Számítási modellek

1. előadás

Számítási modellek – a kezdetek

- Alex Thue 1910-es évek, sztringek algebrai, kombinatorikai tulajdonságai $\mapsto$ a sztringeket átíró *szabályrendszer*
- Alan Turing és kortársai 1930-as évek közepe: Turing gép, kiszámíthatóság; első számítási modellek (matematikai gépek) és kifejező erejük
- 40-es évek: véges automata formalizálása, 50-es évek vége: veremautomata formalizálása
- Noam Chomsky 1950-es évek a formális grammatika fogalma
- 1950-es évek: Backus-Naur Forma
- gépi fordítási projektek, matematikai nyelvészethez kapcsolódó munkák, 1950-es 60-as évek
- Rabin és Scott: formális eszközök kifejlesztése számítási modellek kifejezőerejének és korlátainak vizsgálatára
- Chomsky-féle nyelvhierarchia, 1956-64
- 60-as évek: időigény, tárigény, bonyolultsági osztályok, NP-teljesség
- új, nemkonvencionális modellek

Alapfogalmak, jelölések

ábécé, betű, szó, szó hossza

Ábécé: Egy véges, nemüres halmaz.
Az ábécé elemeit **betűk**nek nevezzük.

Egy V ábécé elemeiből képzett véges sorozatokat V **feletti szavaknak** vagy sztringeknek nevezzük. Egy $u = t_1 \cdots t_n$ szóban lévő betűk számát (n) a szó **hosszának** nevezzük. Jelölés: $|u| = n$.
A 0 hosszú sorozat jelölése ε, ezt **üres szónak** nevezzük ($|\varepsilon| = 0$).

Példa:
Legyen $V = \{a, b\}$, ekkor a és b a V ábécé két betűje. *abba* és *aabba* egy-egy V feletti szó. $|aabba| = 5$. *aabba* és *baaba* különböző szavak, bár mindkettő 5 hosszú valamint 3 a-t és 2 b-t tartalmaz. A betűk sorrendje számít!

Alapfogalmak, jelölések

az összes szavak halmaza

V^* jelöli a V **ábécé feletti szavak halmazát**, beleértve az üres szót is.

$V^+ = V^* \backslash \{\varepsilon\}$ a V **ábécé feletti, nemüres szavak halmazát** jelöli.

Példa: $V = \{a, b\}$, ekkor
$V^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$.

V^* szavainak egy lehetséges felsorolása a **hosszlexikografikus (shortlex) rendezés** szerinti:

- Feltesszük, hogy az ábécé rendezett. (A fenti példában, mondjuk a előbb van, mint b.)
- A rövidebb szó mindig megelőzi a hosszabbat.
- Az azonos hosszúságú szavak az ábécé rendezettsége alapján meghatározott ábécésorrendben követik egymást.

Ez a rendezés egyértelműen meghatározza a szavak sorrendjét.

Műveletek szavakon

konkatenáció

Legyen V egy ábécé továbbá legyenek $u = s_1 \cdots s_n$ és $v = t_1 \cdots t_k$ V feletti szavak (azaz legyen $s_1, \ldots, s_n, t_1, \ldots, t_k \in V$). Ekkor az $uv := s_1 \cdots s_n t_1 \cdots t_k$ szót az u és v szavak **konkatenáltjának** nevezzük. (Az u szó betűi után írjuk a v szó betűit.)

Nyilván $|uv| = |u| + |v|$.

Példa:
Legyen $V = \{a, b\}$, valamint $u = abb$ és $v = aaaba$ egy-egy V feletti szó. Ekkor $uv = abbaaaba$ illetve $vu = aaabaabb$.

Azaz a konkatenáció **általában nem kommutatív**, de asszociatív (azaz $u(vw) = (uv)w$ teljesül minden $u, v, w \in V^*$-ra)

Műveletek szavakon

i-edik hatvány

Legyen i nemnegatív egész szám és legyen u egy V ábécé feletti szó ($u \in V^*$). Az u szó **i-edik hatványa** alatt az u szó i darab példányának konkatenáltját értjük és u^i -vel jelöljük.

Konvenció szerint $u^0 := \varepsilon$. Ekkor $u^{n+k} = u^n u^k$ teljesül ($k, n \in \mathbb{N}$)

Példa:
Legyen $V = \{a, b\}$ és legyen $u = abb$. Ekkor $u^0 = \varepsilon$, $u^1 = abb$, $u^2 = abbabb$ és $u^3 = abbabbabb$.

$(ab)^3 \neq a^3 b^3$! ! ! !

$(ab)^3 = ababab$, $a^3 b^3 = aaabbb$.

Műveletek szavakon

tükörkép

Legyen u egy V ábécé feletti szó. Az u szó u^{-1}-gyel jelölt **tükörképén** vagy fordítottján azt a szót értjük, amelyet úgy kapunk, hogy u betűit fordított sorrendben írjuk le.
Azaz ha $u = a_1 \cdots a_n$, $a_i \in V$, $1 \leqslant i \leqslant n$, akkor $u^{-1} = a_n \cdots a_1$.

Példa:
Legyen $V = \{a, b\}$, valamint $u = abba$ és $v = baabba$ egy-egy V ábécé feletti szó. Ekkor $u^{-1} = abba$ és $v^{-1} = abbaab$.

Ha $u = u^{-1}$, akkor u-t **palindrom** tulajdonságúnak, vagy palindrómának nevezzük.

Tehát $abba$ egy palindróma.

Alapfogalmak, jelölések

részszó, prefix, suffix

Legyen V egy ábécé és legyenek u és v szavak V felett. Az u szó a v szó **részszava**, ha $v = xuy$ teljesül, valamely $x, y \in V^*$ szavakra.

Ha $x = \varepsilon$, akkor u-t a v szó **prefixének**, ha pedig $y = \varepsilon$, akkor u-t a v szó **suffixének** nevezzük.

Példa:

Legyen $V = \{a, b\}$ és $u = abba$.

u részszavai: $\varepsilon, a, b, ab, bb, ba, abb, bba, abba$.

u prefixei: $\varepsilon, a, ab, abb, abba$.

u suffixei: $\varepsilon, a, ba, bba, abba$.

abb **nem suffixe** u**-nak!!!** A suffix nem a tükörkép szó prefixe!!!

Alapfogalmak, jelölések

nyelv

Legyen V egy ábécé, V^* egy L részhalmazát V feletti **nyelvnek** nevezzük.
Az üres nyelv (nyelv, amely egyetlen szót sem tartalmaz) jelölése $\varnothing$.
Egy V ábécé feletti nyelv véges nyelv, ha véges számú szót tartalmaz, ellenkező esetben végtelen.

Példák:
Legyen $V = \{a, b\}$ egy ábécé.

- $L_1 = \{a, ba, \varepsilon\}$.
- $L_2 = \{a^i b^i \mid i \geqslant 0\}$.
- $L_3 = \{uu^{-1} \mid u \in V^*\}$.
- $L_4 = \{u \mid u \in \{a, b\}^+, u\text{-ban ugyanannyi } a \text{ van, mint } b\}$.

L_1 véges nyelv, a többi végtelen.

Nyelvekre vonatkozó műveletek

halmazműveletek

Legyen V egy ábécé és legyenek L_1, L_2 nyelvek V felett (vagyis $L_1 \subseteq V^*$ és $L_2 \subseteq V^*$).

$L_1 \cup L_2 = \{u \mid u \in L_1 \text{ vagy } u \in L_2\}$ (az L_1 és az L_2 nyelv **uniója**)

$L_1 \cap L_2 = \{u \mid u \in L_1 \text{ és } u \in L_2\}$ (az L_1 és L_2 nyelv **metszete**)

$L_1 - L_2 = \{u \mid u \in L_1 \text{ és } u \notin L_2\}$ (az L_1 és az L_2 nyelv **különbsége**)

Az $L \subseteq V^*$ nyelv **komplementere** a V ábécére nézve az $\bar{L} = V^* - L$ nyelv.

Nyelvekre vonatkozó műveletek

konkatenáció

Legyen V egy ábécé és $L_1, L_2 \subseteq V^*$. Az L_1 és az L_2 nyelvek **konkatenációján** az $L_1L_2 = \{u_1u_2 \mid u_1 \in L_1, u_2 \in L_2\}$ nyelvet értjük. Asszociatív, de nyilván **nem kommutatív** művelet.

Példák:
$V = \{a, b\}$, $L_1 = \{ab, bb\}$, $L_2 = \{\varepsilon, a, bba\}$,

$L_1L_2 = \{ab\varepsilon, bb\varepsilon, aba, bba, abbba, bbbba\}$
$= \{ab, bb, aba, bba, abbba, bbbba\}$. (mindenkit mindenkivel!)

$L_3 = \{a^{2n}b^{2n} \mid n \in \mathbb{N}\}$, $L_4 = \{a^{3n}b^{3n} \mid n \in \mathbb{N}\}$

$L_3L_4 = \{a^{2n}b^{2n}a^{3k}b^{3k} \mid n, k \in \mathbb{N}\}$.

Nyelvekre vonatkozó műveletek

tükörkép nyelv, i-edik hatvány

Legyen V egy ábécé és legyen $L \subseteq V^*$. Ekkor $L^{-1} = \{u^{-1} \mid u \in L\}$ a **tükörképe** (megfordítása) az L nyelvnek.
Egy L nyelv *i*-**edik hatványát** $L^0 := \{\varepsilon\}$ és $L^i := LL^{i-1}$ $(i \geqslant 1)$ definiálják. Azaz L^i jelöli az L i-edik iterációját a konkatenáció műveletére nézve.

Példa: $L = \{a, bb\}$.

$L^0 = \{\varepsilon\}$,
$L^1 = \{a, bb\}$,
$L^2 = \{a\,a, a\,bb, bb\,a, bb\,bb\}$,
$L^3 = \{a\,a\,a, a\,a\,bb, a\,bb\,a, a\,bb\,bb, bb\,a\,a, bb\,a\,bb, bb\,bb\,a, bb\,bb\,bb\}$,

Nyelvekre vonatkozó műveletek

Kleene lezárt

Az L nyelv **iteratív lezártja** (vagy röviden lezártja vagy Kleene-lezártja) alatt az $L^* = \bigcup_{i \geqslant 0} L^i$ nyelvet értjük.

Az L nyelv **pozitív lezártja** alatt az $L^+ = \bigcup_{i \geqslant 1} L^i$ nyelvet értjük.

Észrevétel:
Nyilvánvalóan, $L^+ = L^*$, ha $\varepsilon \in L$ és $L^+ = L^* - \{\varepsilon\}$ ha $\varepsilon \notin L$.

Példa: $L = \{a, bb\}$.

$L^0 = \{\varepsilon\}$,
$L^1 = \{a, bb\}$,
$L^2 = \{a\,a, a\,bb, bb\,a, bb\,bb\}$,
$L^3 = \{a\,a\,a, a\,a\,bb, a\,bb\,a, a\,bb\,bb, bb\,a\,a, bb\,a\,bb, bb\,bb\,a, bb\,bb\,bb\}$,
$L^4 = \{a\,a\,a\,a, a\,a\,a\,bb, a\,a\,bb\,a, \ldots\}$, $\ldots$

Ezen szavak uniója együttesen alkotja L^*-t. L^+ ettől csak annyiban tér el, hogy nincs benne az ε szó.

Nyelvcsaládok

Ha X egy halmaz, jelölje $\mathcal{P}(X)$ az X halmaz hatványhalmazát, azaz $\mathcal{P}(X) = \{A \mid A \subseteq X\}$.

Nyelvcsalád (vagy nyelvosztály) alatt nyelveknek egy halmazát értjük.

Tehát ha V egy ábécé:
- $a \in V$: betű
- $u \in V^*$: szó
- $L \subseteq V^*$ vagy $L \in \mathcal{P}(V^*)$: nyelv
- $\mathcal{L} \subseteq P(V^*)$ vagy $\mathcal{L} \in \mathcal{P}(\mathcal{P}(V^*))$: nyelvcsalád

Példa: $V = \{a, b\}$
- véges nyelvek nyelvcsaládja
- csak b betűvel kezdődő szavakat tartalmazó nyelvek nyelvcsaládja
- reguláris kifejezésekkel leírható nyelvek nyelvcsaládja

Grammatikák

Definíció

Egy $G = \langle N, T, P, S \rangle$ rendezett négyest **grammatikának** (generatív grammatikán vagy (generatív) nyelvtannak) nevezünk ha

- N és T diszjunkt véges ábécék (azaz $N \cap T = \varnothing$). N elemeit **nemterminális**, T elemeit pedig **terminális** szimbólumoknak nevezzük.
- $S \in N$ a grammatika **kezdőszimbóluma**.
- P rendezett (x, y) párok véges halmaza, ahol $x, y \in (N \cup T)^*$ és x legalább egy nemterminális szimbólumot tartalmaz.

A P halmaz elemeit **szabályoknak** (vagy átírási szabályoknak vagy produkcióknak) hívjuk. A gyakorlatban az (x, y) jelölés helyett szinte mindig az $x \to y$ jelölést használjuk amennyiben $\to$ nem eleme $N \cup T$-nek.

Grammatikák

Példák:

$G_1 = \langle \{S, A, B\}, \{a, b, c\}, \{S \to c, S \to AB, A \to aA, B \to \varepsilon, abb \to aSb\}, S \rangle$ **nem** grammatika, mivel minden szabály baloldalának tartalmaznia kell legalább egy N-beli szimbólumot.

$G_2 = \langle \{S, A, B, C\}, \{a, b, c\}, \{S \to a, S \to AB, A \to Ab, B \to \varepsilon, aCA \to aSc\}, S \rangle$ grammatika.

Generatív grammatikák

egylépéses levezetés

Definíció

Legyen $G = \langle N, T, P, S \rangle$ egy generatív grammatika és legyen $u, v \in (N \cup T)^*$. A v szó közvetlenül vagy **egy lépésben levezethető** az u szóból G -ben, jelölése $u \Rightarrow_G v$, ha $u = u_1 x u_2$ és $v = u_1 y u_2$,ahol $u_1, u_2 \in (N \cup T)^*$ és $x \to y \in P$.

Példa:

$G_2 = \langle \{S, A, B, C\}, \{a, b, c\}, \{S \to a, S \to AB, A \to Ab, B \to \varepsilon, aCA \to aSc\}, S \rangle$.

Ekkor $BB\mathbf{aCA}a \Rightarrow BB\mathbf{aSc}a$, $\mathbf{A}BB \Rightarrow \mathbf{Ab}BB$, $B\mathbf{B} \Rightarrow B$.

Reláció reflexív, tranzitív lezártja

- Legyenek $X_1, \ldots X_n$ halmazok és $R \subseteq X_1 \times \cdots \times X_n$, ekkor R-et *n*-változós relációnak nevezzük.
- $R \subseteq X \times Y$ és $S \subseteq Y \times Z$ bináris relációk **kompozíciója** $R \circ S = \{(x, z) \subseteq X \times Z \mid \exists y \in Y : (x, y) \in R \wedge (y, z) \in S\}$
- $R^1 := R$, $R^i := R \circ R^{i-1}$ $(i \geqslant 2)$ definiálja R *i*-edik hatványát
- Ha $R \subseteq X \times X$, akkor $S := R \cup \{(x, x) \mid x \in X\}$ az R reláció **reflexív lezártja**
- Ha $R \subseteq X \times X$, akkor $R^+ := \bigcup_{i=1,2,\ldots} R^i$ az R reláció **tranzitív lezártja**
- Ha $R \subseteq X \times X$, akkor $R^* := R^+ \cup \{(x, x) \mid x \in X\}$ az R reláció **reflexív, tranzitív lezártja**

Reláció reflexív, tranzitív lezártja

Példa:

$X = \{1, 2, 3, 4, 5\}$, $R = \{(1, 2), (2, 3), (3, 4), (2, 5)\}$ ekkor

$S = \{(1, 2), (2, 3), (3, 4), (2, 5), (1, 1), (2, 2), (3, 3), (4, 4), (5, 5)\}$ az R reflexív lezártja

$R^1 = R = \{(1, 2), (2, 3), (3, 4), (2, 5)\}$

$R^2 = \{(1, 3), (2, 4), (1, 5)\}$

$R^3 = \{(1, 4)\}$

$R^4 = R^5 = \ldots = \varnothing$

$R^+ = \{(1, 2), (2, 3), (3, 4), (2, 5), (1, 3), (2, 4), (1, 5), (1, 4)\}$ az R tranzitív lezártja

$R^* = \{(1, 2), (2, 3), (3, 4), (2, 5), (1, 3), (2, 4), (1, 5), (1, 4), (1, 1), (2, 2), (3, 3), (4, 4), (5, 5)\}$ az R reflexív, tranzitív lezártja

Reláció reflexív, tranzitív lezártja

Észrevételek:

- A bináris relációk éppen az irányított gráfok (a reláció elemei megfeleltethetők az irányított éleknek).
- A bináris relációk reflexív, tranzitív lezártja egy olyan gráfnak felel meg, melyben u-ból v-be pontosan akkor van él, ha az eredeti gráfban van (véges) irányított út.
- $\Rightarrow_G$ egy bináris reláció a végtelen $X = (N \cup T)^*$ halmaz felett.
- A $\Rightarrow_G$ egylépéses levezetés reláció tehát természetes módon megad egy végtelen irányított gráfot az $(N \cup T)^*$ csúcshalmazon.
- A $\Rightarrow_G^*$ reflexív tranzitív lezárt gráfban u-ból v-be pontosan akkor van irányított él, ha véges sok egylépéses levezetésen át el lehet jutni u-ból v-be

Grammatikák

többlépéses levezetés

Definíció

Legyen $G = \langle N, T, P, S \rangle$ egy grammatika és legyen $u, v \in (N \cup T)^*$. u-ból (több lépésben)**levezethető** v, ha $(u, v) \in \Rightarrow_G^*$. ($\Rightarrow_G^*$ a $\Rightarrow_G$ reflexív tranzitív lezártja)

u-ból k **lépésben levezethető** v, ha $(u, v) \in \Rightarrow_G^k$. ($\Rightarrow_G^k$ a $\Rightarrow_G$ k-adik hatványa).

A kezdőszimbólumból levezethető szavakat **mondatformának** nevezzük.
Megadunk egy alternatív, közvetlen definíciót is:

Definíció

Legyen $G = \langle N, T, P, S \rangle$ egy grammatika és legyen $u, v \in (N \cup T)^*$. u-ból (több lépésben) **levezethető** v, ha $u = v$ vagy van olyan $n \geqslant 1$ és $w_0, \ldots w_n \in (N \cup T)^*$, hogy $w_{i-1} \Rightarrow_G w_i$ $(1 \leqslant i \leqslant n)$ és $w_0 = u$ és $w_n = v$. Jelölés: $u \Rightarrow_G^* v$.

Grammatikák

többlépéses levezetés

Ha egyértelmű melyik nyelvtanról van szó, akkor $\Rightarrow_G$ helyett röviden $\Rightarrow$-t írhatunk.

Példa:

$G_2 = \langle \{S, A, B, C\}, \{a, b, c\}, \{S \to a, S \to AB, A \to Ab, B \to \varepsilon, aCA \to aSc\}, S \rangle$.

Ekkor $BBaCAa \Rightarrow BBaScA$ és $BBaSca \Rightarrow BBaABca$, tehát $BBaCAa \Rightarrow^* BBaABca$.

$S \Rightarrow AB \Rightarrow AbB \Rightarrow AbbB$, tehát $AbbB$ egy mondatforma.

Generált nyelv

Definíció

Legyen $G = \langle N, T, P, S \rangle$ egy tetszőleges generatív grammatika. A G grammatika által **generált** $L(G)$ **nyelv** alatt az $L(G) = \{w \mid S \Rightarrow_G^* w, w \in T^*\}$ szavakból álló halmazt értjük.

Példák:

- $G = \langle N, T, P, S \rangle$, ahol $N = \{S, A, B\}$, $T = \{a, b\}$ és $P = \{S \to aSb, S \to ab, S \to ba\}$.

  Ekkor $L(G) = \{a^n abb^n, a^n bab^n \mid n \geqslant 0\}$.
- $G_1 = \langle N, T, P, S \rangle$, ahol $N = \{S, A, B\}$, $T = \{a, b\}$ és $P = \{S \to ASB, S \to \varepsilon, AB \to BA, BA \to AB, A \to a, B \to b\}$.

  Ekkor $L(G_1) = \{u \in \{a, b\}^* \mid u\text{-ban ugyanannyi } a \text{ van, mint } b\}$.

Ekvivalens grammatikák

Minden grammatika generál egy nyelvet, de ugyanazat a nyelvet több különböző grammatika is generálhatja.

Két **grammatika ekvivalens**, ha ugyanazt a nyelvet generálják.

Példa:
$G_2 = \langle N, T, P, S \rangle$, ahol $N = \{S, A, B\}$, $T = \{a, b\}$ és $P = \{S \to SS, S \to aSb, S \to bSa, S \to \varepsilon\}$.

Belátható, hogy
$L(G_2) = \{u \in \{a, b\}^* \mid u\text{-ban ugyanannyi } a \text{ van, mint } b\}$. Így ez a grammatika ekvivalens az előzővel.

Grammatikák Chomsky féle osztályozása

Legyen $G = \langle N, T, P, S \rangle$ egy generatív grammatika. A G generatív grammatika i-típusú ($i = 0, 1, 2, 3$), ha P szabályhalmazára teljesülnek a következők:

- $i = 0$: nincs korlátozás,
- $i = 1$: P minden szabálya $u_1 A u_2 \to u_1 v u_2$ alakú, ahol $u_1, u_2, v \in (N \cup T)^*$, $A \in N$, és $v \neq \varepsilon$, kivéve az $S \to \varepsilon$ alakú szabályt, feltéve, hogy P-ben ilyen szabály létezik. Ha P tartalmazza az $S \to \varepsilon$ szabályt, akkor S nem fordul elő P egyetlen szabályának jobb oldalán sem,
- $i = 2$: P minden szabálya $A \to v$ alakú, ahol $A \in N$ és $v \in (N \cup T)^*$,
- $i = 3$: P minden szabálya vagy $A \to uB$ vagy $A \to u$, alakú, ahol $A, B \in N$ és $u \in T^*$.

Grammatikaosztályba sorolás

Példa: Legyen $G = \langle N, T, P, S \rangle$, ahol $N = \{S, A, B\}$, $T = \{a, b\}$ és $P = \{S \to ASB, S \to \varepsilon, AB \to BA, BA \to AB, A \to a, B \to b\}$. Szabályról szabályra nézzük meg, hogy az adott szabály melyik típusba engedi sorolni a grammatikát.

$S \to ASB$	0, 1, 2
$S \to \varepsilon$	0, 1, 2, 3
$AB \to BA$	0
$BA \to AB$	0
$A \to a$	0, 1, 2, 3
$B \to b$	0, 1, 2, 3

Az első 2 szabály nem lehet együtt 1-típusú grammatikában.

A 3. és 4. szabály a 0-son kívül mindent kizár, így a grammatika csak a 0-típusba skatulyázható.

Nyelvek Chomsky féle osztályozása

Noam Chomsky (1928–) amerikai nyelvész, filozófus, politikai aktivista. A generatív nyelvtan elméletének megalkotója, kidolgozója a róla elnevezett Chomsky-hierarchiának. (Wikipédia)

Egy L nyelvet **i-típusúnak** mondunk, ahol $i = 0, 1, 2, 3$, ha i-típusú grammatikával generálható. $\mathcal{L}_i$ ($i = 0, 1, 2, 3$) jelöli az i-típusú nyelvek nyelvosztályát.

Példa: A korábbi 0-típusú G_1 és 2-típusú G_2 grammatikák egyaránt $L = \{u \in \{a, b\}^* \mid u\text{-ban ugyanannyi } a \text{ van, mint } b\}$-t generálják, így $L \in \mathcal{L}_2$.

