

$n = |u| \geqslant 1$ esetén legyen $r = \sum_{i=1}^{n} |T \cup N|^i$. Ekkor r a $T \cup N$ halmaz legfeljebb n hosszú, nemüres szavainak száma.

Mivel G hossz-nemcsökkentő, ezért u levezetései nem tartalmaznak n-nél hosszabb mondatformát,így u minden r-nél hosszabb levezetése tartalmaz ismétlődő mondatformát.

1-típusú grammatikák szóproblémája

Tétel

Eldönthető, hogy egy $G = \langle N, T, P, S\rangle$ hossz-nemcsökkentő grammatika és $u \in T^*$ szó esetén $u \in L(G)$ teljesül-e.

Bizonyítás: Ha $u = \varepsilon$, akkor $u \in L(G) \Leftrightarrow S \to \varepsilon \in P$.

$n = |u| \geqslant 1$ esetén legyen $r = \sum_{i=1}^{n} |T \cup N|^i$. Ekkor r a $T \cup N$ halmaz legfeljebb n hosszú, nemüres szavainak száma.

Mivel G hossz-nemcsökkentő, ezért u levezetései nem tartalmaznak n-nél hosszabb mondatformát,így u minden r-nél hosszabb levezetése tartalmaz ismétlődő mondatformát.

Ebből következően ha $S \Rightarrow_G^* u$, akkor u-nak létezik legfeljebb r hosszú levezetése is, hiszen egy levezetésben az ismétlődő mondatformák közötti levezetést kihagyva ugyanannak a szónak egy rövidebb levezetését kapjuk.

1-típusú grammatikák szóproblémája

Tétel

Eldönthető, hogy egy $G = \langle N, T, P, S\rangle$ hossz-nemcsökkentő grammatika és $u \in T^*$ szó esetén $u \in L(G)$ teljesül-e.

Bizonyítás: Ha $u = \varepsilon$, akkor $u \in L(G) \Leftrightarrow S \to \varepsilon \in P$.

$n = |u| \geqslant 1$ esetén legyen $r = \sum_{i=1}^{n} |T \cup N|^i$. Ekkor r a $T \cup N$ halmaz legfeljebb n hosszú, nemüres szavainak száma.

Mivel G hossz-nemcsökkentő, ezért u levezetései nem tartalmaznak n-nél hosszabb mondatformát,így u minden r-nél hosszabb levezetése tartalmaz ismétlődő mondatformát.

Ebből következően ha $S \Rightarrow_G^* u$, akkor u-nak létezik legfeljebb r hosszú levezetése is, hiszen egy levezetésben az ismétlődő mondatformák közötti levezetést kihagyva ugyanannak a szónak egy rövidebb levezetését kapjuk.

Tehát $u \in L(G)$, akkor és csak akkor, ha G legfeljebb r hosszú levezetéssel generálható. Ezek viszont algoritmikusan előállíthatók.

0-típusú grammatikák szóproblémája

Kevés pozitív eredmény mondható 0-típusú nyelvek algoritmikus kérdései kapcsán.

0-típusú grammatikák szóproblémája

Kevés pozitív eredmény mondható 0-típusú nyelvek algoritmikus kérdései kapcsán.

Bizonyítás nélkül megemlítjük, hogy a 0-típusú grammatikák szóproblémája algoritmikusan eldönthetetlen.

0-típusú grammatikák szóproblémája

Kevés pozitív eredmény mondható 0-típusú nyelvek algoritmikus kérdései kapcsán.

Bizonyítás nélkül megemlítjük, hogy a 0-típusú grammatikák szóproblémája algoritmikusan eldönthetetlen.

Egy parciálisan eldöntő (helyes választ adó, azonban nem mindig termináló) algoritmust azonban készíthetünk.

0-típusú grammatikák szóproblémája

Kevés pozitív eredmény mondható 0-típusú nyelvek algoritmikus kérdései kapcsán.

Bizonyítás nélkül megemlítjük, hogy a 0-típusú grammatikák szóproblémája algoritmikusan eldönthetetlen.

Egy parciálisan eldöntő (helyes választ adó, azonban nem mindig termináló) algoritmust azonban készíthetünk.

Legyen $G = \langle N, T, P, S \rangle$ 0-típusú grammatika és $u \in T^*$. Készíthetünk egy végtelen gráfot, melynek csúcsai $(N \cup T)^*$ elemeivel címkézettek és $x \in (N \cup T)^*$-ból akkor és csak akkor van irányított él $y \in (N \cup T)^*$-ba, ha $x \Rightarrow_G y$. A gráf minden csúcsa véges kifokú, hiszen P véges és egy szabály baloldala legfeljebb annyiszor fordulhat elő a mondatformában, mint amennyi a szó hossza.

0-típusú grammatikák szóproblémája

Kevés pozitív eredmény mondható 0-típusú nyelvek algoritmikus kérdései kapcsán.

Bizonyítás nélkül megemlítjük, hogy a 0-típusú grammatikák szóproblémája algoritmikusan eldönthetetlen.

Egy parciálisan eldöntő (helyes választ adó, azonban nem mindig termináló) algoritmust azonban készíthetünk.

Legyen $G = \langle N, T, P, S \rangle$ 0-típusú grammatika és $u \in T^*$. Készíthetünk egy végtelen gráfot, melynek csúcsai $(N \cup T)^*$ elemeivel címkézettek és $x \in (N \cup T)^*$-ból akkor és csak akkor van irányított él $y \in (N \cup T)^*$-ba, ha $x \Rightarrow_G y$. A gráf minden csúcsa véges kifokú, hiszen P véges és egy szabály baloldala legfeljebb annyiszor fordulhat elő a mondatformában, mint amennyi a szó hossza.

Tehát $u \in L(G)$ akkor és csak akkor, ha ebben a gráfban egy S-ből indított szélességi keresés megtalálja u-t. (Ha $u \notin L(G)$, akkor tipikusan nem terminál az algoritmus.)