A Chomsky féle grammatikaosztályok elnevezése

- A 0-típusú grammatikákat **mondatszerkezetű** (phrase-structure) grammatikáknak is nevezzük, ami nyelvészeti eredetükre utal.
- A 1-típusú grammatikák a **környezetfüggő** (context-sensitive) grammatikák, mert szabályai olyanok, hogy, egy A nemterminális valamely előfordulása egy v szóval csak u_1 és u_2 kontextus jelenlétében helyettesíthető.
- 2-típusú grammatikákat **környezetfüggetlen** (context-free) grammatikáknak mondjuk, mert szabályai olyanok, hogy az A nemterminális v-vel való helyettesítése bármely kontextusban megengedett.
- A 3-típusú grammatikákat **reguláris** (regular) vagy véges állapotú (finite state) grammatikáknak hívjuk, a véges állapotú automatákkal való kapcsolatuk miatt.

A 0,1,2,3-típusú nyelvek osztályait rendre **rekurzíven felsorolható**, **környezetfüggő**, **környezetfüggetlen**, valamint **reguláris** nyelvosztálynak is mondjuk.

A Chomsky-féle hierarchia

A grammatikák alakjából nyilvánvaló, hogy $\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_0$ és $\mathcal{L}_1 \subseteq \mathcal{L}_0$.

Tétel (Chomsky-féle nyelvhierarchia)

$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0.$$

Itt az $\mathcal{L}_2$ és az $\mathcal{L}_1$ nyelvosztályok közötti tartalmazási reláció nem látható azonnal a megfelelő grammatikák definíciójából.

Szintén $\mathcal{L}_1$-et tudják generálni a **hossznemcsökkentő** grammatikák. Ezek $p \to q$ szabályaira $|p| \leqslant |q|$ teljesül kivéve az $S \to \varepsilon$ alakú szabályt, feltéve, hogy P-ben ilyen szabály létezik. Ha P tartalmazza az $S \to \varepsilon$ szabályt, akkor S nem fordul elő P egyetlen szabályának jobb oldalán sem.

Grammatikák a gyakorlatban

BNF (Backus-Naur Form) **John Backus, Peter Naur ALGOL 60**

A BNF egy széles körben használt metanyelv melynek segítségével szabályok alkothatók meg (például egy programozási nyelv szintaktikai szabályai).Építőkövei:

- $\langle$név$\rangle$, **fogalmak** (vagy más néven **nemterminálisok**)
- ::=, a szabályok bal- és jobboldalának elválasztászára
- a sztringek alkotóelemei (**terminálisok**)

Egy szabály bal- és jobboldalból áll, köztük ::=, baloldalon pontosan 1 fogalom, jobboldalon terminálisok és fogalmak véges sorozata állhat.

A BNF megfelel a környezetfüggetlen grammatikának. Példa:

$\langle$azonosító$\rangle$::= $\langle$betű$\rangle \mid \langle$betű$\rangle\langle$azvég$\rangle$
$\langle$azvég$\rangle$::= $\langle$betű$\rangle \mid \langle$számjegy$\rangle \mid \langle$betű$\rangle\langle$azvég$\rangle \mid \langle$számjegy$\rangle\langle$azvég$\rangle$
$\langle$betű$\rangle$::=$a \mid \cdots \mid z$
$\langle$számjegy$\rangle$::=$0 \mid \cdots \mid 9$

Kontrollált környezetfüggetlen grammatikák

BNF-fel nem írható le teljeskörűen a programozási nyelvek szintaxisa (lásd pl. deklaráció fogalma, az $\{uu \mid u \in V^*\}$ nyelv nem környezetfüggetlen).

Módosítható-e úgy a környezetfüggetlen grammatikák modellje, hogy a természetes, egyszerű szabályalakot megtartsuk, de a modell kifejező ereje nőjön?

Tudnánk-e nem környezetfüggetlen nyelveket generálni környezetfüggetlen grammatikákkal azáltal, hogy például a produkciós szabályok alkalmazhatóságára feltételeket szabunk.

A válasz igen, nézzünk erre néhány példát!

3 példát nézünk, a **programozott grammatikák** és a **mátrixgrammatikák** esetén az alkalmazott szabályok sorrendiségére vonatkozik a kontroll, míg a **véletlen szövegfeltételekkel adott grammatikák** esetében a szabályok alkalmazhatósága az aktuális mondatforma tulajdonságaitól függ.

Progamozott grammatikák

Környezetfüggetlen programozott grammatikán egy $G = \langle N, T, P, S\rangle$ négyest értünk, ahol

- N és T diszjunkt véges ábécék,
- $S \in N$ a kezdőszimbólum
- P rendezett hármasok véges halmaza, melyek $r = (p, \sigma, \phi)$ alakúak, ahol p egy környezetfüggetlen szabály, $\sigma, \phi \subseteq P$.

σ-t az r szabály **siker-halmazának**, ϕ a szabály **kudarc-halmazának** nevezzük.

Programozott grammatikák

Jelölje Q a **megengedett** szabályok halmazát. Kezdetben $Q := P$.

Egy $u \in (N \cup T)^*$ mondatformából **egy lépésben levezethető** a $v \in (N \cup T)^*$ mondatforma, ha van olyan $r = (A \to w, \sigma, \phi) \in Q$, melyre a következők egyike teljesül

- $u = xAy$ és $v = xwy$ valamely $x, y \in (N \cup T)^*$-ra, ekkor a szabályalkalmazás **sikeres** és $Q := \sigma$,
- u nem tartalmazza A-t és $v = u$, ekkor a szabályalkalmazás **sikertelen** és $Q := \phi$.

Tehát adott mindig megengedett szabályok egy $Q \subseteq P$ részhalmaza. Választunk egy $r \in Q$ szabályt. Amennyiben sikerül a kiválasztott szabályt alkalmazni, akkor az adott szabály siker-halmazából, ha nem járunk sikerrel, akkor pedig a kudarc-halmazából kell a következő szabályt választani.

Az így kapott $\Rightarrow$ reláció reflexív, tranzitív lezártjaként definiáljuk $\Rightarrow^*$-t. $L(G) := \{w \in T^* \mid S \Rightarrow^* w\}$.

Programozott grammatikák

Ha $r = (p, \sigma, \varnothing)$ minden $r \in P$ szabályra fennáll, akkor a programozott grammatika **előfordulásellenőrzés nélküli**, egyébként **előfordulásellenőrzéses**.

Példa: Legyen $G = \langle \{S, A\}, \{a\}, S, \{r_1, r_2, r_3\} \rangle$, ahol
$r_1 = (S \to AA, \{r_1\}, \{r_2\})$
$r_2 = (A \to S, \{r_2\}, \{r_1, r_3\})$
$r_3 = (S \to a, \{r_3\}, \varnothing)$

$S \overset{r_1}{\Rightarrow} AA$ (siker:$\{r_1\}$) $\overset{r_1}{\Rightarrow} AA$ (kudarc:$\{r_2\}$) $\overset{r_2}{\Rightarrow} SA$ (siker:$\{r_2\}$)
$\overset{r_2}{\Rightarrow} SS$ (siker:$\{r_2\}$) $\overset{r_2}{\Rightarrow} SS$ (kudarc:$\{r_1, r_3\}$) $\overset{r_3}{\Rightarrow} aS$ (siker:$\{r_3\}$)
$\overset{r_3}{\Rightarrow} aa$.

Az SS-en kudarccal végződött $A \to S$ átírás után választhattuk volna az r_1 szabályt is, ez újra megduplázta volna az S-ek számát.

G egy előfordulásellenőrzéses programozott grammatika és $L(G) = \{a^{2^n} \mid n \geqslant 0\}$.

Mátrixgrammatikák

Előfordulásellenőrzéses környezetfüggetlen mátrixgrammatikán egy $G = \langle N, T, M, S, F \rangle$ ötöst értünk, ahol

- N és T diszjunkt véges ábécék,
- $S \in N$ a kezdőszimbólum,
- $M = \{m_1, m_2, \ldots, m_n\}$, $n \geqslant 1$, sorozatok véges halmaza
- $m_i = (p_{i_1}, \ldots p_{i_{k(i)}}), k(i) \geqslant 1, 1 \leqslant i \leqslant n$, ahol minden $p_{i_j}, 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant k(i)$, egy környezetfüggetlen szabály, és
- F M-beli sorozatok szabályainak egy részhalmaza, azaz $F \subseteq \{p_{i_j} \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant k(i)\}$.

M elemeit **mátrixoknak** nevezzük.

Ha $F = \varnothing$, akkor a mátrixgrammatikát **előfordulásellenőrzés nélkülinek** nevezzük.

Mátrixgrammatikák

Legyen $G = \langle N, T, M, S, F \rangle$ egy tetszőleges mátrixgrammatika. Ekkor $u \in (N \cup T)^*$-ból **egy lépésben levezethető** a $v \in (N \cup T)^*$ szó az $m_i = (A_{i_1} \to v_{i_1}, \ldots A_{i_{k(i)}} \to v_{i_{k(i)}}) \in M$ mátrix szerint, ha léteznek $w_1, \ldots, w_{k(i)+1} \in (N \cup T)^*$ szavak a következő feltételekkel:

- $w_1 = u$ és $w_{k(i)+1} = v$
- ha $w_j = x_j A_{i_j} y_j$, valamely $x_j, y_j \in (N \cup T)^*$-ra, akkor $w_{j+1} = x_j v_{i_j} y_j$ $(1 \leqslant j \leqslant k(i))$
- ha A_{i_j} nem fordul elő w_j-ben akkor $w_{j+1} = w_j$ és $A_{i_j} \to v_{i_j} \in F$ $(1 \leqslant j \leqslant k(i))$

Jelölés: $\Rightarrow_{m_i}$ vagy röviden $\Rightarrow$,

$\Rightarrow^*$ (többlépéses levezetés) és $L(G)$ (generált nyelv) definíciója a szokásos.

Mátrixgrammatikák

1. **Példa:** $u = XYAX$, $v = XYCX$.
   - F bármi, $m = (A \to B, B \to C)$, ekkor $u \Rightarrow v$,
   - $F = \varnothing$, $m = (A \to B, E \to F, B \to C)$, ekkor $u \not\Rightarrow v$,
   - $F \supseteq \{E \to F\}$, $m = (A \to B, E \to F, B \to C)$, ekkor $u \Rightarrow v$.
2. **Példa:**

Legyen $G = \langle N, T, M, S, \varnothing \rangle$ egy mátrixgrammatika előfordulásellenőrzés nélkül, ahol $N = \{S, A, B\}$, $T = \{a, b\}$, és $M = \{m_1, m_2, m_3, m_4, m_5\}$, ahol

$m_1 = (S \to AB)$,
$m_2 = (A \to bA, B \to bB)$, $\quad m_4 = (A \to aA, B \to aB)$,
$m_3 = (A \to b, B \to b)$, $\quad m_5 = (A \to a, B \to a)$.

$S \overset{m_1}{\Longrightarrow} AB \overset{m_4}{\Longrightarrow} aAaB \overset{m_4}{\Longrightarrow} aaAaaB \overset{m_2}{\Longrightarrow} aabAaabB \overset{m_4}{\Longrightarrow} aabaAaabaB \overset{m_3}{\Longrightarrow} aababaabab$.

Könnyen látható, hogy $L(G) = \{ww \mid w \in \{a, b\}^+\}$.

Véletlen szövegfeltételekkel adott grammatika

Környezetfüggetlen véletlen szövegfeltételekkel adott grammatikán (random context grammar) egy $G = \langle N, T, P, S \rangle$ négyest értünk, ahol

- N és T diszjunkt véges ábécék,
- $S \in N$ a kezdőszimbólum (axióma),
- P rendezett hármasok véges halmaza, amelyek (p, Q, R), alakúak, ahol p egy környezetfüggetlen szabály, $Q, R \subseteq N$.

elnevezés: Q-t az (p, Q, R) hármas **megengedő** (permitting), R-et pedig **tiltó** (forbidding) kontextusának nevezzük.

Véletlen szövegfeltételekkel adott grammatika

Egy környezetfüggetlen véletlen szövegfeltételekkel adott grammatika **előfordulásellenőrzés nélküli**, ha minden $(p, Q, R) \in P$ esetén $R = \varnothing$. **Előfordulásellenőrzéses**: amennyiben $R \neq \varnothing$ is megengedett.

Más elnevezés az előfordulásellenőrzés nélküli esetre: környezetfüggetlen megengedő véletlen szövegfeltételekkel adott grammatika (permitting random context grammar, pRC grammar).

Hasonlóan definálható a környezetfüggetlen tiltó véletlen szövegfeltételekkel adott grammatika (forbidding random context grammar, fRC grammar).

Véletlen szövegfeltételekkel adott grammatika

Legyen $G = \langle N, T, P, S \rangle$ egy tetszőleges véletlen szövegfeltételekkel adott grammatika. Ekkor az v szó **egy lépésben levezethető** u-ból ($u, v \in (N \cup T)^*$), ha létezik $(A \to w, Q, R) \in P$ $x, y \in (N \cup T)^*$, hogy

- $u = xAy$ és $v = xwy$
- minden Q-beli nemterminális előfordul xy-ban
- semelyik R-beli nemterminális sem fordul elő xy-ban

Jelölés: $\Rightarrow$. $\Rightarrow^*$ (többlépéses levezetés) és $L(G)$ (generált nyelv) definíciója a szokásos.

Példa:

- A $BABaC$ mondatformára az $(A \to XY, \{C\}, \{A\})$ szabály alkalmazható, mivel C van, A viszont nincs a $BBaC$ környezetben. $BABaC \Rightarrow BXYBaC$.
- A $BABaC$ mondatformára az $(A \to XY, \{A, B\}, \{X\})$ szabály nem alkalmazható, mert nincs az átírandón felül további A a szóban, azaz a $BBaC$ környezetben.

Véletlen szövegfeltételekkel adott grammatika

Példa: Legyen $G = \langle N, T, P, S \rangle$ egy véletlen szövegfeltételekkel adott grammatika, ahol $N = \{S, X, Y, A\}$, $T = \{a\}$, és $P = \{r_1, r_2, r_3, r_4, r_5\}$, ahol

$r_1 = (S \to XX, \varnothing, \{Y, A\})$,
$r_2 = (X \to Y, \varnothing, \{S\})$,
$r_3 = (Y \to S, \varnothing, \{X\})$,
$r_4 = (S \to A, \varnothing, \{X, Y\})$,
$r_5 = (A \to a, \varnothing, \{S\})$.

$S \Rightarrow XX \Rightarrow XY \Rightarrow YY \Rightarrow YS \Rightarrow SS \Rightarrow SXX \Rightarrow XXXX \Rightarrow XYXX \Rightarrow XYYX \Rightarrow XYYY \Rightarrow YYYY \Rightarrow YYSY \Rightarrow SYSY \Rightarrow SSSY \Rightarrow SSSS \Rightarrow SSAS \Rightarrow SSAA \Rightarrow ASAA \Rightarrow AAAA \Rightarrow AAAa \Rightarrow aAAa \Rightarrow aaAa \Rightarrow aaaa.$

Végig kényszerpályán a levezetés, egyedül csupa S-nél van opció, ez mindig 2-hatványnál fordul elő.

Tehát $L(G) = \{a^{2^n} \mid n \geqslant 0\}$.

Nyelvcsaládok, kifejező erő

$\mathcal{L}(\mathrm{PR}_{\mathrm{ac}})$, illetve $\mathcal{L}(\mathrm{PR}^{\varepsilon}_{\mathrm{ac}})$ jelöli rendre az előfordulásellenőrzéses ε-mentes környezetfüggetlen szabályokból, valamint a tetszőleges környezetfüggetlen szabályokból álló programozott grammatikák által generált nyelvek osztályát.

Ha a grammatika előfordulásellenőrzés nélküli, akkor az ac index hiányzik.

Hasonlóképpen: $\mathcal{L}(\mathrm{MAT}_{\mathrm{ac}})$, $\mathcal{L}(\mathrm{MAT}^{\varepsilon}_{\mathrm{ac}})$, $\mathcal{L}(\mathrm{RC}_{\mathrm{ac}})$, illetve $\mathcal{L}(\mathrm{RC}^{\varepsilon}_{\mathrm{ac}})$.

Tétel

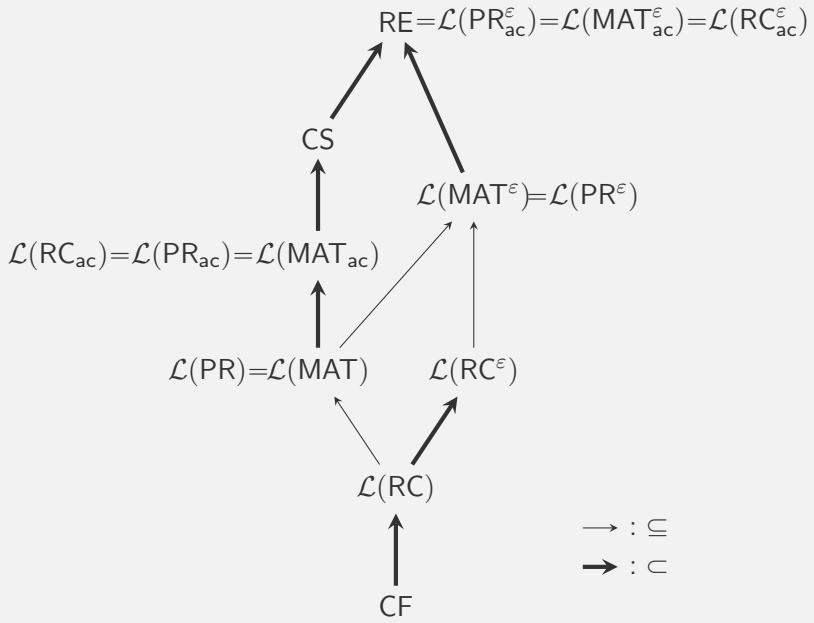
Fennállnak a következők

$$\mathcal{L}_2 \subset \mathcal{L}(\mathrm{PR}_{\mathrm{ac}}) = \mathcal{L}(\mathrm{MAT}_{\mathrm{ac}}) = \mathcal{L}(\mathrm{RC}_{\mathrm{ac}}) \subset \mathcal{L}_1$$

és

$$\mathcal{L}(\mathrm{PR}^{\varepsilon}_{\mathrm{ac}}) = \mathcal{L}(\mathrm{MAT}^{\varepsilon}_{\mathrm{ac}}) = \mathcal{L}(\mathrm{RC}^{\varepsilon}_{\mathrm{ac}}) = \mathcal{L}_0.$$

Nyelvcsaládok, kifejező erő



Számítási modellek

2. előadás

Reguláris kifejezések

Motiváció

Mintaillesztés reguláris kifejezések alapján:

- Vírusbejegyzések pásztázása.
- Természetes nyelvfeldolgozás.
- Strukturális szövegkiemelés szövegszerkesztőkben.
- Programnyelvek specifikációja.
- Információ elérése digitális könyvtárakban.
- Genomika. (genom, mint karakterlánc)
- Szöveg szűrése (spam, malware).
- Adatbeviteli mezők (dátum, e-mail, URL, hitelkártya) érvényesítése.

Reguláris kifejezések

Definíció

Legyenek V és $V' = \{\varepsilon, \cdot, +, *, (,)\}$ diszjunkt ábécék. A V ábécé feletti **reguláris kifejezéseket** rekurzív módon a következőképpen definiáljuk:

1. $\varnothing$ és ε reguláris kifejezés V felett,
2. minden $a \in V$ reguláris kifejezés V felett,
3. Ha R reguláris kifejezés V felett, akkor R^* is reguláris kifejezés V felett,
4. Ha Q és R reguláris kifejezések V felett, akkor $(Q \cdot R)$ és $(Q + R)$ is reguláris kifejezések V felett

Reguláris kifejezések

Minden egyes reguláris kifejezés egy-egy reguláris nyelvet reprezentál, melyet az alábbi rekurzióval definiálunk.

Definíció

1. $\varnothing$ az $\varnothing$ nyelvet reprezentálja,
2. ε az $\{\varepsilon\}$ nyelvet reprezentálja,
3. a $(\in V)$ az $\{a\}$ nyelvet reprezentálja,
4. ha R az L nyelvet reprezentálja, akkor R^* az L^* nyelvet reprezentálja,
5. ha R_1 az L_1 és R_2 az L_2 nyelvet reprezentálja, akkor $(Q \cdot R)$ az $L_1 L_2$ nyelvet, míg $(Q + R)$ az $L_1 \cup L_2$ nyelvet reprezentálja.

Azaz a $*, \cdot$ és $+$ szimbólumok jelölik rendre a lezárt, a konkatenáció és az unió műveleteket.

Reguláris kifejezések

Legyen P, Q és R egy-egy reguláris kifejezés. Ekkor fennállnak a következő egyenlőségek:

1. $P + (Q + R) = (P + Q) + R$
2. $P \cdot (Q \cdot R) = (P \cdot Q) \cdot R$
3. $P + Q = Q + P$
4. $P \cdot (Q + R) = P \cdot Q + P \cdot R$
5. $(P + Q) \cdot R = P \cdot R + Q \cdot R$
6. $P^* = \varepsilon + P \cdot P^*$
7. $\varepsilon \cdot P = P \cdot \varepsilon = P$
8. $P^* = (\varepsilon + P)^*$

A műveletek precedenciasorrendje $*, \cdot, +$, ennek és az asszociatív szabályok figyelembevételével bizonyos zárójelpárok elhagyhatók.

A reguláris kifejezések kifejező ereje

Példa:

Az $(a + b)a^*$ és $aa^* + ba^*$ reguláris kifejezések által reprezentált nyelv ugyanaz. $\{aa^n \mid n \in \mathbb{N}\} \cup \{ba^n \mid n \in \mathbb{N}\}$.

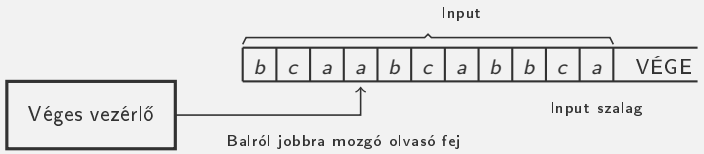
Másrészt az $a + ba^*$ által reprezentált nyelv $\{a, b, ba, ba^2, ba^3, \ldots\}$.

Tétel

- Minden reguláris kifejezés egy reguláris (3-típusú) nyelvet reprezentál.
- Mnden reguláris (3-típusú) nyelvhez megadható egy, ezen nyelvet reprezentáló reguláris kifejezés.

Megjegyzés: Mivel $\mathcal{L}_3$ nem csak a reguláris műveletekre (Kleene-lezárt, konkatenáció unió), hanem a metszet, különbség, komplementer műveletekre is zárt ($\mathcal{L}_2, \mathcal{L}_1, \mathcal{L}_0$ csak a reguláris műveletekre) ezért a gyakorlatban gyakran ezen műveletekkel kiterjesztett reguláris kifejezéseket használnak.

Véges automata



- Formális nyelvek azonosítása nemcsak generatív eszközökkel, hanem felismerő eszközökkel is lehetséges. Ilyenek az automaták, amelyek szavak feldolgozására és azonosítására alkalmasak.
- A grammatikák szintetizáló, míg az automaták analitikus megközelítést alkalmaznak.
- Az automata egy szó feldolgozása után kétféleképpen viselkedhet, vagy elfogadja (igen), vagy elutasítja (nem).

Véges automata

- A véges automata diszkrét időintervallumokban végrehajtott lépések sorozatán keresztül működik.
- A véges automata a kezdőállapotból vagy kezdeti állapotból indul, az inputszó az inputszalagon helyezkedik el, az olvasófej pedig az inputszó legbaloldalibb szimbólumán áll.
- Az automata, miután elolvasott egy szimbólumot, az olvasófejet egy pozícióval jobbra mozgatja, majd állapotot vált az állapot-átmenet függvény szerint.
- Ha az automata elolvasta az inputot, megáll (felismeri vagy elutasítja a szót).

Véges automata

Alkalmazási területek:

- Önműködő ajtó
- Kávéautomata
- Mintafelismerés
- Matematikai rejtvények (pl. átkelés a folyón)
- Mintafelismerés Markov-láncokban.
- Beszédfeldolgozás, optikai karakterfelismerés.
- Piaci részesedések eloszlásának előrejelzése.

Véges automata

Jelölés: ha X egy halmaz, $\mathcal{P}(X)$ jelöli X **hatványhalmazát**, azaz X részhalmazainak halmazát.

Definíció

A **véges automata** egy rendezett ötös, $A = \langle Q, T, \delta, Q_0, F \rangle$, ahol

- Q az állapotok egy véges, nemüres halmaza,
- T az inputszimbólumok véges ábécéje,
- $\delta : \ Q \times T \to \mathcal{P}(Q)$ ún. állapot-átmenet függvény,
- $Q_0 \subseteq Q$ a kezdőállapotok halmaza,
- $F \subseteq Q$ az elfogadó állapotok halmaza.

Definíció

Ha $\forall (q, a) \in Q \times T$ esetén $|\delta(q, a)| = 1$ és $|Q_0| = 1$, akkor **determinisztikus véges automatáról** beszélünk.

Ilyenkor δ egy $Q \times T \to Q$ (totális) függvénynek tekinthető, a kezdőállapot pedig $q_0 \in Q$.

Determinisztikus és nemdeterminisztikus véges automata

Determinisztikus véges automata: (VDA) A δ függvény egyértékű. Ez azt jelenti, hogy minden (q, a) párra, ahol $(q, a) \in Q \times T$ pontosan egy olyan s állapot létezik, amelyre $\delta(q, a) = s$ fennáll. Egyetlen $q_0 \in Q$ kezdőállapot van.

Nemdeterminisztikus véges automata: (VNDA) Többértékű állapot-átmenet függvény is megengedett, azaz δ definiálható úgy, mint egy $Q \times T \to \mathcal{P}(Q)$ leképezés. Ekkor nemdeterminisztikus véges automatáról beszélünk. Több kezdőállapot is megengedett (a kezdőállapotok $Q_0 \subseteq Q$ halmaza). Előfordulhat, hogy az $\delta(q, a) = \varnothing$ valamely (q, a)-ra, azaz elakad a gép. Több kezdőállapot lehet, $Q_0 \subseteq Q$ a kezdőállapotok halmaza.

Véges automata

Alternatív jelölés: Az állapot-átmeneteket $qa \to p$ alakú szabályok formájában is megadhatjuk $p \in \delta(q, a)$ esetén.

Jelöljük M_δ -val az $A = \langle Q, T, \delta, Q_0, F \rangle$ nemdeterminisztikus véges automata δ állapot-átmenet függvénye által az előbbi módon származó szabályok halmazát.

Ekkor, ha minden egyes (q, a) párra M_δ pontosan egy $qa \to p$ szabályt tartalmaz, akkor a véges automata determinisztikus, egyébként nemdeterminisztikus.

Véges automaták – egy- és többlépéses redukció

Definíció

Legyen $A = \langle Q, T, \delta, Q_0, F \rangle$ egy véges automata és legyenek $u, v \in QT^*$ szavak. Az A automata az u szót **egy lépésben** (közvetlenül) a v szóra **redukálja** (jelölés: $u \Rightarrow_A v$), ha van olyan $qa \to p \in M_\delta$ szabály (vagyis $p \in \delta(q, a)$) és olyan $w \in T^*$ szó, hogy $u = qaw$ és $v = pw$ teljesül.

Példa: Ha $\delta(q, a) = \{r, s\}$, akkor $qabbab \Rightarrow sbbab$. (VNDA)

Definíció

A $\Rightarrow_A^* \subseteq QT^* \times QT^*$ reláció a $\Rightarrow_A$ reláció reflexív, tranzitív lezártja. Ha $u \Rightarrow_A^* v$, akkor azt mondjuk hogy A az u szót több lépésben (közvetetten) a v szóra **redukálja**.

Példa: Ha $\delta(q, a) = \{r, s\}$ és $\delta(s, b) = \{q, r\}$ akkor $qabbab \Rightarrow sbbab \Rightarrow rbab$ és így $qabbab \Rightarrow^* rbab$. (VNDA)

Véges automata

Példa: $T = \{a, b, c\}$. Adjunk VDA-t mely a legfeljebb 5 hosszú szavakat fogadja el!

Megoldás: I. (Képlettel)

$\langle \{q_0, \ldots, q_6\}, \{a, b, c\}, \delta, q_0, \{q_0, \ldots, q_5\} \rangle$

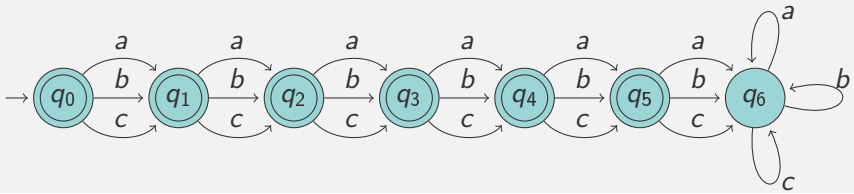
$\delta(q_i, t) = q_{i+1}$, $(i{=}0, \ldots, 5,\ t \in \{a, b, c\})$

$\delta(q_6, t) = q_6$. $(t \in \{a, b, c\})$

II. (Táblázattal)

	a	b	c
$\leftrightarrows q_0$	q_1	q_1	q_1
$\leftarrow q_1$	q_2	q_2	q_2
$\leftarrow q_2$	q_3	q_3	q_3
$\leftarrow q_3$	q_4	q_4	q_4
$\leftarrow q_4$	q_5	q_5	q_5
$\leftarrow q_5$	q_6	q_6	q_6
q_6	q_6	q_6	q_6

III. (Átmenetdiagrammal)



Véges automaták által felismert nyelv

Definíció

Az $A = \langle Q, T, \delta, Q_0, F \rangle$ nemdeterminisztikus véges automata által **elfogadott/felismert nyelv**:
$L(A) = \{u \in T^* \mid q_0 u \Rightarrow_A^* p$ valamely $q_0 \in Q_0$-ra és $p \in F$-re$\}$

Megjegyzés: Determinisztikus esetben $Q_0 = \{q_0\}$ egyelemű, és minden $u \in T^*$-ra $q_0 u$ legfeljebb egyféleképp redukálható valamely $p \in F$-re.

Tehát a determinisztikus esetben az felismert nyelv definíciója így egyszerűsödik:

Definíció

Az $A = \langle Q, T, \delta, q_0, F \rangle$ determinisztikus véges automata által **felismert nyelv**: $L(A) = \{u \in T^* \mid q_0 u \Rightarrow_A^* p,$ ahol $p \in F\}$

Véges automaták determinizálása

Tétel

Minden $A = \langle Q, T, \delta, Q_0, F \rangle$ nemdeterminisztikus véges automatához megkonstruálható egy $A' = \langle Q', T, \delta', q_0', F' \rangle$ determinisztikus véges automata úgy, hogy $L(A) = L(A')$ teljesül.

A konstrukció:

$$Q' := \mathcal{P}(Q), \quad q_0' := Q_0, \quad F' := \{q' \in Q' \mid q' \cap F \neq \emptyset\},$$

$$\delta'(q', a) := \bigcup_{q \in q'} \delta(q, a).$$

Véges automaták determinizálása – Példa

	a	b
$\to q_0$	$\{\}$	$\{q_1, q_2\}$
$\leftrightarrows q_1$	$\{q_0\}$	$\{\}$
$\leftarrow q_2$	$\{q_1\}$	$\{q_2\}$

VNDA

		a	b
	$\{\}$	$\{\}$	$\{\}$
	$\{q_0\}$	$\{\}$	$\{q_1, q_2\}$
$\leftarrow$	$\{q_1\}$	$\{q_0\}$	$\{\}$
$\leftarrow$	$\{q_2\}$	$\{q_1\}$	$\{q_2\}$
$\leftrightarrows$	$\{q_0, q_1\}$	$\{q_0\}$	$\{q_1, q_2\}$
$\leftarrow$	$\{q_0, q_2\}$	$\{q_1\}$	$\{q_1, q_2\}$
$\leftarrow$	$\{q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_2\}$
$\leftarrow$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_1, q_2\}$

VDA

Megjegyzés: Q' q_0'-ből elérhetetlen állapotai elhagyhatóak.

A véges automaták számítási ereje

Tétel

- Minden A nemdeterminisztikus véges automatához meg tudunk adni egy 3-típusú G grammatikát úgy, hogy $L(G) = L(A)$ teljesül.
- Minden 3-típusú G grammatikához meg tudunk adni egy A véges automatát úgy, hogy $L(A) = L(G)$ teljesül.

Következmény

- Minden reguláris kifejezéshez van olyan véges automata, amelyik a reguláris kifejezés által reprezentált nyelvet ismeri fel.
- A véges automaták által felismert nyelvek reprezentálhatóak reguláris kifejezéssel.

Vagyis a VDA-k, VNDA-k, reguláris kifejezések számítási ereje megegyezik a reguláris grammatikáéval, éppen az $\mathcal{L}_3$-beli nyelveket tudjuk megadni velük.

Véges automata – Myhill-Nerode tétel

Definíció

Legyen L egy T ábécé feletti nyelv. Az L nyelv által indukált E_L reláció alatt egy olyan bináris relációt értünk a T^*-on, amelyre teljesül, hogy bármely $u, v \in T^*$-ra uE_Lv akkor és csak akkor, ha nincs olyan $w \in T^*$, hogy az uw és vw szavak közül pontosan az egyik eleme L-nek.

E_L ekvivalenciareláció és jobb-invariáns. (Jobb-invariáns: ha uE_Lv, akkor uwE_Lvw is fennáll minden $w \in T^*$ szóra.)

Az E_L reláció indexén ekvivalenciaosztályainak számát értjük.

Tétel(Myhill-Nerode)

$L \subseteq T^*$ akkor és csak akkor ismerhető fel determinisztikus véges automatával, ha E_L véges indexű.

Minimális (állapotszámú) véges automata

Definíció

Az A determinisztikus véges automata minimális állapotszámú (minimális), ha nincs olyan A' determinisztikus véges automata, amely ugyanazt a nyelvet ismeri fel, mint A, de A' állapotainak száma kisebb, mint A állapotainak száma.

Tétel

Az L reguláris nyelvet elfogadó minimális (állapotszámú) determinisztikus véges automata az izomorfizmus erejéig egyértelmű.

Megjegyzés: $\forall$ L reguláris nyelvhez Myhill-Nerode tétel alapján készíthető egy $A_{\mathrm{MN}} = \langle Q, T, \delta, q_0, F \rangle$ minimális automata.

Legyenek $w_0, \ldots, w_{n-1}$ E_L ekvivalenciaosztályainak 1-1 reprezentánsa, ahol $w_0 = \varepsilon$. $Q := \{q_i \mid 0 \leqslant i \leqslant n-1\}$, $\delta(q_i, t) := q_j$, akkor és csak akkor, ha $w_i t E_L w_j$. $F := \{q_i \mid w_i \in L\}$.

Kis Bar-Hillel lemma

A Myhill-Nerode tétel szükséges és elégséges feltételt ad egy nyelv $\mathcal{L}_3$-ba tartozására.

A Kis Bar-Hillel lemma szükséges feltételt ad egy nyelv $\mathcal{L}_3$-ba tartozására, így arra használható, hogy egy nyelvről bizonyítsuk, hogy nem reguláris: ha egy L nyelvre nem teljesül a Kis Bar-Hillel lemma feltétele, akkor $L \notin \mathcal{L}_3$.

Tétel (Kis Bar-Hillel lemma)

Minden $L \in \mathcal{L}_3$ nyelvhez van olyan $n \in \mathbb{N}$ konstans, hogy minden $w \in L$ szó esetén ha tekintjünk egy tetszőleges olyan $w = uw'v$ felbontását, ahol $|w'| \geqslant n$, akkor van w'-nek olyan y részszava ($w' = xyz$), hogy $0 < |y| \leqslant n$, és minden $i \geqslant 0$ esetén $uxy^izv \in L$.

Megjegyzés A tétel gyakran használt másik elnevezése: "pumpálási lemma"

Sztochasztikus automata

- A **sztochasztikus automata** a nemdeterminisztikus véges automata olyan általánosítása, ahol mind a kezdőállapotot, mind pedig egy adott (átmenet, betű) párra az új állapotot egy valószínűségi eloszlás alapján véletlenszerűen választjuk.
- Jelölje $s_1, \ldots, s_n$ a sztochasztikus automata állapotait. Ekkor x inputszimbólum hatására az automata s állapotból valamely s_i állapotba megy, $p_i(s, x)$ valószínűséggel, ahol minden s-re és x-re fennáll a következő:

$$\sum_{i=1}^{n} p_i(s, x) = 1, \quad p_i(s, x) \geqslant 0$$

- A kezdőállapot helyett definiáljuk a **kezdőállapotok eloszlását**, azaz minden állapot kezdőállapot valamilyen rögzített valószínűséggel.
- Az **elfogadott nyelv** $L(PA, S_1, \eta)$ függ a végállapotok S_1 halmazától és a $0 \leqslant \eta < 1$ valós számtól, az ún. **vágási ponttól**.

Sztochasztikus automata

- Az elfogadott nyelv $L(PA, S_1, \eta)$ az összes olyan szót tartalmazza, amely által az automata állapot-átmenetek sorozatán keresztül valamely S_1 -beli állapotba jut, ahol a valószínűség nagyobb, mint η.
- n-dimenziós **sztochasztikus mátrix** alatt egy $(p_{ij})_{1 \leqslant i,j \leqslant n}$ négyzetes mátrixot értünk, melyre (1) $p_{ij} \geqslant 0$ $(1 \leqslant i, j \leqslant n)$ (2) $\sum_{j=1}^{n} p_{ij} = 1$ $(1 \leqslant i \leqslant n)$.
- n-dimenziós **sztochasztikus sorvektornak** (oszlopvektornak) egy olyan n-dimenziós sorvektort (oszlopvektort) nevezünk, amelynek komponensei nemnegatívak és a komponensek összege 1.
- Ha a sztochasztikus sorvektornak csak egy komponense 1, akkor **koordinátavektorról** beszélünk.
- Az n-dimenziós E_n egységmátrix sztochasztikus mátrix.

Sztochasztikus automata

Definíció

A **véges sztochasztikus automata** egy V ábécé felett egy $PA = \langle S, s_0, M \rangle$ rendezett hármas, ahol

- $S = \{s_1, \ldots, s_n\}$ az állapotok egy véges, nemüres halmaza,
- s_0 egy n-dimenziós sztochasztikus sorvektor, a kezdeti állapotok eloszlása
- M egy leképezés, amely V -t leképezi az n-dimenziós sztochasztikus mátrixok halmazába.

Valamely $x \in V$ -re az $M(x)$ mátrix (i, j)-dik eleme $p_j(s_i, x)$, annak a valószínűsége, hogy az x szimbólum hatására PA az s_i állapotból az s_j állapotba lép.

Példa Tekintsük a következő sztochasztikus automatát:

$PA_1 = \langle \{s_1, s_2\}, (1, 0), M \rangle$ az $\{x, y\}$ ábécé felett, ahol

$$M(x) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad M(y) = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

Sztochasztikus automata

Legyen $PA = \langle S, s_0, M \rangle$ egy V ábécé feletti véges sztochasztikus automata. Ekkor az M függvény V -ről a következőképpen terjeszthető ki V^*-ra:

- $\hat{M}(\varepsilon) := E_n$
- $\hat{M}(x_1 \cdots x_n) := M(x_1)M(x_2)\cdots M(x_n)$, ahol $k \geqslant 2, x_i \in V$.

$\hat{M}$ helyett a továbbiakban M-et írunk.

Valamely $w \in V^*$ szóra az $M(w)$ mátrix (i,j)-edik elemét $p_j(s_i, w)$ jelöli, amely annak a valószínűsége, hogy az automata a w inputszó feldolgozása után az s_i állapotból éppen az s_j állapotba jut.

Sztochasztikus automata

Legyen $PA = \langle S, s_0, M \rangle$ egy V ábécé feletti véges sztochasztikus automata és legyen $w \in V^*$. Az $s_0 M(w)$ sztochasztikus sorvektor a w **eredményeként kapott állapoteloszlás**, melyet $PA(w)$-vel jelölünk.

Észrevétel: $PA(\varepsilon) = s_0$.

Definíció

Legyen $PA = \langle S, s_0, M \rangle$ egy V ábécé feletti véges sztochasztikus automata, $0 \leqslant \eta < 1$ egy valós szám, és $\bar{s}_1$ egy n-dimenziós oszlopvektor, amelynek minden komponense vagy 0, vagy 1.

Az $\bar{s}_1$ **által** η **vágási ponttal elfogadott nyelvet** az $L(PA, \bar{s}_1, \eta) = \{w \in V^* \mid s_0 M(w) \bar{s}_1 > \eta\}$ halmaz definiálja.

$\bar{s}_1$-ra úgy gondolhatunk, hogy 1 koordinátái kijelölik a végállapotok halmazát. $L(PA, \bar{s}_1, \eta)$ ekkor azon szavak halmaza, amelyekre η-nál nagyobb valószínűséggel kerül PA végállapotba.

Sztochasztikus automata

Definíció

Egy L nyelvet η-**sztochasztikusnak** mondunk, ha valamely $PA = \langle S, s_0, M \rangle$ véges sztochasztikus automatára és $\bar{s}_1$ oszlopvektorra $L = L(PA, \bar{s}_1, \eta)$ teljesül.

Definíció

Egy L nyelvet **sztochasztikusnak** nevezünk, ha valamely $0 \leqslant \eta < 1$-re η-sztochasztikus.

Az alábbi tételt nem bizonyítjuk.

Tétel

Minden reguláris nyelv sztochasztikus, de nem minden sztochasztikus nyelv reguláris. A 0-sztochasztikus nyelvek regulárisak.

Sztochasztikus automata

Példa: Tekintsük az előző példában szereplő automatát.
Emlékeztetőül:
$V = \{x, y\}$, $PA_1 = \langle \{s_1, s_2\}, (1,0), M \rangle$, ahol

$$M(x) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad M(y) = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

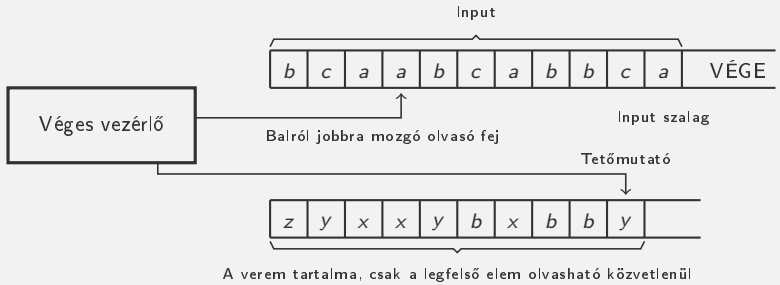
Ekkor
$PA_1(x^n) = (1,0)M(x^n) = (1,0)$, ha n páros,
$PA_1(x^n) = (0,1)$, ha n páratlan, és
$PA_1(w) = (1/2, 1/2)$, ha w legalább egy y -t tartalmaz.

Tehát $\bar{s}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ esetén

$$L(PA, \bar{s}_1, \eta) = \begin{cases} V^* - (xx)^* & \text{ha } 0 \leqslant \eta < 1/2 \\ x(xx)^* & \text{ha } 1/2 \leqslant \eta < 1 \end{cases}$$

Tehát $V^* - (xx)^*$ 1/3-sztochasztikus, míg $x(xx)^*$ 2/3-sztohasztikus nyelv. Így mindkettő sztochasztikus nyelv.

Verematautomata



- A verematautomata a véges automata általánosítása potenciálisan végtelen veremmel és véges kontrollal.
- A verem esetében az új adat mindig a már meglévő veremtartalom tetejéhez adódik, kivétele fordított sorrendben történik.
- alapértelmezetten nemdeterminisztikus

Verematautomata

Jelölés: ha X egy halmaz, jelölje $\mathcal{P}_{\text{véges}}(X)$ az X véges részhalmazainak halmazát.

Definíció

A **verematautomata** egy $A = \langle Z, Q, T, \delta, z_0, q_0, F \rangle$, rendezett hetes, ahol

- Z a veremszimbólumok véges halmaza (veremábécé),
- Q az állapotok véges halmaza,
- T az inputszimbólumok véges halmaza (inputábécé),
- $\delta : Z \times Q \times (T \cup \{\varepsilon\}) \to \mathcal{P}_{\text{véges}}(Z^* \times Q)$, az ún. átmeneti függvény,
- $z_0 \in Z$ a kezdeti (kezdő) veremszimbólum,
- $q_0 \in Q$ a kezdeti állapot (kezdőállapot),
- $F \subseteq Q$ az elfogadó állapotok vagy végállapotok halmaza.

Verematautomata

- A verem tetején lévő szimbólum, az aktuális állapot és az inputszimbólum együttesen határozzák meg a következő átmenetet.
- Minden lépésben mindenképpen kiveszünk egyetlen egy elemet a verem tetejéről és beteszünk helyette néhányat. $(0, 1, 2, \ldots$ darabot)
- Ha $\delta(z, q, \varepsilon)$ nem üres, akkor ún. **ε-átmenet** (ε-lépés, ε-mozgás) hajtható végre, ami lehetővé teszi, hogy a verematautomata anélkül változtassa meg az állapotát, hogy valamilyen szimbólumot olvasson az inputszalagról.
- ε-mozgásra lehetőség van már az első inputszimbólum elolvasása előtt is illetve még az utolsó inputszimbólum elolvasása után is.

Verematautomata konfigurációi

Definíció

A verematautomata **konfigurációja** alatt egy zqw alakú szót értünk, ahol $z \in Z^*$ a verem aktuális tartalma és $q \in Q$ az aktuális állapot és $w \in T^*$ az input még feldolgozatlan része.

z első betűje van a verem alján, míg utolsó betűje a verem tetején.

Az input olvasófeje w első betűjén áll.

Így a q baloldalán lévő szimbólum van a verem tetején, míg a jobboldalán lévő szimbólum az input következő feldolgozandó betűje.

Definíció

Az $A = \langle Z, Q, T, \delta, z_0, q_0, F \rangle$ verematautomata $w \in T^*$ bemenethez tartozó **kezdőkonfigurációja** $z_0 q_0 w$.

Alapvető veremműveletek megvalósítása

Legyen $t \in T \cup \{\varepsilon\}$, $q, r \in Q$ és $z \in Z$

- $(\varepsilon, r) \in \delta(z, q, t)$: a z elemet kivehetjük a veremből (POP művelet)
- $(z, r) \in \delta(z, q, t)$: a verem tartalma változatlan maradhat
- $(z', r) \in \delta(z, q, t)$: z-t lecserélhetjük z'-re a verem tetején ($z' \in Z$)
- $(zz', r) \in \delta(z, q, t)$: z'-t a verem tetejére (z-re rá) tehetjük ($z' \in Z$) (PUSH művelet)
- Egyéb lehetőségek, például $(zz'z'', r) \in \delta(z, q, t)$: $z'z''$-t a verem tetejére tehetjük, z'' lesz a tetején ($z', z'' \in Z$) .
- Általánosan $(w, r) \in \delta(z, q, t)$, ahol $w \in Z^*$ tetszőleges Z feletti szó. A w szó kerül z helyére és w utolsó betűje lesz a verem tetején.

Veremautomata – egylépéses redukció

Definíció

Az A veremautomata az $\alpha \in Z^*QT^*$ konfigurációt a $\beta \in Z^*QT^*$ konfigurációra **redukálja egy lépésben**, amelyet $\alpha \Rightarrow_A \beta$-val jelölünk, ha létezik olyan $z \in Z, q, p \in Q, a \in T \cup \{\varepsilon\}$, $r, u \in Z^*$ és $w \in T^*$, hogy $(u, p) \in \delta(z, q, a)$ és $\alpha = rzqaw$ és $\beta = rupw$ teljesül.

Példák:

- ha A-ban $\delta(c, q_1, a) = \{(dd, q_2), (\varepsilon, q_4)\}$ és $z_0cddcq_1ababba$ egy konfiguráció, akkor $z_0cddcq_1ababba \Rightarrow_A z_0cdddq_2babba$ és $z_0cddcq_1ababba \Rightarrow_A z_0cddq_4babba$ is teljesül,
- ha A-ban $\delta(c, q_3, \varepsilon) = \{(dd, q_2)\}$ és $z_0cddcq_3ababba$ egy konfiguráció, akkor $z_0cddcq_3ababba \Rightarrow_A z_0cdddq_2ababba$
- ha A-ban $\delta(c, q_5, \varepsilon) = \varnothing$ és $\delta(c, q_5, a) = \varnothing$, akkor nem létezik olyan C konfiguráció, melyre $z_0ccq_5aab \Rightarrow_A C$

Veremautomata – többlépéses redukció

Definíció

Az A veremautomata az $\alpha \in Z^*QT^*$ konfigurációt a $\beta \in Z^*QT^*$ konfigurációra **redukálja**, amelyet $\alpha \Rightarrow_A^* \beta$-val jelölünk, ha vagy $\alpha = \beta$, vagy létezik olyan $\alpha_1, \ldots, \alpha_n$ szavakból álló véges sorozat, ahol $\alpha = \alpha_1, \beta = \alpha_n$ és $\alpha_i \Rightarrow_A \alpha_{i+1}$, $1 \leqslant i \leqslant n-1$.

Tehát $\Rightarrow_A^* \subseteq Z^*QT^* \times Z^*QT^*$ a $\Rightarrow_A$ reláció reflexív, tranzitív lezártja.

Példa:

Ha $\delta(d, q_6, b) = \{(\varepsilon, q_5)\}$ és $\delta(d, q_5, \varepsilon) = \{(dd, q_2), (\varepsilon, q_4)\}$ akkor

$\#cddq_6bab \Rightarrow_A \#cdq_5ab \Rightarrow_A \#cdddq_2ab$ és

$\#cddq_6bab \Rightarrow_A \#cdq_5ab \Rightarrow_A \#cq_4ab$.

Tehát $\#cddq_6bab \Rightarrow_A^* \#cdddq_2ab$ és $\#cddq_6bab \Rightarrow_A^* \#cq_4ab$.

Veremautomata – felismert nyelv

Definíció

Az A veremautomata által **elfogadó állapottal (végállapottal) elfogadott nyelv**

$$L(A) = \{w \in T^* \mid z_0q_0w \Rightarrow_A^* up, \text{ ahol } u \in Z^*, p \in F\}.$$

Determinisztikus veremautomata

Definíció

Az $A = \langle Z, Q, T, \delta, z_0, q_0, F \rangle$ veremautomatát **determinisztikusnak** nevezzük, ha minden $(z, q, a) \in Z \times Q \times T$ esetén $|\delta(z, q, a)| + |\delta(z, q, \varepsilon)| = 1$.

Tehát minden $q \in Q$ és $z \in Z$ esetén

- vagy $\delta(z, q, a)$ pontosan egy elemet tartalmaz minden $a \in T$ inputszimbólumra és $\delta(z, q, \varepsilon) = \varnothing$,
- vagy $\delta(z, q, \varepsilon)$ pontosan egy elemet tartalmaz és $\delta(z, q, a) = \varnothing$ minden $a \in T$ inputszimbólumra.

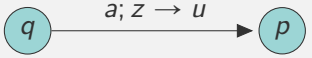
Észrevétel: Ha minden $(z, q, a) \in Z \times Q \times T$ esetén $|\delta(z, q, a)| + |\delta(z, q, \varepsilon)| \leqslant 1$ akkor a veremautomata a felismert nyelv módosulása nélkül kiegészíthető determinisztikus veremautomatává. Így tágabb értelemben az ezt a feltételt teljesítő veremautomatákat is tekinthetjük determinisztikus veremautomatának.

Veremautomaták alternatív reprezentációi

- **Átírási szabályokkal:**
  A δ leképezést szabályok formájában is megadhatjuk. Az így nyert szabályhalmazt M_δ -val jelöljük. Tehát ezzel az alternatív jelöléssel:

  $zqa \to up :\in M_\delta \iff (u, p) \in \delta(z, q, a)$,
  $zq \to up :\in M_\delta \iff (u, p) \in \delta(z, q, \varepsilon)$.
  $(p, q \in Q, a \in T, z \in Z, u \in Z^*)$
- **Átmenetdiagrammal:**

  $p, q \in Q, a \in T \cup \{\varepsilon\}, z \in Z, u \in Z^*$ esetén:

  $\iff (u, p) \in \delta(z, q, a)$

  A végállapotokat duplán karikázzuk. A kezdőállapotot $\to$ jelöli.

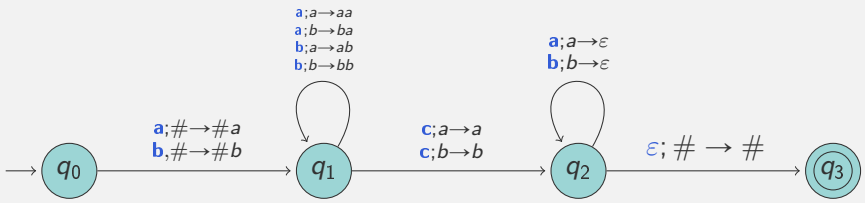
Veremautomata – példa

1. Példa: Legyen $L_1 = \{wcw^{-1} \mid w \in \{a, b\}^+\}$. Készítsünk egy A veremautomatát, melyre $L(A) = L_1$.

Megoldás:
$A = \langle \{q_0, q_1, q_2, q_3\}, \{\mathbf{a},\mathbf{b},\mathbf{c}\}, \{\#, a, b\}, \delta, q_0, \#, \{q_3\} \rangle$, ahol:

$$
\begin{array}{rl}
(\#t, q_1) \in \delta(\#, q_0, \mathbf{t}) & \forall t \in \{a, b\} \\
(zt, q_1) \in \delta(z, q_1, \mathbf{t}) & \forall z, t \in \{a, b\} \\
(z, q_2) \in \delta(z, q_1, \mathbf{c}) & \forall z \in \{a, b\} \\
(\varepsilon, q_2) \in \delta(t, q_2, \mathbf{t}) & \forall t \in \{a, b\} \\
(\#, q_3) \in \delta(\#, q_2, \varepsilon) &
\end{array}
$$

DETERMINISZTIKUS



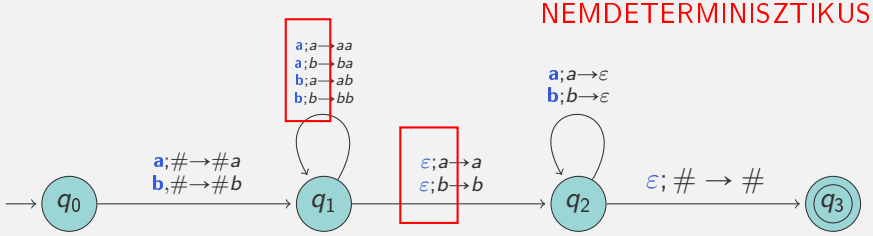
Veremautomata – példa

2. Példa: Legyen $L_2 = \{ww^{-1} \mid w \in \{a, b\}^+\}$. Készítsünk egy A veremautomatát, melyre $L(A) = L_2$.

Megoldás:
$A = \langle \{q_0, q_1, q_2, q_3\}, \{\mathbf{a},\mathbf{b}\}, \{\#, a, b\}, \delta, q_0, \#, \{q_3\} \rangle$, ahol:

$$
\begin{array}{rl}
(\#t, q_1) \in \delta(\#, q_0, \mathbf{t}) & \forall t \in \{a, b\} \\
(zt, q_1) \in \delta(z, q_1, \mathbf{t}) & \forall z, t \in \{a, b\} \\
(z, q_2) \in \delta(z, q_1, \varepsilon) & \forall z \in \{a, b\} \\
(\varepsilon, q_2) \in \delta(t, q_2, \mathbf{t}) & \forall t \in \{a, b\} \\
(\#, q_3) \in \delta(\#, q_2, \varepsilon) &
\end{array}
$$

NEMDETERMINISZTIKUS



Üres veremmel elfogadott nyelv

Definíció

Az A veremautomata által **üres veremmel elfogadott nyelv** $N(A) = \{w \in T^* \mid z_0 q_0 w \Rightarrow_A^* p, \text{ ahol } p \in Q\}$.

Megjegyzés:
Vegyük észre, hogy ha a verem üres, akkor az automata működése blokkolódik, mivel nincs átmenet definiálva üres verem esetére. (lásd δ definíciója).

Így a verem az input teljes feldolgozása után, az utolsó átmenettel kell üressé váljon.

Szintén a blokkolás elkerülése végett definiáltuk úgy a kezdőkonfigurációt, hogy a veremábécé egy eleme (z_0) már eleve a veremben van.

Megjegyzés: Vegyük észre, hogy az elfogadó állapotok halmaza irreleváns $N(A)$ szempontjából.

Üres veremmel elfogadott nyelv

Példa: Az alábbi $A = \langle \{\$, a\}\{q_0, q_1\}, \{a, b\}, \delta, \$, q_0, \{\,\}\rangle$ veremautomata esetén $N(A) = \{a^n b^n \mid n \geqslant 1\}$, azaz ezt nyelvet ismeri fel üres veremmel.

M_δ :

$\$q_0 a \to \$aq_0$
$aq_0 a \to aaq_0$
$aq_0 b \to q_1$
$aq_1 b \to q_1$
$\$q_1 \to q_1$.

A determinisztikus, a^2b^3-re:

$\$q_0 aabbb \Rightarrow \$aq_0 abbb \Rightarrow \$aaq_0 bbb \Rightarrow \$aq_1 bb \Rightarrow \$q_1 b \Rightarrow q_1 b$.

A elutasítja $aabbb$-t, mivel hiába lett üres a verem, még volt hátra az inputból.

A veremautomaták számítási ereje

Tétel

Bármely L nyelvre ekivalensek a következő állítások

- L környezetfüggetlen, azaz környezetfüggetlen (2-es típusú) grammatikával generálható
- L (nemdeterminisztikus) veremautomatával végállapottal felismerhető
- L (nemdeterminisztikus) veremautomatával üres veremmel felismerhető

A determinisztikus veremautomaták számítási ereje kisebb.

Tétel

Minden reguláris (3-as típusú) nyelv felismerhető determinisztikus veremautomatával, de létezik olyan (2-es típusú) környezetfüggetlen nyelv, ami nem ismerhető fel determinisztikus veremautomatával.

Ilyen nyelv például $L_2 = \{ww^{-1} \mid w \in \{a, b\}^+\}$.

Nagy Bar-Hillel lemma

Egy szükséges feltétel egy nyelv környezetfüggetlenségére (és így veremautomatával való felismerhetőségére is).

Tétel (Nagy Bar-Hillel lemma)

Minden L környezetfüggetlen nyelvhez meg tudunk adni két, p és q természetes számot úgy, hogy minden olyan szó L-ben, amely hosszabb, mint p felírható $uxwyz$ alakban, ahol $|xwy| \leqslant q, xy \neq \varepsilon$, továbbá, ekkor minden $ux^i wy^i z$, $i \geqslant 0$ alakú szó is benne van az L nyelvben ($u, x, w, y, z \in T^*$).

Példák: ($|u|_t$: a t betűk száma u-ban)

$\in \mathcal{L}_3$	$\in \mathcal{L}_2 - \mathcal{L}_3$	$\in \mathcal{L}_1 - \mathcal{L}_2$
$\{u \mid abbab \subseteq u\}$	$\{u \in \{a, b\}^* \mid u = u^{-1}\}$	$\{uu \mid u \in \{a, b\}^*\}$
$\{u \mid abbab \not\subseteq u\}$	$\{a^n b^n \mid n \in \mathbb{N}\}$	$\{a^n b^n c^n \mid n \in \mathbb{N}\}$
7-tel osztható számok	$\{u \in \{a, b\}^* \mid \lvert u\rvert_a = \lvert u\rvert_b\}$	$\{a^{n^2} \mid n \in \mathbb{N}\}$
$((a + bb)^* + ab)^*$	helyes ()-k nyelve	$\{a^{2^n} \mid n \in \mathbb{N}\}$

Számítási modellek

3. előadás

Az algoritmikus fogalmának modelljei

Az 1930-as évektől egyre nagyobb igény mutatkozott az algoritmus matematikai modelljének megalkotására. Egymástól függetlenül több bíztató kísérlet is született:

- Kurt Gödel: rekurzív függvények
- Alonso Church: λ-kalkulus
- Alan Turing: Turing gép

Melyik az „igazi", melyiket válasszuk?

Az 1930-as évek második felétől sorra születtek olyan tételek, melyek ezen modellek megegyező számítási erejét mondták ki. A későbbiek során számos további számítási modellről sikerült bebizonyítani, hogy számítási erejük a Turing gépekkel ekvivalens. Például:

- 0. típusú grammatika
- veremautomata 2 vagy több veremmel
- C, Java, stb.

A Church-Turing tézis

Valójában nem ismerünk olyan algoritmikus rendszert, amelyről tudnánk, hogy erősebb a Turing gépnél, és a legtöbb algoritmikus rendszerre bizonyított, hogy gyengébb, vagy ekvivalens.

Már a 30-as években megfogalmazásra került a következő:

Church-Turing tézis

Minden formalizálható probléma, ami megoldható algoritmussal, az megoldható Turing géppel is.

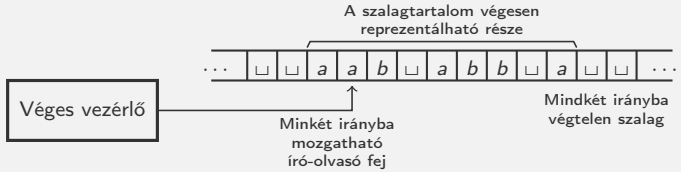
(illetve bármilyen, a Turing géppel azonos számítási erejű absztrakt modellben)

NEM TÉTEL!!!

A Church-Turing tézis nem bizonyítható, hiszen nem egy formális matematikai állítás, az algoritmus intuitív fogalmát használja.

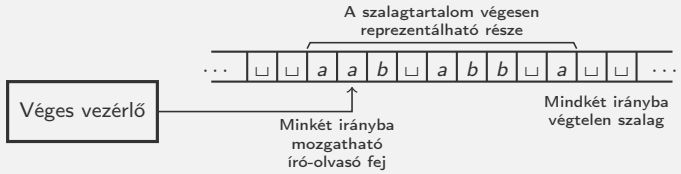
Ha elfogadjuk a tézis igazságát, a Turing gép (illetve bármely a Turing gépekkel ekvivalens modell) informálisan tekinthető az algoritmus matematikai modelljének.

Turing gépek – Informális kép



- a Turing gép (TG) az algoritmus egyik lehetséges modellje
- a TG egyetlen programot hajt végre (de bármely inputra!!!), azaz tekinthető egy célszámítógépnek.
- informálisan a gép részei a vezérlőegység (véges sok állapottal), egy mindkét irányba végtelen szalag, és egy mindkét irányba lépni képes író-olvasó fej
- kezdetben egy input szó van a szalagon (ε esetén üres), a fej ennek első betűjéről indul, majd a szabályai szerint működik. Ha eljut az elfogadó állapotába elfogadja, ha eljut az elutasító állapotába elutasítja az inputot. Van egy harmadik lehetőség is: nem jut el soha a fenti két állapotába, "végtelen ciklusba" kerül.

Turing gépek – Informális kép



- a gép alapértelmezetten determinisztikus, minden (nem megállási) konfigurációnak van és egyértelmű a rákövetkezője.
- végtelen szalag: potenciálisan végtelen tár
- egy $\mathcal{P}$ probléma példányait egy megfelelő ábécé felett elkódolva a probléma „igen"-példányai egy $L(\mathcal{P})$ formális nyelvet alkotnak. $L(\mathcal{P})$ (és így a probléma maga is) algoritmikusan eldönthető, ha van olyan mindig termináló Turing gép, mely pontosan $L(\mathcal{P})$ szavait fogadja el.
- a Church-Turing tézis értelmében informálisan úgy gondolhatjuk, hogy éppen a TG-pel eldönthető problémák (nyelvek) az algoritmikusan eldönthető eldöntési problémák.

Turing gépek

Definíció

A **Turing gép** (továbbiakban röviden TG) egy $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n \rangle$ rendezett hetes, ahol

- Q az állapotok véges, nemüres halmaza,
- $q_0, q_i, q_n \in Q$, q_0 a kezdő- q_i az elfogadó- és q_n az elutasító állapot,
- Σ és Γ ábécék, a bemenő jelek illetve a szalagszimbólumok ábécéje úgy, hogy $\Sigma \subseteq \Gamma$ és $\sqcup \in \Gamma \backslash \Sigma$.
- $\delta : (Q \backslash \{q_i, q_n\}) \times \Gamma \to Q \times \Gamma \times \{L, S, R\}$ az átmenet függvény. δ értelmezési tartománya $(Q \backslash \{q_i, q_n\}) \times \Gamma$.

$\{L, S, R\}$ elemeire úgy gondolhatunk mint a TG lépéseinek irányaira (balra lépés, helyben maradás, jobbra lépés).
Megjegyzés: Elég 2 irány, a helyben maradó lépések helyettesíthetők egy jobbra és egy balra lépéssel egy, csak erre az átmenetre használt új állapoton keresztül.

Turing gépek

Konfiguráció

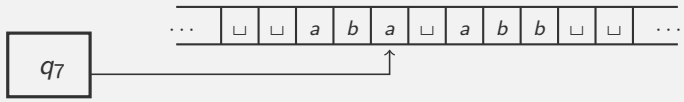
A TG működtetését a gép konfigurációival írhatjuk le.

Definíció

A TG **konfigurációja** egy uqv szó, ahol $q \in Q$ és $u, v \in \Gamma^*, v \neq \varepsilon$.

Az uqv konfiguráció egy tömör leírás a TG aktuális helyzetéről, mely a gép további működése szempontjából minden releváns információt tartalmaz: a szalag tartalma uv (uv előtt és után a szalagon már csak $\sqcup$ van), a gép a q állapotban van és az író-olvasó fej a v szó első betűjén áll.

Példa:



A fenti helyzetet az $abq_7a\sqcup abb$ konfigurációval írhatjuk le.

Turing gépek

Konfiguráció

Definíció

A gép egy $u \in \Sigma^*$-beli szóhoz tartozó **kezdőkonfigurációja** a $q_0u\sqcup$ szó. (Vagyis q_0u, ha $u \neq \varepsilon$ és $q_0\sqcup$, ha $u = \varepsilon$).

Elfogadó konfigurációi azon konfigurációk, melyre $q = q_i$.

Elutasító konfigurációi azon konfigurációk, melyre $q = q_n$.

Az elfogadó és elutasító konfigurációk közös neve **megállási konfiguráció**.

Két konfigurációt **azonosnak tekintünk**, ha csak balra/jobbra hozzáírt $\sqcup$-ekben térnek el egymástól.

Például a $\sqcup abq_2\sqcup$ és a $abq_2\sqcup\sqcup$ konfigurációk azonosak.

Turing gépek

Konfigurácóátmenet

Jelölje C_M egy M TG-hez tartozó lehetséges konfigurációk halmazát. Az M Turing gép $\vdash \subseteq C_M \times C_M$ **konfigurációátmenet relációját** az alábbiak szerint definiáljuk. (közvetlen v. egylépéses konfigurációátmenet)

Definíció

Legyen $uqav$ egy konfiguráció, ahol $a \in \Gamma$, $u, v \in \Gamma^*$.

- Ha $\delta(q,a) = (r,b,R)$, akkor $uqav \vdash ubrv'$, ahol $v' = v$, ha $v \neq \varepsilon$, különben $v' = \sqcup$,
- ha $\delta(q,a) = (r,b,S)$, akkor $uqav \vdash urbv$,
- ha $\delta(q,a) = (r,b,L)$, akkor $uqav \vdash u'rcbv$, ahol $c \in \Gamma$ és $u'c = u$, ha $u \neq \varepsilon$, különben $u' = u$ és $c = \sqcup$.

Példa: Tegyük fel, hogy $\delta(q_2, a) = (q_5, b, L)$ és $\delta(q_5, c) = (q_1, \sqcup, R)$. Legyen továbbá $C_1 = bcq_2a\sqcup b$, $C_2 = bq_5cb\sqcup b$, $C_3 = b\sqcup q_1b\sqcup b$. Ekkor $C_1 \vdash C_2$ és $C_2 \vdash C_3$.

Turing gépek

Konfigurációátmenet, felismert nyelv

Az $\vdash^* \subseteq C_M \times C_M$ többlépéses konfigurációátmenet reláció a $\vdash$ reláció reflexív, tranzitív lezártja.

Példa: (folytatás) Legyen C_1, C_2, C_3 ugyanaz, mint a fenti példában. Mivel $C_1 \vdash C_2$ és $C_2 \vdash C_3$ is teljesült, ezért $C_1 \vdash^* C_1$, $C_1 \vdash^* C_2$, $C_1 \vdash^* C_3$ is fennállnak.

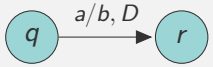
Definíció

Az M TG által **felismert nyelv**
$L(M) = \{u \in \Sigma^* \mid q_0u\sqcup \vdash^* xq_iy$ valamely $x, y \in \Gamma^*, y \neq \varepsilon$ -ra$\}$.

Figyeljük meg, hogy $L(M)$ csak Σ feletti szavakat tartalmaz.

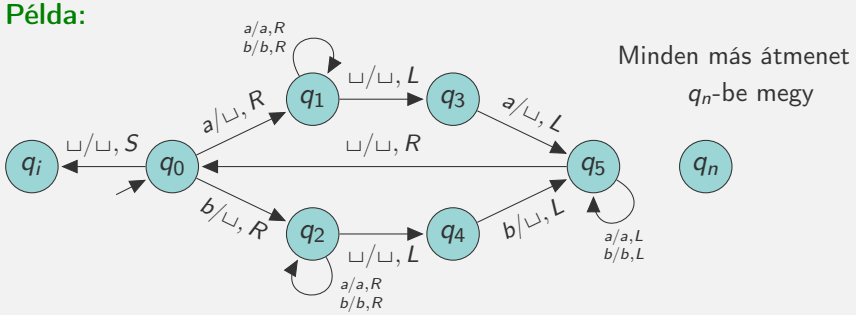
Turing gépek – Példa

Az **átmenetdiagram**:



megfelel a $\delta(q,a) = (r,b,D)$ átmenetnek
$(q, r \in Q, a, b \in \Gamma, D \in \{L,S,R\})$

Példa:



Minden más átmenet q_n-be megy

$L(M) = \{ww^{-1} \mid w \in \{a,b\}^*\}$. Példa konfigurációátmenetek sorozatára az aba input esetén:
$q_0aba \vdash q_1ba \vdash bq_1a \vdash baq_1\sqcup \vdash bq_3a \vdash q_5b \vdash q_5\sqcup b \vdash q_0b \vdash q_2\sqcup \vdash q_4\sqcup \vdash q_n\sqcup$.

Turing gépek időigénye

Definíció

Az M Σ inputábécéjű TG **futási ideje** az $u \in \Sigma^*$ szóra a konfigurációátmenetek száma az u-hoz tartozó kezdőkonfigurációból egy megállási konfigurációba.

Vagyis a működés során megtett lépések száma. Ha a gép nem áll meg egy szóra, akkor a futási ideje erre a szóra ∞.

Példa: Az előző példában a TG-ünk az aba inputra 10 lépésben jutott megállási konfigurációba, így aba-ra a futási idő 10.

Definíció

Legyen $f : \mathbb{N} \to \mathbb{N}$ egy függvény. Azt mondjuk, hogy az M Σ inputábécéjű TG $f(n)$ **időkorlátos** (vagy M $f(n)$ időigényű), ha minden $u \in \Sigma^*$ szóra M futási ideje $\leqslant f(|u|)$.

Példa: Meggondolható, hogy az előző példában látott TG $O(n)$ iterációt végez iterációnként $O(n)$ lépéssel, így $O(n^2)$ időkorlátos.

Rekurzíve felsorolható és rekurzív nyelvek

Definíció

Egy $L \subseteq \Sigma^*$ nyelv **Turing-felismerhető**, ha $L = L(M)$ valamely M TG-re.

Egy $L \subseteq \Sigma^*$ nyelv **eldönthető**, ha létezik olyan M TG, mely minden bemeneten megállási konfigurációba jut és $L(M) = L$.

A Turing-felismerhető nyelveket **rekurzívan felsorolhatónak** (vagy **parciálisan rekurzívnak**, vagy **félig eldönthetőnek**) az eldönthető nyelveket pedig **rekurzívnak** is nevezik.

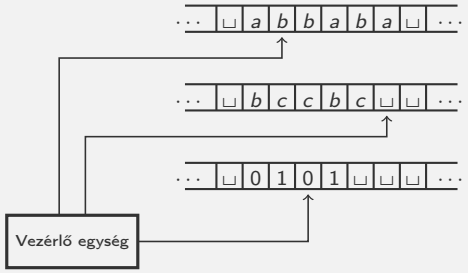
Definíció:

$\mathrm{RE} = \{L \mid L \text{ Turing-felismerhető}\}$,

$\mathrm{R} = \{L \mid L \text{ eldönthető}\}$.

Nyilván $\mathrm{R} \subseteq \mathrm{RE}$.

Többszalagos Turing gép – Informális kép



- Véges vezérlő egység, $k(\geqslant 1)$ darab kétirányba végtelen szalag, minden szalaghoz egy-egy saját író-olvasó fej.
- Egy ütem: Minden szalagról a fejek által mutatott betűk egyszerre történő beolvasása, átírása és a fejek léptetése egyszerre, de egymástól független irányokba.
- Az egyszalagos géppel analóg elfogadás fogalom.
- Az egyszalagos géppel analóg időigény fogalom (1 lépés = 1 ütem).

k-szalagos Turing gép

Definíció

Adott egy $k \geqslant 1$ egész szám. A ***k*-szalagos Turing gép** egy olyan $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n \rangle$ rendezett hetes, ahol

- Q az állapotok véges, nemüres halmaza,
- $q_0, q_i, q_n \in Q$, q_0 a kezdő- q_i az elfogadó- és q_n az elutasító állapot,
- Σ és Γ ábécék, a bemenő jelek illetve a szalagszimbólumok ábécéje úgy, hogy $\Sigma \subseteq \Gamma$ és $\sqcup \in \Gamma \backslash \Sigma$,
- $\delta : (Q \backslash \{q_i, q_n\}) \times \Gamma^k \to Q \times \Gamma^k \times \{L, S, R\}^k$ az átmenet függvény.

δ az egész $(Q \backslash \{q_i, q_n\}) \times \Gamma^k$-n értelmezett függvény.

k-szalagos Turing gépek konfigurációi

Definíció

k-szalagos TG **konfigurációja** egy $(q, u_1, v_1, \ldots, u_k, v_k)$ szó, ahol $q \in Q$ és $u_i, v_i \in \Gamma^*, v_i \neq \varepsilon$ $(1 \leqslant i \leqslant k)$.

Ez azt reprezentálja, hogy

- az aktuális állapot q és
- az i. szalag tartalma $u_i v_i$ $(1 \leqslant i \leqslant k)$ és
- az i. fej v_i első betűjén áll $(1 \leqslant i \leqslant k)$.

Definíció

Az u szóhoz tartozó **kezdőkonfiguráció**: $u_i = \varepsilon$ $(1 \leqslant i \leqslant k)$, $v_1 = u\sqcup$, és $v_i = \sqcup$ $(2 \leqslant i \leqslant k)$.

Azaz, az input szó az első szalagon van, ennek az első betűjéről indul az első szalag feje. A többi szalag kezdetben üres.

k-szalagos Turing gépek megállási konfigurációi

Definíció

A $(q, u_1, v_1, \ldots, u_k, v_k)$ konfiguráció, ahol $q \in Q$ és $u_i, v_i \in \Gamma^*$, $v_i \neq \varepsilon$ $(1 \leqslant i \leqslant k)$,

- **elfogadó konfiguráció**, ha $q = q_i$,
- **elutasító konfiguráció**, ha $q = q_n$,
- **megállási konfiguráció**, ha $q = q_i$ vagy $q = q_n$.

k-szalagos TG-ek egylépéses konfigurációátmenete

Definíció

Egy $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n \rangle$ k-szalagos Turing gép $\vdash \subseteq C_M \times C_M$ **egylépéses konfigurációátmenet** relációját az alábbiak szerint definiáljuk.

Legyen A $C = (q, u_1, a_1 v_1, \ldots, u_k, a_k v_k)$ egy konfiguráció, ahol $a_i \in \Gamma$, $u_i, v_i \in \Gamma^*$ $(1 \leqslant i \leqslant k)$. Legyen továbbá $\delta(q, a_1, \ldots, a_k) = (r, b_1, \ldots, b_k, D_1, \ldots, D_k)$, ahol $q, r \in Q$, $b_i \in \Gamma, D_i \in \{L, S, R\}$ $(1 \leqslant i \leqslant k)$. Ekkor $C \vdash (r, u_1', v_1', \ldots, u_k', v_k')$, ahol minden $1 \leqslant i \leqslant k$-ra

- ha $D_i = R$, akkor $u_i' = u_i b_i$ és $v_i' = v_i$, ha $v_i \neq \varepsilon$, különben $v_i' = \sqcup$,
- ha $D_i = S$, akkor $u_i' = u_i$ és $v_i' = b_i v_i$,
- ha $D_i = L$, akkor $u_i = u_i' c$ $(c \in \Gamma)$ és $v_i' = c b_i v_i$ ha $u_i \neq \varepsilon$, különben $u_i' = \varepsilon$ és $v_i' = \sqcup b_i v_i$.

k-szalagos TG-ek többlépéses konfigurációátmenete

Tehát egy szalagjára vetítve a többszalagos TG pont úgy működik, mint az egyszalagos TG.

Példa:

Legyen k=2 és $\delta(q, a_1, a_2) = (r, b_1, b_2, R, S)$ a TG egy átmenete. Ekkor $(q, u_1, a_1 v_1, u_2, a_2 v_2) \vdash (r, u_1 b_1, v_1', u_2, b_2 v_2)$, ahol $v_1' = v_1$, ha $v_1 \neq \varepsilon$, különben $v_1' = \sqcup$.

Vegyük észre, hogy a fejek nem kell, hogy egyazon irányba lépjenek.

A k-szalagos TG-ek **többlépéses konfigurációátmenet** relációját ugyanúgy definiáljuk, mint az egyszalagos esetben. Jelölés: $\vdash^*$.

k-szalagos TG – felismert nyelv, időigény

Definíció

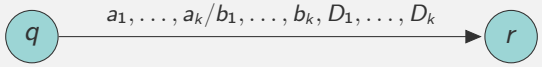
$M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n \rangle$ k-szalagos TG által felismert nyelv: $L(M) = \{ u \in \Sigma^* \mid (q_0, \varepsilon, u\sqcup, \varepsilon, \sqcup, \ldots, \varepsilon, \sqcup) \vdash^* (q_i, x_1, y_1, \ldots x_k, y_k),$ valamely $x_1, y_1 \ldots, x_k, y_k \in \Gamma^*, y_1, \ldots, , y_k \neq \varepsilon$ -ra$\}$.

Azaz, csakúgy mint az egyszalagos esetben, azon inputábécé feletti szavak halmaza, melyekkel (az első szalagján) a TG-et indítva az az elfogadó, q_i állapotában áll le.

- A k-szalagos TG-ek által **felismerhető** illetve **eldönthető** nyelvek fogalma szintén analóg az egyszalagos esettel.
- Egy k-szalagos Turing gép **futási ideje** egy u szóra a hozzá tartozó kezdőkonfigurációból egy megállási konfigurációba megtett lépések száma.
- Az **időigény** definíciója megegyezik az egyszalagos esetnél tárgyalttal.

k-szalagos Turing gép – átmenetdiagram

A k-szalagos TG-ek **átmenetdiagramja** egy csúcs- és élcímkézett irányított gráf, melyre

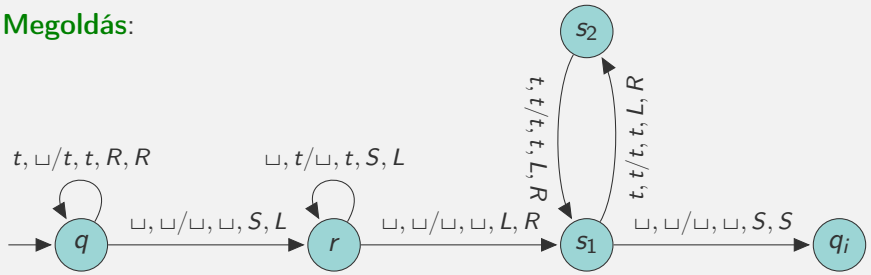


$$\iff \delta(q, a_1, \ldots, a_k) = (r, b_1, \ldots, b_k, D_1, \ldots, D_k)$$
$$(q, r \in Q, a_1, \ldots, a_k, b_1, \ldots, b_k \in \Gamma, D_1, \ldots, D_k \in \{L, S, R\})$$

Feladat: Készítsünk egy M kétszalagos Turing gépet, melyre $L(M) = \{ww^{-1} \mid w \in \{a, b\}^*\}$!

k-szalagos Turing gép – példa

Megoldás:



$t \in \{a, b\}$ tetszőleges. A többi átmenet q_n-be megy.

Például $(q, \varepsilon, abba, \varepsilon, \sqcup) \vdash (q, a, bba, a, \sqcup) \vdash (q, ab, ba, ab, \sqcup) \vdash$
$(q, abb, a, abb, \sqcup) \vdash (q, abba, \sqcup, abba, \sqcup) \vdash (r, abba, \sqcup, abb, a) \vdash$
$(r, abba, \sqcup, ab, ba) \vdash (r, abba, \sqcup, a, bba) \vdash (r, abba, \sqcup, \varepsilon, abba) \vdash$
$(r, abba, \sqcup, \varepsilon, \sqcup abba) \vdash (s_1, abb, a, \varepsilon, abba) \vdash (s_2, ab, ba, a, bba) \vdash$
$(s_1, a, bba, ab, ba) \vdash (s_2, \varepsilon, abba, abb, a) \vdash$
$(s_1, \varepsilon, \sqcup abba, abba, \sqcup) \vdash (q_i, \varepsilon, \sqcup abba, abba, \sqcup)$

Mennyi a TG időigénye? Ez egy $O(n)$ időkorlátos TG, mivel egy n hosszú inputra legfeljebb $3n + 3$ lépést tesz.

k-szalagos TG szimulálása egyszalagossal

Definíció

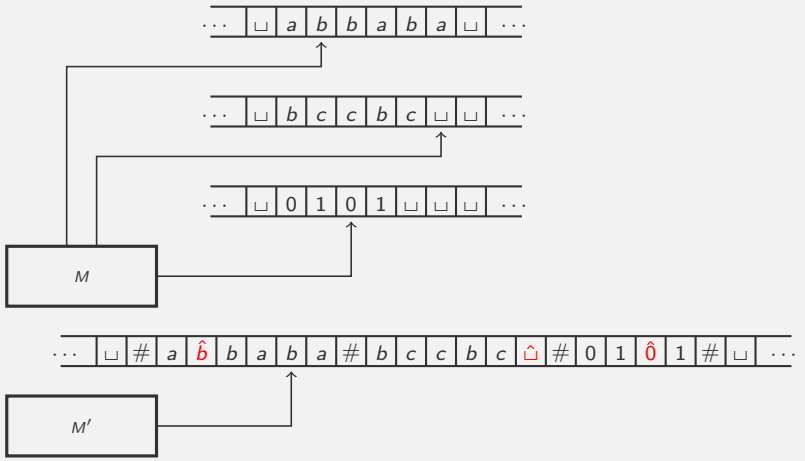
Két TG **ekvivalens**, ha ugyanazt a nyelvet ismerik fel.

Tétel

Minden M k-szalagos Turing géphez megadható egy vele ekvivalens M' egyszalagos Turing gép. Továbbá, ha M legalább lineáris időigényű $f(n)$ időkorlátos gép (azaz $f(n) = \Omega(n)$), akkor M' $O(f(n)^2)$ időkorlátos.

Többszalagos TG szimulálása egyszalagossal

A szimuláció alapötlete:



A szalagok tartalmát #-ekkel elválasztva eltárolhatjuk M' egyetlen szalagján, a fejek helyzetét ˆ-al megjelölve.

Turing-gép egy irányban végtelen szalaggal

Egyirányban végtelen szalagos TG: Bal oldalon zárt a TG szalagja, a fej nem tud "leesni", ha a legbaloldalibb cellán balra lépés az utsítás, akkor a fej helyben marad.

Tétel

Minden egyszalagos M TG-hez van vele ekvivalens egyirányban végtelen szalagos M'' TG.

A bizonyítás ötlete, hogy először egy kétszalagos egyirányban végtelen szalagos géppel szimulálunk egy egyszalagos kétirányban végtelen szalagost (az első szalagon tárolva a kezdőpozíciótól jobbra levő tartalmat, a másodikon pedig a balra lévő tartalom tükörképét). Majd a kétszalagos egyirányban végtelen szalagos gépet szimuláljuk egy ugyanilyen egyszalagossal.

Megjegyzés: Nyilvánvalóan a másik irány is igaz.

Nemdeterminisztikus Turing gép

Definíció

Az (egyszalagos) **nemdeterminisztikus Turing gép** (NTG) csak átmenetfüggvényében különbözik a determinisztikustól.
$\delta : (Q \backslash \{q_i, q_n\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, S, R\})$.

Azaz míg a **determinisztikus** esetben a δ átmenetfüggvény minden egyes $(Q \backslash \{q_i, q_n\}) \times \Gamma$-beli párhoz **pontosan egy**, addig egy **nemdeterminisztikus** TG **akárhány** (pl. 0,1,5,100) darab $Q \times \Gamma \times \{L, S, R\}$-beli rendezett hármast rendelhet hozzá.

Nemdeterminisztikus Turing-gép

egylépéses konfigurációátmenet

A konfiguráció fogalma azonos, jelölje most is C_M az M gép lehetséges konfigurációinak halmazát.A $\vdash \subseteq C_M \times C_M$ **egylépéses konfigurációátmenet** relációt a következőképpen definiáljuk.

Definíció

Legyen $uqav$ egy konfiguráció, ahol $a \in \Gamma$, $u, v \in \Gamma^*$.

- Ha $(r, b, R) \in \delta(q, a)$, akkor $uqav \vdash ubrv'$, ahol $v' = v$, ha $v \neq \varepsilon$, különben $v' = \sqcup$,
- ha $(r, b, S) \in \delta(q, a)$, akkor $uqav \vdash urbv$,
- ha $(r, b, L) \in \delta(q, a)$, akkor $uqav \vdash u'rcbv$, ahol $c \in \Gamma$ és $u'c = u$, ha $u \neq \varepsilon$, különben $u' = u$ és $c = \sqcup$.

Míg a **determinisztikus** esetben minden nem-megállási C konfigurációhoz **pontosan egy** C' konfiguráció létezett, melyre $C \vdash C'$, addig a **nemdeterminisztikus** esetben **több** ilyen is létezhet (de véges sok, hiszen $|Q \times \Gamma \times \{L, S, R\}|$ véges).

Nemdeterminisztikus Turing-gép

többlépéses konfigurációátmenet, felismert nyelv

Többlépéses konfigurációátmenet: $\vdash$ reflexív tranzitív lezártja, jelölése $\vdash^*$.

Legyen C egy konfiguráció. Míg a **determinisztikus** esetben **legfeljebb egy** C' megállási konfiguráció létezhetett, melyre $C \vdash^* C'$, addig a **nemdeterminisztikus** esetben **több** ilyen is létezhet.

Definíció

Az M NTG által felismert nyelv
$L(M) = \{u \in \Sigma^* \mid q_0 u \sqcup \vdash^* x q_i y$ valamely $x, y \in \Gamma^*, y \neq \varepsilon$ -ra$\}$.

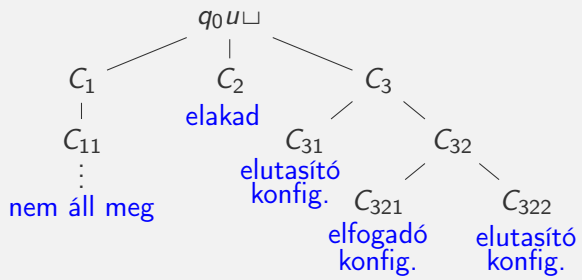
$\vdash^*$ fogalmának módosulása miatt a felismert nyelv fogalma is módosult. Egy NTG-re úgy gondolhatunk, hogy **több számítása is lehet egyazon szóra**. Akkor fogad el egy szót, ha az adott szóra **legalább egy számítása q_i-ben ér véget**.

Nemdeterminisztikus Turing-gép

Definíció

Egy M TG egy $u \in \Sigma^*$ inputjához tartozó **nemdeterminisztikus számítási fája** egy gyökeres fa, melynek csúcsai M konfigurációival címkézettek. $q_0 u \sqcup$ a gyökér címkéje. Ha C egy csúcs címkéje, akkor $|\{C' \mid C \vdash C'\}|$ gyereke van és ezek címkéi éppen $\{C' \mid C \vdash C'\}$ elemei.

Példa:



M elfogadja u-t, hiszen a $q_0 u \sqcup \vdash C_3 \vdash C_{32} \vdash C_{321}$ számítása elfogadó konfigurációba visz. **Egyetlen** elfogadó számítás is elég!

Nemdeterminisztikus Turing-gép

Tehát adott inputra több számítás is lehetséges, ezek lehetnek elfogadóak, elutasítóak, elakadóak (ha olyan C-be jut, melyre $\{C' \mid C \vdash C'\} = \varnothing$), illetve végtelenek.

Észrevétel: $u \in L(M) \Leftrightarrow$ az u-hoz tartozó nemdeterminisztikus számítási fának van olyan levele, ami elfogadó konfiguráció.

Definíció

Az M NTG **eldönti** az $L \subseteq \Sigma^*$ nyelvet, ha felismeri továbbá minden $u \in \Sigma^*$ input szóhoz tartozó nemdeterminisztikus számítási fa véges és a fa minden levele elfogadó vagy elutasító konfiguráció.

Definíció

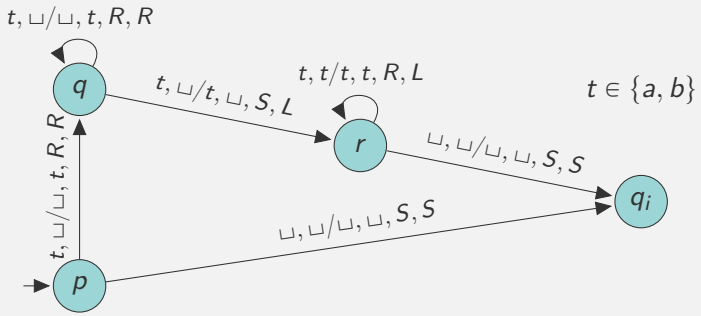
Az M NTG $f(n)$ **időkorlátos** (időigényű), ha minden $u \in \Sigma^*$ n hosszú szóra u számítási fája legfeljebb $f(n)$ magas.

Megjegyzés: a nemdeterminisztikus Turing-gép definíciója értelemszerűen kiterjeszthető k-szalagos gépekre is, így beszélhetünk k-szalagos nemdeterminisztikus Turing-gépekről is.

Nemdeterminisztikus Turing-gép

Példa

Az alábbi M nemdeterminisztikus Turing-gépre $L(M) = \{ww^{-1} \mid w \in \{a,b\}^*\}$.



$(p, \varepsilon, abba, \varepsilon, \sqcup) \vdash (q, \varepsilon, bba, a, \sqcup) \vdash (r, \varepsilon, bba, \varepsilon, a) \vdash (q_n, \varepsilon, bba, \varepsilon, a)$

$(p, \varepsilon, abba, \varepsilon, \sqcup) \vdash (q, \varepsilon, bba, a, \sqcup) \vdash (q, \varepsilon, ba, ab, \sqcup) \vdash (r, \varepsilon, ba, a, b) \vdash (r, b, a, \varepsilon, ab) \vdash (r, ba, \sqcup, \varepsilon, \sqcup ab) \vdash (q_i, ba, \sqcup, \varepsilon, \sqcup ab)$

NTG szimulálása TG-pel

Tétel

Minden $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n \rangle$ $f(n)$ idejű NTG-hez megadható egy ekvivalens, $2^{O(f(n))}$ idejű M' determinisztikus TG.

A szimuláció alapötlete: Egy w bemenetre járja be M' az M gép w-hez tartozó nemdeterminisztikus számítási fájának csúcsait szélességi bejárással.

Minden csúcs megfeleltethető egy a gyökértől az adott csúcsig tartó parciális számításnak.

Ha a csúcsnak megfelelő számítás nem ért még véget vagy pedig egy elutasító számításnak felel meg, akkor vegyük a szélességi bejárás szerinti következő csúcsot.

Amennyiben a csúcs egy elfogadó számításnak felel meg M' fogadja el w-t.

Időigény: A legfeljebb n hosszúságú, a fa gyökeréből induló utak összhossza n-nek exponenciális függvénye.

Nemdeterminisztikus Turing-gép

Következmények:

1. Egy nyelv Turing-felismerhető, akkor és csak akkor, ha valamely nemdeterminisztikus Turing-gép felismeri.
2. Egy nyelv eldönthető, akkor és csak akkor, ha van azt eldöntő nemdeterminisztikus Turing-gép.

Turing-ekvivalens számítási modellek

Most bemutatunk néhány olyan számítási modellt, amelyeknek a számítási ereje megegyezik a Turing gépek számítási erejével.

Ezen modellek Turing-ekvivalenciája a Church-Turing tézis alátámasztásául szolgálnak és így egyúttal, ha a Church-Turing tézist igaznak gondoljuk, az algoritmus fogalmának alternatív matematikai modelljeit adhatják.

0-típusú grammatikák és a TG-ek kapcsolata

Tétel

Minden G grammatikához megadható egy $L(G)$-t felismerő NTG.

Bizonyítás: Legyen M-nek 3 szalagja, az első a TG bemenetét, a második a G grammatika szabályait tartalmazza. Ezeket a működés során csak olvassuk.

A harmadik szalagon mindig egy α mondatforma áll (kezdetben G kezdőszimbóluma).

A Turing gép nemdeterminisztikusan választ egy $p \to q$ szabályt és α-ban egy pozíciót. Ha az adott pozícióban éppen p kezdődik, azaz $\alpha = xpy$, akkor p-t q-ra cseréli, az új mondatforma xqy lesz.

Ha az 1. és a 3. szalag tartalma megegyezik a gép q_i-ben megáll. M ezt minden iteráció előtt ellenőrzi. Így $L(M) = L(G)$. □

Következmény: Előző tételünk alapján determinisztikus TG is adható.

0-típusú grammatikák és a TG-ek kapcsolata

Tétel

Minden $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n \rangle$ determinisztikus TG-hez megadható egy $L(M)$-et generáló G grammatika.

Bizonyítás: G mondatformái M konfigurációit fogják kódolni. A G grammatika éppen fordítottan fog haladni. Nemdeterminisztikusan előállít egy elfogadó konfigurációt, majd ebből megpróbál egy kezdőkonfigurációt levezetni.

Legyen $G = \langle (\Gamma \backslash \Sigma) \cup Q \cup \{S, A, \rhd, \lhd\}, \Sigma, P, S \rangle$.
P szabályai:

1. $S \to \rhd A q_i A \lhd$
2. $A \to aA \mid \varepsilon$ $(\forall a \in \Gamma)$
3. $bq' \to qa$, ha $\delta(q, a) = (q', b, R)$
4. $q'b \to qa$, ha $\delta(q, a) = (q', b, S)$
5. $q'cb \to cqa$, ha $\delta(q, a) = (q', b, L)$ $(\forall c \in \Gamma)$
6. $\sqcup \lhd \to \lhd$, $\lhd \to \varepsilon$, $\rhd \sqcup \to \rhd$, $\rhd q_0 \to \varepsilon$

0-típusú grammatikák és a TG-ek kapcsolata

1. $S \to \rhd A q_i A \lhd$
2. $A \to aA \mid \varepsilon$ $\qquad (\forall a \in \Gamma)$
3. $bq' \to qa$, ha $\delta(q,a) = (q', b, R)$
4. $q'b \to qa$, ha $\delta(q,a) = (q', b, S)$
5. $q'cb \to cqa$, ha $\delta(q,a) = (q', b, L)$ $\qquad (\forall c \in \Gamma)$
6. $\sqcup \lhd \to \lhd$, $\quad \lhd \to \varepsilon$, $\quad \rhd \sqcup \to \rhd$, $\quad \rhd q_0 \to \varepsilon$

1-2. generálunk egy tetszőleges elfogadó konfigurációt

3-5. a konfigurációátmeneteket fordított irányban szimuláljuk. Pl. ha $\alpha cqa\beta \vdash \alpha q'cb\beta$ egy $\delta(q,a) = (q', b, L)$ szabály szerint, akkor most a grammatikában az 5-ös pont szerint $q'cb$ íródhat át cqa-ra.

6. Ha a mondatformánk egy kezdőkonfiguráció (esetleg néhány extra $\sqcup$-el), akkor ezek takarítják el a már felesleges jeleket.
A levezetés hosszára vonatkozó indukcióval megmutatható, hogy

$q_0 w \vdash^* \alpha q_i \beta$ a.cs.a. $S \Rightarrow^* \rhd \alpha q_i \beta \lhd \Rightarrow^* \rhd \sqcup^i q_0 w \sqcup^j \lhd \Rightarrow^* w$. $\square$

Többvermű gépek

- Egy k-vermű gép egy determinisztikus veremautomata k veremmel.
- Az inputszót a veremautomatához hasonlóan kapja. Van egy véges vezérlőegysége, amely egy véges halmazból veszi fel az állapotát.
- Adott egy minden verem számára közös veremábécé.
- A többvermű gép átmenete függ: (1) a vezérlőegység állapotától, (2) az olvasott inputszimbólumtól (3) az egyes vermek tetején lévő veremszimbólumtól.

Többvermű gépek

- A többvermű gép egy állapot-átmenet során: (1) állapotot vált, (2) az egyes vermek tetején lévő legfelső veremszimbólumot valahány (a 0-t is beleértve) veremszimbólum sorozatával helyettesíti, amely általában különböző az egyes vermek esetében.
- Egy tipikus konfiguráció- (állapot-)átmenet:
$\delta(q, a, X_1, \ldots, X_k) = (p, \gamma_1, \ldots, \gamma_k)$.

  q állapotban X_i szimbólum hatására az i-dik verem tetején, $i = 1, \ldots, k$, az automata az a inputszimbólumot (vagy valamilyen inputszimbólum, vagy ε) feldolgozza az inputból, p állapotba megy és X_i szimbólumot az i-edik verem tetején γ_i-re cseréli, minden $i = 1, \ldots, k$-ra).
- A többvermű gép a végállapotába jutva fogadja el az egyes szavakat.

A többvermű gépek számítási ereje

Feltesszük, hogy az input végén (nem része annak) van egy \$ speciális szimbólum, amelyet **végjelzőnek (endmarker)** is hívunk. A végjelző jelzi az input összes betűjének a feldolgozását.

Jelölje $\mathcal{L}_{k\mathrm{V}}$ a k-veremmel felismerható nyelvek osztályát. Nyilvánvalóan

$$\mathcal{L}_{0\mathrm{V}} \subseteq \mathcal{L}_{1\mathrm{V}} \subseteq \cdots \subseteq \mathcal{L}_{k\mathrm{V}} \subseteq \mathcal{L}_{(k+1)\mathrm{V}} \subseteq \cdots$$

Mivel a 0-vermek a véges automaták és az 1-vermek a determinisztikus veremautomaták, ezért

$$\mathcal{L}_{0\mathrm{V}} = \mathcal{L}_3, \quad \mathcal{L}_3 \subset \mathcal{L}_{1\mathrm{V}} \subset \mathcal{L}_2.$$

A többvermű gépek számítási ereje

Állítás

Minden $k \in \mathbb{N}$-re $\mathcal{L}_{kV} \subseteq \mathrm{RE}$

Bizonyítás: Könnyű szimulálni egy k-vermet egy $(k+1)$-szalagos TG-pel: az első szalagon tároljuk a bemenetet, a többin pedig az egyes vermek tartalmát. Ha egy lépésben k darab betűt írunk valamelyik verembe, azt k lépésben, betűnként szimulálhatjuk.

Tétel

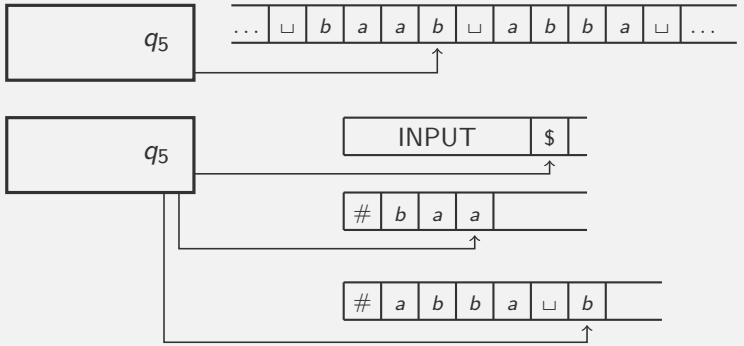
$\mathrm{RE} \subseteq \mathcal{L}_{2V}$.

Következmény

Minden $k \geqslant 2$-re $\mathcal{L}_{kV} = \mathrm{RE}$.

A többvermű gépek számítási ereje

A tétel bizonyítása:



Ötlet: Az egyik verem az író-olvasó fejtől balra, a másik az attól jobbra lévő szimbólumokat tárolja.

A többvermű gépek számítási ereje

- Kezdetben S vermei a # veremalja szimbólummal indulnak (Feltehető, hogy # nem eleme M szalagábécéjének.). # -t csak a verem aljának jelzésére használjuk.
- Tegyük fel, hogy S bemenetén ott van a w szó. Másolja S a w szót az első vermébe és fejezze be a másolást, ha az input végét jelző \$-t olvasta.
- S egyesével kiveszi a szimbólumokat az első veméből és belerakja az összeset a másodikba. (ε-átmenetekkel, innentől minden átmenet ilyen.) Ekkor az első verem üres lesz, a második fogja w -t tartalmazni, w első betűje lesz a verem tetején.
- S az M kezdőállapotának megfeleltetett állapotába lép.

A többvermű gépek számítási ereje

S M egy átmenetét a következőképpen szimulálja:

- Feltehető, hogy S ismeri M állapotát, amelyet jelöljünk q-val. Ehhez S-ben M minden állapotához legyen egy saját állapot a véges vezérlőegységében.
- S ismeri az M író-olvasó által éppen pásztázott X szimbólumot is: ez S második vermének a legfelső szimbóluma. Kivételt képez az az eset, amikor a második verem csak a veremalja szimbólumot tartalmazza, ilyenkor M épp egy üres szimbólumra ért.
- Tehát S ismeri M következő átmenetét. S vezérlőegysége M következő állapotának megfelelően kerül a saját, ennek megfelelő állapotába.

A többvermű gépek számítási ereje

1. eset:

Ha M az X szimbólumot Y -nal helyettesíti és a fej **jobbra mozog**, akkor S Y-t az első vermébe rakja, ami azt fejezi ki, hogy Y az M író-olvasó fejétől balra van, X -t pedig kivesszük S második veremének tetejéről. Előfordulhat az alábbi két kivételes eset.

- Ha a második verem csak a #-t tartalmazza (azaz $X = \sqcup$), akkor a második verem tartalma nem változik. (Hiszen M újabb üres cella felé mozdul jobbra.)
- Ha $Y = \sqcup$ ÉS az első verem tetején # áll, akkor az első verem tartalma változatlan marad. (M író-olvasó fejétől balra továbbra is minden cella üres.)

2. eset:

Ha M az X szimbólumot Y -nal helyettesíti és a fej **nem mozog**, akkor S Y-ra cseréli X -t a második verem tetején.

A többvermű gépek számítási ereje

3. eset:

Ha M az X szimbólumot Y -nal helyettesíti és a fej **balra mozog**, akkor S az első verem legfelső szimbólumát, nevezzük Z -nek, eltávolítja a verem tetejéről, és X -et YZ -vel helyettesíti a második veremben (Z lesz legfelül.). Ez azt jelenti, hogy ami egy pozícióval az író-olvasó fejétől balra volt, most az író-olvasó fej alatt lesz.

- Kivételt képez az az eset, amikor Z a veremalja szimbólum. Ekkor S X-et $Y\sqcup$-re cseréli a második veremben (azaz $\sqcup$ kerül a második verem tetejére), az elsőből pedig nem távolít el semmit.

Ezen felül S természetesen M állapotváltozásait lekövetve aktualizálja az állapotát.

S elfogad, ha M új állapota elfogadó, egyébként S M következő lépésének szimulálásával folytatja működését. □

Számláló gépek

A **számláló gépek (counter machines)** olyan matematikai gépek, amelyek véges sok (de előre rögzített számú) **regiszterrel** rendelkeznek, amelyek korlát nélküli méretű természetes számokat tárolhatnak. A számláló gépek tipikusan egy nagyon limitált **utasításkészlettel** működtethetők. Ezen utasításkészlet 1-2 aritmetikai továbbá 1-2 vezérlő utasításból áll.

A számláló gépek a **regiszter gépek** legegyszerűbb, speciális esete. Később találkozni fogunk majd a RAM-géppel is, amelyik szintén speciális regiszter gép, a számláló gép olyan általánosítása, ahol indirekt címzésű utasítások is rendelkezésre állnak.

Számláló gépek

A számláló gépek egy lehetséges reprezentációja egy szekvenciális programkód, ahol a programsorok címkézettek.

Az utasításkészlet alapján többfajta számláló gép ismeretes. Mi **Marvin Minsky** egy 1967-es modelljét tekintjük, ahol mindössze 3 utasítás létezik:

- INC(r): 1-gyel növeli az r regiszter értékét
- DEC(r): 1-gyel csökkenti az r regiszter értékét
- IF r regiszter értéke $= 0$ THEN GOTO z: ha a regiszter értéke 0, akkor a z címkéjű utasításra ugrik, különben a következő utasítással folytatja.

Matematikai gépként is reprezentálhatjuk. Ilyenkor a gép következő állapota függ az aktuális állapotától, az inputszimbólumtól, valamint attól, hogy valamely számlálója nulla-e vagy sem. Egy állapot-átmenet során a gép megváltoztathatja állapotát továbbá hozzáadhat vagy kivonhat 1-et bármely számlálójából, feltéve, hogy az nem 0-val egyenlő.

Számláló gépek k-verem reprezentációja

Egy lehetséges (valamivel általánosabb) gépként történő reprezentáció:

Definíció

A **számláló gép** (counter machine) egy korlátozott többvermű gép a következő megszorításokkal:

- Csak kétfajta veremszimbólum létezik: Z_0 (veremalja szimbólum) és X.
- Kezdetben egyetlen Z_0 van minden veremben.
- Z_0 -t csak Z_0X^i alakú szó helyettesítheti valamely $i \geqslant 0$-ra.
- X -t csak X^i helyettesítheti valamely $i \geqslant 0$-ra.

Ez azt jelenti, hogy Z_0-ból legfeljebb 1 lehet minden egyes veremben mégpedig a verem alján. Z_0-n kívül csak X-ek lehetnek a vermekben.

Így egy számláló 0 voltát úgy ellenőrizhetjük, hogy megnézzük Z_0 vagy X van-e a verem tetején.

A számláló gépek számítási ereje

Észrevételek:

- A számláló gép által elfogadott bármely nyelv rekurzívan felsorolható. Ez következik abból, az előző tételünkből, hogy többvermes gépeket lehet TG-pel szimulálni.
- Az egyszámlálós gép által elfogadott bármely nyelv környezetfüggetlen. A számlálót tekinthetjük úgy, mint egy vermet. Következésképpen a számlálógép speciális esete az egyvermű gépnek, a veremautomatának.

Tétel

Minden rekurzívan felsorolható nyelv felismerhető egy 3 számlálóval rendelkező számláló géppel.

A számláló gépek számítási ereje

Bizonyítás:
Elegendő egy tetszőleges 2-vermet 3-számlálós géppel szimulálni.

Tegyük fel, hogy a kétvermű gép (közös) veremábécéje $r-1$ elemű. Ezen szimbólumokat azonosíthatjuk 1 és $r-1$ közötti számjegyekkel, az $u = X_n \cdots X_1$ szóval leírt veremtartalmat (a verem tetején u utolsó betűje áll) pedig tekinthető egy r-es számrendszerbeli alakban felírt számnak. Azaz a verem tartalma (amelynek a teteje u utolsó betűje) dekódolhatóan reprezentálható egy $X_n r^{n-1} + X_{n-1} r^{n-2} + \cdots + X_2 r + X_1$ természetes számmal.

Az első két számláló tárolja tehát a két verem tartalmát, egy-egy természetes számként. A harmadik számláló egy segédszámláló. Arra szolgál, hogy a két másik számlálót igazítsa, amennyiben ez szükséges, pl. ha valamelyik számlálót r -rel szeretnénk osztani vagy szorozni.

A számláló gépek számítási ereje

A vermen végrehajtható három alapművelet a következő:

(1) a legfelső elem eltávolítása (pop),
(2) a legfelső elem átírása
(3) egy elemnek a verembe történő helyezése (push).

A k-verem műveletei ezen alapműveletek kompozíciói, így elegendő ezeket számlálógéppel szimulálni.

Például, ha a legfelső X szimbólumot k darab szimbólumból álló szóval szeretnénk helyettesíteni, akkor ezt a műveletet k részre bonthatjuk, X helyettesítésére és $k-1$ darab push műveletre.

Ezen alapműveletek végrehajtását a számláló segítségével a következőképpen oldhatjuk meg:

A számláló gépek számítási ereje

(1): pop művelet

A legfelső elemnek a veremből való eltávolítása megfelel a neki megfelelő számláló i értékének $\lfloor i/r \rfloor$ -rel való helyettesítésének és a maradék (X_1) eldobásának. Kezdjünk el 1-eket kivonni ebből a számlálóból. A 0 kezdeti értékű harmadik számlálóhoz minden r. kivonás után adjunk hozzá 1-et. Amikor a csökkenő számláló értéke 0 lesz, akkor a harmadik számlálóé éppen $\lfloor i/r \rfloor$. Ezután ismételten növeljük az eredeti számlálót 1-gyel és a harmadikat csökkentsük 1-gyel, amíg a harmadik számláló újra 0 nem lesz. Ekkor az a számláló, amelyik i-t tárolta előzőleg, $\lfloor i/r \rfloor$ lesz.

(2): a tetőelem megváltoztatása

Ahhoz, hogy X -et Y -ra cseréljük a verem tetején, amelyet i reprezentál a számoló gépnél, növelni vagy csökkenteni kell i-t egy r -nél nem nagyobb értékkel. Ha $Y > X$ akkor i-t $Y - X$ -szel növeljük. Ha $Y < X$, akkor i-t $X - Y$ -nal csökkentjük.

A számláló gépek számítási ereje

(3) push művelet

Ahhoz, hogy X -et azon verembe helyezzük, amely eredetileg i-t tartalmazza, i-t $ir + X$ -szel kell helyettesíteni. Először r -rel szorzunk. Ehhez folyamatosan csökkentjük i-t 1-gyel, és növeljük a harmadik számlálót (amely 0-ról indul) r -rel. Amikor az eredeti számláló értéke 0 lesz, akkor a harmadik számlálóé ir. A harmadik számlálót visszamásoljuk az eredetibe (lépésenként eggyel növelve az eredeti, és eggyel csökkentve a harmadik számláló értékét). Ekkor a harmadik számláló értéke újra 0 lesz. Végül az eredeti számlálót X -szel megnöveljük.

Inicializálás:

A konstrukció befejezéséhez szükséges még a számlálók inicializálása a vermek kezdeti állapotának szimulálásához, amikor még csak a kezdőszimbólumot tartalmazzák a vermek. Ez a lépés megoldható úgy, hogy a két számlálót valamilyen 1 és $r - 1$ közötti értékkel növeljük, amely megfelel a kezdőszimbólumnak. □

A számláló gépek számítási ereje

Tétel

Minden rekurzívan felsorolható nyelv elfogadható kétszámlálós géppel.

Bizonyítás:

Elegendő annak bizonyítása, hogyan szimuláljunk három számlálót kettővel. Az ötlet az, hogy a három számlálót egy egész számmal fogjuk reprezentálni. Jelöljük a három szóbanforgó számlálót i-vel, j-vel és k-val és legyen a választott egész szám $m = 2^i 3^j 5^k$.

Az egyik számláló ezt a számot fogja tárolni, míg egy másik szorozni vagy osztani fogja m-et valamelyik előbbi prímszámmal (2-vel, 3-mal, 5-tel).

A háromszámlálós gép szimulálása a következőképpen történik:

A számláló gépek számítási ereje

(1) Növeljük 1-gyel i-t, j-t vagy k-t.

i 1-gyel való növeléséhez, szorozzuk m-et 2-vel. Az előbb láttuk, hogyan kell egy számot valamely r konstanssal megszorozni egy második számlálót (most a másodikat) használva. Hasonlóan, j növeléséhez, szorozzuk m-et 3-mal, k növeléséhez pedig 5-tel.

(2) i, j vagy k valamelyike 0-val egyenlő-e?

Ahhoz, hogy megmondjuk, hogy $i = 0$-e, meg kell határoznunk, hogy vajon m osztható-e 2-vel. Másoljuk m-et a második számlálóba egyesével és használjuk a számlálógép állapotát annak megjegyzésére, hogy m-et páros vagy páratlan sokszor csökkentettük-e. Ha m-et páratlan sokszor csökkentettük, mikor 0-vá válik, akkor $i = 0$. Ekkor m-et visszaállítjuk a második számláló tartalmának az elsőbe másolásával. Hasonlóan tesztelhetjük, hogy $j = 0$-e (m osztható-e 3-mal), illetve, hogy $k = 0$-e (m osztható-e 5-tel).

A számláló gépek számítási ereje

(3) Csökkentsük 1-gyel i-t, j-t vagy k-t.

i, j illetve k csökkentéséhez osszuk m-et rendre 2-vel, 3-mal vagy 5-tel. Láttuk korábban hogyan kell az osztást végrehajtani egy második számlálót használva. Ha nem 0 az osztás maradéka, akkor elfogadás nélkül terminálunk mivel a számláló értéke 0 volt. Amennyiben sikerült maradék nélkül osztani, akkor éppen a megfelelő számláló 1-gyel való csökkentését szimuláltuk. □

Miért pont 2,3,5? Gondoljuk meg, miért fontos prímek (általánosabban: relatív prímek) használata!

Összegzés a számláló gépekről:

- A k számlálós számláló gépek számítási ereje $k \geqslant 2$-re megegyezik a TG-ek számítási erejével.
- Az egyetlen számlálóval rendelkező számláló gépek CF egy valódi részhalmazát alkotják. Például $\{a^n b^n \mid n \in \mathbb{N}\}$ felismerhető 1 számlálós géppel, de $\{a^n b^k a^k b^n \mid n, k \in \mathbb{N}\}$ nem.

Számítási modellek

4. előadás

A logika két modellje

Nulladrendű logika:

A logika **nulladrendű modelljében** a formulák ítéletváltozókból $X, Y, \ldots$ épülnek fel logikai műveletek $\neg, \wedge, \vee$ segítségével. Példa: $\neg(X \vee \neg(Y \wedge \neg Z))$.

A változókat **kiértékelhetjük** igazra/hamisra. Adott változókiértékelés mellett a formula szerkezete alapján rekurzíve kiszámítható a **formula Boole-értéke** ($\neg$: negáció, $\wedge$: logikai és, $\vee$: megengedő vagy).

Egy formula **kielégíthető**, ha a 2^n lehetséges változókiértékelés közül legalább egy esetben a formula Boole-értéke igaz, **kielégíthetetlen** különben (n a változók száma). φ **tautologikusan igaz**, ha minden változókiértékelés esetén igaz ($\models \varphi$). Egy Φ formulahalmaz **tautologikus következménye** φ, ha minden olyan változókiértékelésben, amiben Φ összes formulája igaz, igaz φ is ($\Phi \models \varphi$).

A logika két modellje

Ha X egy ítéletváltozó, X-et és $\neg X$-et **literálnak**. Literálok diszjunkcióját **elemi diszjunkciónak** vagy más néven **klóznak**. Ilyenek konjunkcióját **konjunktív normálformának** (KNF) nevezzük. Példa: $(\neg X \vee Y) \wedge (X \vee \neg Y \vee Z) \wedge \neg Z$.

Állítás: Minden formulához van vele ekvivalens KNF.

Elsőrendű logika:

Adott **függvényszimbólumok**, **predikátumszimbólumok** és **konstansok** egy-egy véges halmaza. Az előbbiek az **aritásukkal** együtt (hány változósak). Továbbá legyenek $x, y, \ldots$ ún. **individuumváltozók**.

A **termek** konstansokból, változókból és függvényjelekből épülnek fel figyelembe véve az aritást. Például $g(f(x, a), y)$, itt a konstans, f, g 2 aritású függvényjelek.

Az **atomi formula** egyetlen predikátumszimbólum aritásnyi term argumentummal. A **formulák** atomi formulákból épülnek fel $\neg, \wedge, \vee, \exists x, \forall x$ segítségével.

A logika két modellje

Példa: $P(x, f(x)) \vee \exists y Q(y)$, itt P 2 aritású, Q 1 aritású predikátumszimbólum, f 1 aritású függvényjel.

A **formulák Boole-értékének** meghatározásához először keresünk egy matematikai struktúrát: egy U alaphalmazt, majd ezen a függvényszimbólumoknak illetve predikátumszimbólumoknak aritásuk szerint függvényeket illetve relációkat feleltetünk meg U-n. A konstansoknak szintén megfeleltetünk 1-1 U-beli elemet (interpretáció). Majd a változóknak U-beli értéket adunk (változókiértékelés).

Így a termeknek lesz U-beli értékük. Egy n argumentumú atomi formula akkor legyen igaz, ha a termeiből álló érték n-es a predikátumszimólumnak megfeleltetett relációban áll. Innen$\neg, \wedge, \vee$: szokásos, $\exists x$: létezik U-beli elem..., $\forall x$: minden U-beli elemre ...

Egy formula **kielégíthető**, ha van olyan interpretáció és változókiértékelés melyre a formula Boole-értéke igaz. φ **logikailag igaz**, ha minden interpretáció és változókiértékelés esetén igaz. **Logikai következmény:** mint nulladrendben.

A Turing gépek egy elkódolása

Feltehető, hogy $\Sigma = \{0, 1\}$.

Egy M Turing-gép **kódja** (jelölése $\langle M \rangle$) a következő:
Legyen $M = (Q, \{0,1\}, \Gamma, \delta, q_0, q_i, q_n)$, ahol

- $Q = \{p_1, \ldots, p_k\}$, $\Gamma = \{X_1, \ldots, X_m\}$, $D_1 = R$, $D_2 = S$, $D_3 = L$
- $k \geqslant 3$, $p_1 = q_0$, $p_{k-1} = q_i$, $p_k = q_n$,
- $m \geqslant 3$, $X_1 = 0$, $X_2 = 1$, $X_3 = \sqcup$.
- Egy $\delta(p_i, X_j) = (p_r, X_s, D_t)$ átmenet kódja $0^i 1 0^j 1 0^r 1 0^s 1 0^t$.
- $\langle M \rangle$ az átmenetek kódjainak felsorolása 11-el elválasztva.

Észrevétel: $\langle M \rangle$ 0-val kezdődik és végződik, nem tartalmaz 3 darab 1-t egymás után.

$\langle M, w \rangle := \langle M \rangle 111 w$

Létezik nem Turing-felismerhető nyelv

Jelölés: Minden $i \geqslant 1$-re,

- jelölje w_i a $\{0,1\}^*$ halmaz i-ik elemét a hossz-lexikografikus rendezés szerint.
- jelölje M_i a w_i által kódolt TG-t (ha w_i nem kódol TG-t, akkor M_i egy tetszőleges olyan TG, ami nem fogad el semmit)

Tétel

Létezik nem Turing-felismerhető nyelv.

Bizonyítás: Két különböző nyelvet nem ismerhet fel ugyanaz a TG. A TG-ek számossága megszámlálható (a fenti kódolás injekció $\{0,1\}^*$-ba, ami megszámlálható). Másrészt viszont a $\{0,1\}$ feletti nyelvek számossága continuum. □

Az **átlós nyelv**: $L_d = \{w_i \mid w_i \notin L(M_i)\}$

Tétel

$L_d \notin RE$.

R és $\mathcal{L}_1$ viszonya

Definíció

A **lineárisan korlátolt automata** (LKA) olyan **nemdeterminisztikus** TG, melynek Σ bemeneti ábécéje két speciális szimbólumot tartalmaz $\rhd$-et (baloldali végejel/endmarker) és $\lhd$-et (jobboldali végejel/endmarkert). Ezen felül

- a bemenetek $\rhd(\Sigma \backslash \{\rhd, \lhd\})^* \lhd$-beliek,
- $\rhd$ és $\lhd$ nem írhatók felül
- $\rhd$-tól balra illetve $\lhd$-től jobbra nem állhat a fej.
- a fej kezdőpozíciója a $\rhd$ tartalmú cella jobb-szomszédja

Magyarán olyan NTG, amely korlátos munkaterülettel rendelkezik.

Nevét egy vele ekvivalens modellről kapta, amelyben a rendelkezésre álló tár az input hosszának konstansszorosa (lineáris függvénye).

R és $\mathcal{L}_1$ viszonya

Tétel

(1) Minden G 1-es típusú grammatikához megadható egy A LKA, melyre $L(A) = L(G)$.

(2) Minden A LKA-hoz megadható egy G 1-es típusú grammatika, melyre $L(G) = L(A)$.

Bizonyítás (vázlat):

(1) Az előző előadáson láttuk, hogy minden 0. típusú grammatikához lehet konstruálni $L(G)$-t felismerő NTG-t.
A konstrukció a 3. szalagján nemdeterminisztikusan szimulált egy G-beli levezetést, az iterációk végén ellenőrizte, hogy az 1. és 3. szalag tartalma megegyezik-e.

Amennyiben G 1-es típusú, azaz hossz-nemcsökkentőek a szabályai, akkor a 3. szalagon lévő mondatforma hossza sose haladhatja meg $|u|$-t, így ez az NTG egy LKA.

R és $\mathcal{L}_1$ viszonya

(2) Az előző előadás konstrukciójának (eléggé technikai jellegű) kis módosításával elérhető, hogy az LKA-hoz (ott NTG-hez) készített grammatika hossz-nemcsökkentő legyen, ehhez viszont ismert, hogy $\exists$ vele ekvivalens 1-típusú grammatika. □

Tétel

Ha A LKA, akkor $L(A)$ eldönthető.

Bizonyítás: A lineáris korlátoltság miatt A lehetséges konfigurációinak száma egy u bemenetre legfeljebb $m(u) = |Q| \cdot |u| \cdot |\Gamma|^{|u|}$, ahol Q az A állapothalmaza és Γ a szalagábécéje. Ha A-nak van elfogadó számítása, akkor van legfeljebb $m(u)$ hosszú elfogadó számítása is (a számítások két azonos konfiguráció közötti része kihagyható).

Működjön az M Turing gép pontosan úgy, mint A, de minden u bemenetre számolja a lépéseit $m(u)$-ig. Ekkor állítsuk le M-et q_n-ben. Nyilván $L(M) = L(A)$ és M minden bemenetre megáll. □

R és $\mathcal{L}_1$ viszonya

Tétel

$\mathcal{L}_1 \subset \mathrm{R}$.

Bizonyítás: Az előző tételek miatt $\mathcal{L}_1 \subseteq \mathrm{R}$.

Legyen $L_{d,\mathrm{LKA}} = \{\langle M\rangle \mid M \text{ LKA és } \langle M\rangle \notin L(M)\}$.

- $L_{d,\mathrm{LKA}}$ eldönthető.

  Egy S TG ugyanis egy M LKA bemenetére menjen q_i-be, ha $\langle M\rangle \notin L(M)$ illetve menjen q_n-be, ha $\langle M\rangle \in L(M)$. Mivel $L(M)$ eldönthető ezért S mindig terminál.
- $L_{d,\mathrm{LKA}}$ felismerhetetlen LKA-val ($\Rightarrow \notin \mathcal{L}_1$)

  (Cantor féle átlós módszerrel)

  Tegyük fel, indirekt, hogy $L_{d,\mathrm{LKA}}$-t egy S LKA felismeri.
  - ha $\langle S\rangle \in L_{d,\mathrm{LKA}} = L(S)$, akkor S felismeri $\langle S\rangle$-et, így $\langle S\rangle \notin L_{d,\mathrm{LKA}}$, ellentmondás,
  - ha $\langle S\rangle \notin L_{d,\mathrm{LKA}} = L(S)$, akkor S nem ismeri fel $\langle S\rangle$-et, így $\langle S\rangle \in L_{d,\mathrm{LKA}}$, ellentmondás. □

Számítási feladatok megoldása TG-pel

Az eldöntési problémák általánosításai a (ki)számítási problémák. Ilyenkor a kiszámítandó f függvény igen/nem helyett más értékeket is adhat eredményül. Ezúttal is algoritmikus megoldást keresünk.

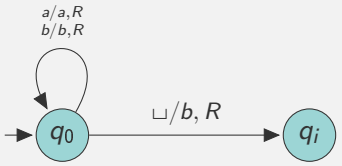
Legyen $\mathrm{Dom}(f) = \Sigma^*$, $\mathrm{Ran}\,(f) \subseteq \Delta^*$ valamely Σ, Δ ábécékre.

Definíció

Azt mondjuk, hogy az $M = \langle Q, \Sigma, \Delta, \delta, q_0, q_i, (q_n)\rangle$ TG **kiszámítja** az $f : \Sigma^* \to \Delta^*$ szófüggvényt, ha minden $u \in \Sigma^*$-beli szóra megáll, és ekkor $f(u) \in \Delta^*$ olvasható az utolsó szalagján.

Megjegyzés: A definíció értelmében nincs szükség q_i és q_n megkülönböztetésére, elég lenne egyetlen megállási állapot. [Ezért van q_n ()-ben.] Az ilyen TG-eket **számító Turing gépnek** nevezzük.

Példa:
$f(u) = ub$
$(u \in \{a, b\}^*)$.



Visszavezetés

Definíció

Az $f : \Sigma^* \to \Delta^*$ szófüggvény **kiszámítható**, ha van olyan Turing-gép, ami kiszámítja.

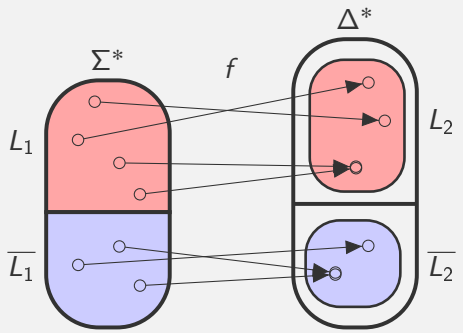
Definíció

$L_1 \subseteq \Sigma^*$ **visszavezethető** $L_2 \subseteq \Delta^*$-ra, ha van olyan $f : \Sigma^* \to \Delta^*$ kiszámítható szófüggvény, hogy $w \in L_1 \Leftrightarrow f(w) \in L_2$. Jelölés: $L_1 \leqslant L_2$.

Megjegyzés: A fogalom Emil Posttól származik, angol szakirodalomban: many-one reducibility.

Visszavezetés

$L_1 \leqslant L_2$



(1) f az egész Σ^*-on értelmezett, (2) f kiszámítható, (3) $f(L_1) \subseteq L_2$ valamint (4) $f(\overline{L}_1) \subseteq \overline{L}_2$.

f nem kell hogy injektív legyen és az se, hogy szürjektív.

Visszavezetés

Tétel

- Ha $L_1 \leqslant L_2$ és $L_2 \in RE$, akkor $L_1 \in RE$.
- Ha $L_1 \leqslant L_2$ és $L_2 \in R$, akkor $L_1 \in R$.

Következmény

- Ha $L_1 \leqslant L_2$ és $L_1 \notin RE$, akkor $L_2 \notin RE$.
- Ha $L_1 \leqslant L_2$ és $L_1 \notin R$, akkor $L_2 \notin R$.

R és RE

Univerzális nyelv: $L_u = \{\langle M, w\rangle \mid w \in L(M)\}$.

Tétel

$L_u \in$ RE\R

Jelölés: Ha $L \subseteq \Sigma^*$, akkor jelölje $\bar{L} = \{u \in \Sigma^* \mid u \notin L\}$.

Tétel

Ha L és $\bar{L} \in$ RE, akkor $L \in$ R.

Következmény: RE nem zárt a komplementer-képzésre.

Tétel

Ha $L \in$ R, akkor $\bar{L} \in$ R.

Megállási probléma: $L_{\text{halt}} = \{\langle M, w\rangle \mid M$ megáll w-n$\}$.

Tétel

$L_{\text{halt}} \in$ RE\R.

Eldönthetetlen problémák

Grammatikák:

Tétel

Eldönthetetlenek az alábbi CF nyelvtanokkal kapcsolatos kérdések. Legyen G_1 és G_2 két CF nyelvtan.

- $L(G_1) \cap L(G_2) \neq \varnothing$
- $L(G_1) = \Gamma^*$ valamely Γ-ra
- $L(G_1) = L(G_2)$
- $L(G_1) \subseteq L(G_2)$

Megjegyzés: Reguláris nyelvekre ezek a kérdések eldönthetők.

Megjegyzés: $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ esetén is eldönthető a szóprobléma.

Input: G és $u \in T^*$. Output: $u \stackrel{?}{\in} L(G)$.

Eldönthetetlen problémák

Logikai kérdések:

Tétel

Eldönthetetlen, hogy φ elsőrendű logikai formulára és Φ formulahalmazra

(1) $\models \varphi$ teljesül-e (φ logikailag igaz-e).
(2) φ kielégíthetetlen-e
(3) φ kielégíthető-e
(4) $\Phi \models \varphi$ teljesül-e

Eldönthető, hogy φ nulladrendű logikai formulára és Φ formulahalmazra

(1) $\models \varphi$ teljesül-e (φ tautologikusan igaz-e).
(2) φ kielégíthetetlen-e
(3) φ kielégíthető-e
(4) $\Phi \models \varphi$ teljesül-e

Bonyolultságelmélet – időbonyolultsági osztályok

A továbbiakban eldönthető problémákkal foglalkozunk, ilyenkor a kérdés az, hogy milyen hatékonyan dönthető el az adott probléma.

- $\mathrm{TIME}(f(n)) =$ $\{L \mid L$ eldönthető $O(f(n))$ időigényű determinisztikus TG-pel$\}$
- $\mathrm{NTIME}(f(n)) =$ $\{L \mid L$ eldönthető $O(f(n))$ időigényű nemdeterminisztikus TG-pel$\}$
- $\mathrm{P} = \bigcup_{k \geqslant 1} \mathrm{TIME}(n^k)$.
- $\mathrm{NP} = \bigcup_{k \geqslant 1} \mathrm{NTIME}(n^k)$.
- Észrevétel: $\mathrm{P} \subseteq \mathrm{NP}$.
- Sejtés: $\mathrm{P} \neq \mathrm{NP}$ (sejtjük, hogy igaz, de bizonyítani nem tudjuk).
- A $\mathrm{P} \neq \mathrm{NP}$ sejtés a Clay Matematikai Intézet által 2000-ben nyilvánosságra hozott 7 Milleneumi Probléma egyike. Igazolásáért vagy cáfolatáért az Intézet 1M$-t fizet.

Milyenek az NP-beli problémák?

P-re úgy gondolunk, hogy ez az osztály tartalmazza a hatékonyan megoldható problémákat. (Ami nem teljesen igaz.)

Milyen problémákat tartalmaz NP?

Egy L NP-beli problémához definíció szerint létezik őt polinom időben eldöntő NTG ami gyakorlatilag a következőképpen működik:

a probléma minden bemenetére próbál polinom időben **„megsejteni”** (értsd: nemdeterminisztikusan előllítani) egy kis méretű „tanút”, ami azt bizonyítja, hogy a bemenet egy igen-példány.

Precíz tétellé is tehető, miszerint akkor és csak akkor NP-beli egy eldöntési probléma, ha minden igen-inputhoz megadható **polinom méretű és polinom időben ellenőrizhető tanú** (azaz, ami igazolja, hogy ő valóban igen-input).

Polinom idejű visszavezetés

Definíció

Az $f : \Sigma^* \to \Delta^*$ szófüggvény **polinom időben kiszámítható**, ha van olyan Turing-gép, ami polinom időben kiszámítja.

Definíció

$L_1 \subseteq \Sigma^*$ **polinom időben visszavezethető** $L_2 \subseteq \Delta^*$-ra, ha van olyan $f : \Sigma^* \to \Delta^*$ polinom időben kiszámítható szófüggvény, hogy $w \in L_1 \Leftrightarrow f(w) \in L_2$. Jelölés: $L_1 \leqslant_p L_2$.

Megjegyzés: A polinom idejű visszavezetést Richard Karpról elnevezve *Karp-visszavezetésnek* vagy *Karp-redukciónak* is nevezik. Angolul: polynomial-time many-one reduction vagy Karp reduction.

$\mathcal{C}$-teljesség

Intuitíve, ha egy problémára visszavezetünk egy másikat, az azt jelenti, hogy az a probléma legalább olyan nehéz, mint amit visszavezettünk rá. Azaz ebben az értelemben a legnehezebb problémák azok, melyekre minden probléma visszavezethető.

Definíció

Legyen $\mathcal{C}$ egy bonyolultsági osztály. Egy L nyelv **$\mathcal{C}$-nehéz** (a polinom idejű visszavezetésre nézve), ha minden $L' \in \mathcal{C}$ esetén $L' \leqslant_p L$.

Definíció

Legyen $\mathcal{C}$ egy bonyolultsági osztály. Egy L nyelv **$\mathcal{C}$-teljes**, ha $L \in \mathcal{C}$ és L $\mathcal{C}$-nehéz.

NP-teljesség

Ha speciálisan $\mathcal{C}$=NP:

Definíció

Egy L nyelv **NP-teljes** (a polinom idejű visszavezetésre nézve), ha

- $L \in$ NP
- L NP-nehéz, azaz minden $L' \in$ NP esetén $L' \leqslant_p L$.

Megjegyzés: Néha úgy fogalmazunk, hogy az L (eldöntési) *probléma* NP-teljes...

Tétel

Legyen L egy NP-teljes probléma. Ha $L \in$ P, akkor P = NP.

NP-teljesség és a P$\stackrel{?}{=}$NP probléma

Intuitive az NP-teljes problémák az NP-beli problémák legnehezebbjei.

Az előző tétel szerint tehát, ha valaki talál egy NP-teljes problémára polinom idejű determinisztikus algoritmust, azzal bizonyítja, hogy P=NP.

Mivel nem tudjuk, hogy P$\stackrel{?}{=}$NP (azt sejtjük, hogy nem igaz!), ezért nyilván egyetlen NP-teljes problémára sem ismeretes polinomiális idejű determinisztikus algoritmus és amennyiben a sejtésünk igaz, ilyet nem is fogunk találni.

Így az NP-teljes problémákra úgy tekinthetünk, mint eldönthető, de **hatékonyan nem eldönthető** problémákra.

Cook-Levin tétel

SAT= $\{\langle\varphi\rangle \mid \varphi$ kielégíthető nulladrendű KNF$\}$

Tétel (Cook-Levin)

SAT NP-teljes.

Tétel

Ha L NP-teljes, $L \leqslant_p L'$ és $L' \in$ NP, akkor L' NP-teljes.

Tehát a polinom idejű visszavezetés fogalmának segítségével további NP-beli nyelvek NP-teljessége bizonyítható.

A következő problémák NP-teljesség ezen tétel alapján bizonyítható.

kSAT

Egy minden tagjában pontosan k különböző literált tartalmazó konjunktív normál formát (KNF-et) **kKNF-nek** nevezünk ($k \geqslant 1$).

Példák: 4KNF:

$(\neg X_1 \vee X_3 \vee X_5 \vee \neg X_6) \wedge (\neg X_1 \vee \neg X_3 \vee X_4 \vee \neg X_6) \wedge (X_1 \vee X_2 \vee \neg X_4 \vee \neg X_6)$.

2KNF: $(\neg X_1 \vee X_3) \wedge (\neg X_1 \vee \neg X_3) \wedge (X_1 \vee X_2) \wedge (\neg X_2 \vee X_3)$.

$k\text{SAT} := \{\langle \varphi \rangle \mid \varphi \text{ kielégíthető } k\text{KNF}\}$.

Tétel

3SAT NP-teljes.

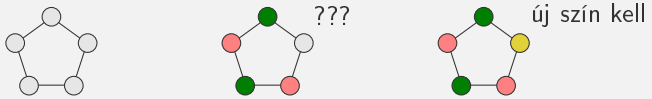
Megjegyzés: $2\text{SAT} \in \text{P}$.

3 színezhetőség

Definíció

Legyen $k \geqslant 1$ egész szám. Egy gráf **k-színezhető**, ha csúcsai k színnel színezhetők úgy, hogy a szomszédos csúcsok színei különbözőek.

Példa: Egy 5 csúcsból álló kör 3-színezhető, de nem 2-színezhető.



$k\text{Színezés} = \{\langle G \rangle \mid G \text{ } k\text{-színezhető}\}$.

Tétel

3Színezés NP-teljes.

Megjegyzés: $2\text{Színezés} \in \text{P}$.

Klikk, független ponthalmaz

Definíció

A G egyszerű, irányítatlan gráf egy teljes részgráfját **klikknek**, egy üres részgráfját **független ponthalmaznak** mondjuk.

Legyen $S \subseteq V(G)$ és $E \in E(G)$. Ha $S \cap E \neq \varnothing$, akkor a csúcshalmaz **lefogja** E-t. Ha S minden $E \in E(G)$ élt lefog, akkor S egy **lefogó ponthalmaz**.

Klikk$= \{\langle G, k \rangle \mid G$-nek van k méretű klikkje$\}$

Független ponthalmaz$=$
$\{\langle G, k \rangle \mid G$-nek van k méretű független ponthalmaza$\}$

Lefogó ponthalmaz$=$
$\{\langle G, k \rangle \mid G$-nek van k méretű lefogó ponthalmaza$\}$

Tétel

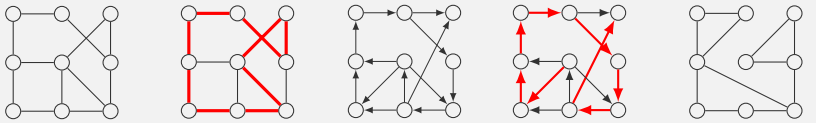
Független ponthalmaz,Klikk, Lefogó ponthalmaz NP-teljes.

Irányítatlan/irányított Hamilton út/kör

Definíció

Adott egy G gráf. Egy a G összes csúcsát pontosan egyszer tartalmazó utat **Hamilton útnak**, egy a G összes csúcsát pontosan egyszer tartalmazó kört **Hamilton körnek** nevezünk. Ha a gráf irányított, a Hamilton útnak/körnek irányítottnak kell lennie.

Jelölés: H-út/ H-kör Hamilton út/ Hamilton kör helyett.



HÚ$= \{\langle G, s, t \rangle \mid$ van a G irányított gráfban s-ből t-be H-út$\}$.

IHÚ$= \{\langle G, s, t \rangle \mid$ van a G irányítatlan gráfban s-ből t-be H-út$\}$.

IHK$= \{\langle G \rangle \mid$ van a G irányítatlan gráfban H-kör$\}$.

Hamilton út problémák NP teljessége

Tétel

HÚ, IHÚ, IHK NP-teljes

Az utazóügynök probléma:

Számítási (optimalizálási) verzió: Adott egy G élsúlyozott irányítatlan gráf nemnegatív élsúlyokkal. Határozzuk meg a legkisebb összsúlyú H-kört (ha van).

Eldöntési verzió:
TSP=$\{\langle G, K\rangle \mid G$-ben van $\leqslant K$ súlyú H-kör$\}$.

Tétel

TSP NP-teljes

További NP-teljes problémák

Lineáris diophantoszi egyenlőtlenségrendszer=
$\{\langle \mathbf{A}, \mathbf{b}\rangle \mid \mathbf{Ax} \leqslant \mathbf{b}$ egészegyütthatós egyenlőtlenségrendszernek van egész megoldása$\}$.

Részletösszeg:= $\{\langle S, K\rangle \mid S$ egész számok egy halmaza, $K \in \mathbb{Z}$, van S-nek egy olyan S' részhalmaza, hogy az S'-beli számok összege $K\}$

Hátizsák:= $\{\langle a_1, \ldots, a_n, b, p_1, \ldots p_n, k\rangle \in (\mathbb{R}^+)^{2n+2} \mid$
$\exists\, I \subseteq \{1, \ldots n\}$, amelyre $\sum_{i\in I} a_i \leqslant b$ és $\sum_{i\in I} p_i \geqslant k\}$.

Partíció:=$\{\langle B\rangle \mid B$ olyan pozitív számok multihalmaza, amely két egyenlő összegű részre particionálható$\}$.

Ládapakolás:=$\{\langle s_1, \ldots, s_n, k\rangle \mid s_i \in \mathbb{Q}^+ (1 \leqslant i \leqslant n)$ súlyok particionálhatók $k \in \mathbb{N}^+$ részre úgy, hogy minden partícióban a súlyok összege $\leqslant 1\}$.

NP lehetséges szerkezete

NP-köztes nyelv

L NP-köztes, ha $L \in$ NP, $L \notin$ P és L nem NP-teljes.

Ladner tétele

Ha P $\neq$ NP, akkor létezik NP-köztes nyelv.

Az alábbi problémáknak se a P-belisége, se NP-nehézsége nem ismeretes (így NP-köztes jelöltek):

- Gráfizomorfizmus=$\{\langle G_1, G_2\rangle \mid$ G_1 és G_2 irányítatlan izomorf gráfok$\}$.
- Prímfaktorizáció: adjuk meg egy egész szám prímtényezős felbontását [számítási feladat],
- Kapuminimalizálás: adott digitális áramkört minél kevesebb logikai kapuval megvalósítani [számítási feladat].

Tárbonyolultság – Az offline Turing gép

A tárbonyolultság mérésének problémája:

Első megközelítésben a tárigény a működés során felhasznált (vagyis a fejek által meglátogatott) cellák száma.

Probléma: Hiába "takarékoskodik" a felhasznált cellákkal a gép, az input hossza így mindig alsó korlát lesz a tárigényre.

Definíció

Az **offline Turing gép** (OTG) egy olyan TG, melynek az első szalagja csak olvasható, a többi írható is. Első szalagját bemeneti szalagnak, további szalagjait munkaszalagoknak nevezzük.
A **nemdeterminisztikus offline Turing gép** (NOTG) ugyanilyen, csak a gép nemdeterminisztikus.

Állítás

Minden TG-hez megadható vele ekvivalens offline TG.

Az offline Turing gépek tárigénye

Definíció

A **számító offline Turing gép** olyan legalább 2 szalagos számító TG, melynek az első szalagja csak olvasható, az utolsó szalagja csak írható. Első szalagját bemeneti, utolsó szalagját kimeneti, többi szalagját munkaszalagnak nevezzük.

Definíció

Egy OTG **többlet tárigénye** egy adott inputra azon celláknak a száma, melyeken a működés során valamelyik munkaszalag feje járt.

Definíció

Egy offline TG $f(n)$ **többlet tárkorlátos**, ha bármely u inputra legfeljebb $f(|u|)$ az többlet tárigénye.

Számító OTG-re hasonlóan. Nemdeterminisztikus OTG-re (NOTG) értelmszerűen módosítva.

Determinisztikus és nemdeterminisztikus tárbonyolultsági osztályok

Így az offline TG-pel **szublineáris** (lineáris alatti) tárbonyolultságot is mérhetünk.

- $\mathrm{SPACE}(f(n)) := \{L \mid L$ eldönthető $O(f(n))$ többlet tárkorlátos determinisztikus offline TG-pel$\}$
- $\mathrm{NSPACE}(f(n)) :=$ $\{L \mid L$ eldönthető $O(f(n))$ többlet tárkorlátos nemdeterminisztikus offline TG-pel$\}$
- $\mathrm{PSPACE} := \bigcup_{k\geqslant 1} \mathrm{SPACE}(n^k)$.
- $\mathrm{NPSPACE} := \bigcup_{k\geqslant 1} \mathrm{NSPACE}(n^k)$.
- $\mathrm{L} := \mathrm{SPACE}(\log n)$.
- $\mathrm{NL} := \mathrm{NSPACE}(\log n)$.

Savitch tétele és a hierarchia tétel

Tétel (Savitch)

Ha $f(n) \geqslant \log n$, akkor $\mathrm{NSPACE}(f(n)) \subseteq \mathrm{SPACE}(f^2(n))$.

Következmény

$\mathrm{PSPACE} = \mathrm{NPSPACE}$

$\mathrm{EXPTIME} := \bigcup_{k\in\mathbb{N}} \mathrm{TIME}(k^n)$.

Hierarchia tétel

$\mathrm{NL} \subset \mathrm{PSPACE}$ és $\mathrm{P} \subset \mathrm{EXPTIME}$.

$\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{NPSPACE} = \mathrm{PSPACE} \subseteq \mathrm{EXPTIME}$

Sejtés: A fenti tartalmazási lánc minden tartalmazása valódi.

Számítási problémák közelítő megoldásai

Jelölje egy kiszámítási (optimalizálási) problémában OPT az optimális értéket (minimumot/maximumot).

Definíció

Egy algoritmust α**-közelítőnek** hívunk, ha minden inputra az algoritmus kimenete megengedett és a visszaadott érték OPT-nak

- minimumkeresési feladat esetén legfeljebb α-szorosa,
- maximumkeresési feladat esetén legalább $1/\alpha$-szorosa,

Megjegyzés: α nem feltétlenül konstans, lehet az input n hosszának egy függvénye is.

Példa: Irányított gráfban maximális aciklikus részgráf keresése.

Rendezzük sorba a csúcsokat. A sorrend szerint haladva, minden csúcsra vizsgáljuk meg, hogy előre-élből vagy hátra-élből van-e több, a kisebbséghez tartozó éleket dobjuk el. A kapott gráf aciklikus és az élek legalább felét tartalmazza, így az algoritmus 2-közelítő. (Itt a megengedett kimenetek az aciklikus részgráfok.)

Számítási problémák közelítő megoldásai

Különösen érdekesek az NP-nehéz kiszámítási problémák (eldöntési verziójuk NP-nehéz), ilyenkor ugyanis nem ismeretes hatékony egzakt megoldás. A közelítő algoritmustól ilyenkor elvárhatjuk, hogy az viszont hatékony (polinomiális) legyen.

Példa: Minimális méretű lefogó ponthalmaz keresése egy G irányítatlan gráfban. A probléma NP-nehéz. Jelölje $\tau(G) = \min\{|S| \mid S \text{ lefogó ponthalmaz } G\text{-ben}\}$.

Megengedett válasz: egy lefogó ponthalmaz.

Mohón vegyük sorban minden, az adott pillanatig fedetlen él mindkét végpontját S-hez, amíg van fedetlen él. Az algoritmus során talált fedetlen élek diszjunktak, tehát $|S|/2$ csúcsra szükség van már csak az ő lefogásukhoz is. Így $\tau(G) \geqslant |S|/2$, tehát találtunk egy legfeljebb $2\tau(G)$ méretű lefogó ponthalmazt.

Az algoritmus tehát 2-közelítő és hatékonysága $O(|V(G)| + |E(G)|)$.

Számítási problémák közelítő megoldásai

Állítás

Ha P$\neq$ NP, akkor TSP-re semmilyen $g(n)$ függvény esetén se létezik polinom idejű $g(n)$-approximáció. (Megengedett válaszok: az ügynök egy körútja.)

Bizonyítás: Ha létezne ilyen, akkor polinom időben megoldhatnánk a Hamilton kör problémát a következőképpen. A Hamilton kör probléma egy tetszőleges G bemeneti gráfjához elkészítjük a TSP egy G' bemeneti gráfját a következőképpen. Legyen $n = |V(G)|$. G' egy teljes, élsúlyozott gráf szintén n csúcson. A G éleinek megfelelő G'-beli élek súlya legyen 1, míg a G nem-éleinek megfelelő G'-beli élek súlya legyen $ng(n)$. G'-t G-ből polinom időben elkészíthetjük. Ha G-ben volt Hamilton-kör, akkor G'-ben van n összsúlyú körút. Ha nem volt, akkor viszont minden körút legalább $ng(n) + n - 1$ súlyú.

Hívjuk meg most a polinom idejű approximációs algoritmust.

Számítási problémák közelítő megoldásai

Ha a válasz $\leqslant ng(n)$, akkor a minimáls összsúlyú körút csupa 1 súlyú élekből áll, így G-ben van Hamilton kör. Ha ennél nagyobb érteket kapunk, akkor a $g(n)$-approximáció miatt n-nél hosszabb az optimális körút, így G-ben nincs Hamilton kör. Így ez az algoritmus polinom időben eldönti IHK-t, amiből IHK NP-teljessége miatt P=NP következik, ami ellentmond a feltételeknek. □

TSP tehát egy rosszul közelíthető probléma. Az alábbi változat viszont jól közelíthető.

Metrikus utazó ügynök probléma: Ugyanaz, mint a TSP, de az élsúlyokra teljesül a háromszög egyenlőtlenség, azaz

$$\forall\{a,b\},\{b,c\},\{a,c\} \in E(G)\text{-re} \quad w(a,c) \leqslant w(a,b) + w(b,c),$$

ahol w a $G = (V, E)$ irányítatlan gráf élsúlyozása.

Állítás

A metrikus utazóügynök probléma polinom időben 2-approximálható.

Számítási problémák közelítő megoldásai

Bizonyítás: Legyen G egy bemenet. Készítsünk el G egy T minimális összsúlyú feszítőfáját. Járjuk be T-t mélységi bejárással. Az így kapott körséta a fa minden élét kétszer látogatja meg és egy minden csúcsot legalább egyszer meglátogat.

T összsúlyánál nyilván nem $\exists$ kisebb összsúlyú minden csúcsot lefedő összefüggő részgráf, így az ügynök minden körútja is legalább ilyen hosszú. Tehát a kapott körséta hossza legfeljebb 2OPT.

A háromszög egyenlőtlenség miatt nem nő a körséta hossza, ha a körséta egy csúcsát kihagyjuk, azaz a két szomszédja között éllel helyettesítjük a csúcs meglátogatásához szükséges 2 élt.

Alkalmazzuk ezt a rövidítést a körséta minden ismétlődő csúcsára, így végül egy Hamilton kört kapunk, melynek összsúlya legfeljebb 2OPT.

Az algoritmus műveletigénye $O(|E(G)|)$.

Megjegyzés: Ismeretes (3/2)-közelítő algoritmus is (Christofides).

Számítási modellek

5. előadás

Boole függvények

Legyenek X és Y logikai változók (0 vagy 1 értékűek). A legfontosabb legfeljebb kétváltozós logikai műveletek:

X	Y	$\neg X$	$X \wedge Y$	$X \vee Y$	$X \to Y$	$X \otimes Y$
1	1	0	1	1	1	0
1	0	0	0	1	0	1
0	1	1	0	1	1	1
0	0	1	0	0	1	0

$X \to Y$ logikailag ekvivalens $\neg X \vee Y$-nal.

$X \otimes Y$ logikailag ekvivalens $(\neg X \wedge Y) \vee (X \wedge \neg Y)$-nal.

A logikai műveleteket Boole-függvényeknek is szokás nevezni.

Definíció

Boole-függvénynek nevezünk egy $f : \{0,1\}^n \to \{0,1\}$ leképezést.

Boole függvények

Sok algoritmikus probléma valójában megfelel egy Boole-függvény kiszámításának.

Példa: Ha egy n csúcsú irányítatlan gráfról szeretnénk eldönteni, hogy rendelkezik-e egy tulajdonsággal, akkor az valójában egy $f : \{0,1\}^N \to \{0,1\}$ Boole-függvény kiszámítását jelenti, ahol $N = \binom{n}{2}$. A változókat feleltessük meg az N csúcspárnak, és egy változó értéke akkor legyen 1, ha a gráfban a két csúcs között van él. Így a változókiértékelések bijekcióba állíthatók a lehetséges gráfokkal.

Akár NP-teljes problémák is leírhatók így, ha például ez a tulajdonság az, hogy van-e a gráfban Hamilton kör.

Boole-polinomok

Definíció

A konjunkció, diszjunkció és negáció műveleteivel felírt kifejezéseket **Boole-polinomoknak** nevezzük.

Tétel

Minden Boole-függvény kifejezhető Boole-polinomként.

Bizonyítás: Legyen $f(X_1, \ldots, X_n) : \{0,1\}^n \to \{0,1\}$ egy tetszőleges Boole-függvény és vezessük be az $f^1 = \{\mathbf{z} \in \{0,1\}^n \mid f(\mathbf{z}) = 1\}$ jelölést. Ha $\mathbf{z} \in \{0,1\}^n$ akkor legyen

$$\varphi_{\mathbf{z}}(X_1, \ldots, X_n) := \bigwedge_{i=1}^{n} L_i,$$

ahol $L_i = X_i$ ha $z_i = 1$ és $L_i = \neg X_i$ ha $z_i = 0$.

Ekkor $\varphi_{\mathbf{z}}(X_1, \ldots, X_n) = 1$ akkor és csak akkor, ha $X_i = z_i$ minden $1 \leqslant i \leqslant n$ esetén.

Boole-polinomok

Tehát a

$$\psi(X_1,\ldots,X_n) = \bigvee_{\mathbf{z}\in f^1} \varphi_{\mathbf{z}}(X_1,\ldots,X_n)$$

Boole-polinomra $\psi(X_1,\ldots,X_n) = 1 \iff$ valamely $\mathbf{z} \in f^1$-re $\varphi_{\mathbf{z}}(X_1,\ldots,X_n) = 1 \iff$ valamely $\mathbf{z} \in f^1$-re $X_i = z_i$ minden $1 \leqslant i \leqslant n \iff (X_1,\ldots,X_n) \in f^1 \iff f(X_1,\ldots X_n) = 1$. □

Példa:

X_1	X_2	X_3	$f(X_1,X_2,X_3)$	$\psi(X_1,X_2,X_3) =$
1	1	1	0	
1	1	0	1	$(X_1 \wedge X_2 \wedge \neg X_3) \vee$
1	0	1	0	
1	0	0	0	
0	1	1	0	
0	1	0	1	$(\neg X_1 \wedge X_2 \wedge \neg X_3) \vee$
0	0	1	1	$(\neg X_1 \wedge \neg X_2 \wedge X_3) \vee$
0	0	0	1	$(\neg X_1 \wedge \neg X_2 \wedge \neg X_3)$

Topologikus sorrend

Definíció

Legyen $G = (V, E)$ egy irányított gráf. A csúcsaink egy $v_1,\ldots,v_n$ sorrendjét **topologikus sorrendnek** nevezzük, ha $(v_i, v_j) \in E \Rightarrow i < j$.

Tétel

Egy irányított gráf akkor és csak akkor aciklikus (azaz, irányított kört nem tartalmazó), ha a csúcsainak van topologikus sorrendje.

A tétel BSc-s tananyag, itt nem bizonyítjuk.

Gráfok körmentességének ellenőrzése és egyúttal aciklikusság esetén a csúcsok topologikus sorrendjének megadására hatékony, O(élszám) futási idejű algoritmusok ismeretesek (például a mélységi keresés befejezési száma szerinti csökkenő sorrend jó; egy gráf körmenetes a.cs.a. ha a mélységi keresés nem talál visszaélt).

Boole-hálózatok

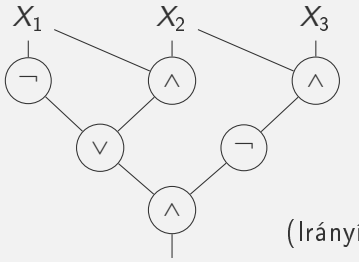
- A logikai hálózat logikai műveleteknek megfelelő kapuk hálózata, speciális esete a Boole-hálózat, ahol a kapuk típusa csak $\neg$, $\wedge$ és $\vee$ lehet.
- A digitális áramkör elméleti megfelelője a Boole-hálózat, a modell számítási erejét tekintve a Turing gépekkel ekvivalens.
- A logikai hálózatok segítségével közvetlenül, jól áttekinthetően leírható az algoritmusok logikai struktúrája.
- Az algoritmusok logikai hálózatokkal történő leírása ötleteket adhat párhuzamos algoritmusok készítésére.
- A P≠NP sejtés támadására potenciálisan alkalmasnak vélt számítási modell.
- A Cook-Levin tételre (a SAT nyelv NP-teljes) alternatív bizonyítás adható Boole-hálózatok segítségével.

Boole-hálózat

Definíció

Boole-hálózat egy csúcscímkézett, aciklikus irányított gráf, melyre
- a források (bemeneti csúcsok) páronként különböző ítéletváltozókkal vannak címkézve,
- a többi csúcs **logikai kapu**, ezek az $\neg$, $\wedge$, $\vee$ logikai műveletek valmelyikével vannak címkézve, a $\neg$ kapuk befoka 1, a $\wedge$ és $\vee$ kapuk befoka 2,
- egyetlen 0 kifokú kapu (nyelő) van, ezt **kimeneti kapunak** (vagy kimeneti csúcsnak) nevezzünk.

Példa:



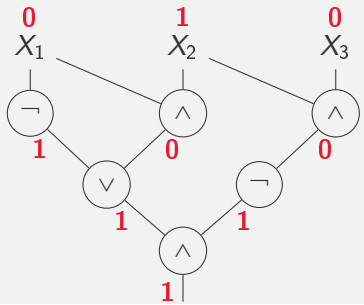
(Irányítás: ↓)

Boole-hálózatok kiértékelése

Definíció

Legyenek $V(C) = \{X_1, \ldots, X_n\}$ a C Boole-hálózat bemeneti csúcsai és $I : V(C) \to \{0,1\}$ egy változókiértékelés. Tekintsük C egy tetszőleges topologikus sorrendjét. Ekkor az egyes logikai kapukhoz tartozó Boole-értéket ezen sorrend szerint a kapu típusa alapján a korábbi kapukhoz és bemeneti csúcsok rendelt Boole-értékek alapján meghatározhatjuk. A C **Boole-hálózat** $\mathcal{B}_I(C)$ **Boole-értéke** I-ben a kimeneti kapuhoz rendelt Boole-érték.

Példa:

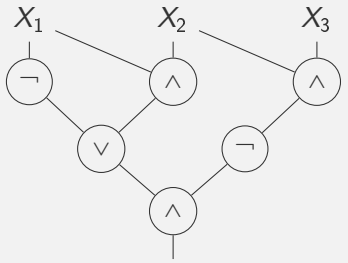


Boole-hálózat által kiszámított Boole-függvény

Definíció

A C **Boole-hálózat által kiszámított Boole-függvény** alatt az $f : (I(X_1), \ldots, I(X_n)) \mapsto \mathcal{B}_I(C)$ függvényt értjük ($I : V(C) \to \{0,1\}$ tetszőleges).

Példa:

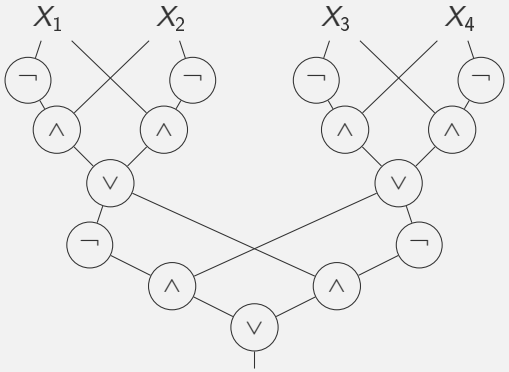


A kiszámított Boole-függvény:

X_1	X_2	X_3	$\mathcal{B}_I(C)$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	1

Boole-hálózatok – Példa

Példa: A $\text{parity}_n : \{0,1\}^n \to \{0,1\}$ Boole függvény akkor és csak akkor ad 1-et, ha páratlan sok 1-es bemenete van. $n = 4$:



$\text{parity}_1(X_1) = X_1$, $\text{parity}_2(X_1, X_2) = X_1 \otimes X_2$,
$\text{parity}_3(X_1, X_2, X_3) = X_1 \otimes (X_2 \otimes X_3)$,
$\text{parity}_4(X_1, X_2, X_3, X_4) = (X_1 \otimes X_2) \otimes (X_3 \otimes X_4)$,
$\text{parity}_n(X_1, \ldots, X_n) = \text{parity}_{n-2}(X_1, \ldots, X_{n-2}) \otimes (X_{n-1} \otimes X_n)$.

Boole-hálózatok – általánosítási lehetőségek

Egy általánosítási lehetőség ha több nyelőt is megengedünk, akkor tetszőleges függvényeket is kiszámíthatunk Boole hálózatok segítségével. Ilyenkor a kimenet egy bitsorozat, melynek a hossza a nyelők száma. Általánosabb függvények kiszámításához kódoljuk át a be- és kimeneteket $\{0,1\}$ feletti szóvá.

Példa:

Megkonstruálható egy olyan Boole-hálózat, amelynek a bemeneti változói $X_0, \ldots, X_{n-1}, Y_0, \ldots, Y_{n-1}$ és az $n+1$ kimeneti kapuján az $X_{n-1} \cdots X_0$ és $Y_{n-1} \cdots Y_0$ bináris számok összegének a bitjeit számítja ki.

Ha $R_0, \ldots R_{n-1}$ a maradékok bitjei és $Z_0, \ldots Z_n$ az eredmény bitjei, akkor könnyen látható, hogy $Z_0 = X_0 \otimes Y_0$, $R_0 = X_0 \wedge Y_0$,

$Z_i = \text{parity}_3(X_i, Y_i, R_{i-1})$ $(1 \leqslant i \leqslant n-1)$,

$R_i = \text{majority}_3(X_i, Y_i, R_{i-1})$ $(1 \leqslant i \leqslant n-1)$,

$Z_n = R_{n-1}$, ahol parity_3-t lásd fenn, míg majority_3 a 3 bit többségi bitjét adja vissza.

Boole-hálózatok – általánosítási lehetőségek

Felmerülhet egyéb kapuk használata is.

Minden Boole-függvény leírható Boole-polinommal, így további kaputípusok alkalmazása nem növeli a modell számítási erejét.

Vegyük észre, hogy a kapuk valójában maguk is Boole-függvények. Így 2-nél több bemenetű kapuk használata esetén a kapuk komplexitását is figyelembe kéne venni a hálózat bonyolultságának definiálásakor különben használhatatlanná válik a modellünk.

Extrém példa: az NP-teljes Hamilton kör probléma kiszámítható egyetlen $\binom{n}{2}$ változós Hamilton-kapuval, melynek $\binom{n}{2}$ bemenete meghatároz egy gráfot (van-e az egyes pontpárok között él), kimenete 1 $\iff$ a gráfban van Hamilton kör.

Ezért a legfeljebb 2 befokú $\neg, \wedge, \vee$ kapukra korlátozzuk a megengedett kaputípusokat. Meggondolható, hogy az egyéb 2 változós műveleteknek megfelelő kapuk mindegyike néhány (egyszámjegyű) $\neg, \wedge, \vee$ kapuval helyettesíthető.

Boole-hálózatok – általánosítási lehetőségek

Felmerülhet az is hogy egy korlátot adjunk meg a kapuk ki-fokára. Néha célszerű lehet feltenni, hogy egy kapu az általa kiszámított bitet nem tudja akárhány helyre „ingyen” szétosztani.

Boole-hálózatok családja

Egyetlen Boole-hálózat önmagában csak adott hosszúságú bemeneteket tud kezelni.

Definíció

Boole-hálózatok egy családja alatt Boole-hálózatok egy végtelen $\mathcal{C} = (C_0, C_1, C_2, \ldots)$ sorozatát értjük, ahol C_n -nek n bemeneti változója van.

Legyen C egy Boole-hálózat az $X_1, \ldots, X_n$ bemeneti változókkal és $w = a_1 \cdots a_n \in \{0,1\}^*$, ekkor $C(w) := \mathcal{B}_I(C)$, ahol $I(X_i) = a_i$ $(1 \leqslant i \leqslant n)$.

Definíció

Boole-hálózatok egy $\mathcal{C}$ családja **eldönti** az $L \subseteq \{0,1\}^*$ nyelvet, ha minden $w \in \{0,1\}^*$ szóra, $w \in L$ akkor és csak akkor, ha $C_n(w) = 1$, ahol $n = |w|$.

Boole-hálózatok mérete és mélysége

Definíció

- Egy Boole-hálózat **mérete** a kapuinak száma
- Egy Boole-hálózat **mélysége** a leghosszabb irányított útjának hossza. (Ez nyilván valamelyik bemeneti csúcstól a nyelőig vezet.)
- A C és C' Boole-hálózatok **ekvivalensek**, ha megegyezik a bemeneti csúcsaiknak a száma (n) és minden $w \in \{0,1\}^n$ esetén $C(w) = C'(w)$.

Definíció

- Egy **Boole-hálózat minimális méretű**, ha nincs vele ekvivalens kisebb méretű hálózat.
- Egy **Boole-hálózat minimális mélységű**, ha nincs vele ekvivalens kisebb mélységű hálózat.

Minimális méretű/mélységű Boole-hálózat család

Definíció

- **Boole-hálózatok** egy $\mathcal{C} = (C_1, \ldots)$ **családja minimális méretű**, ha minden $i \in \mathbb{N}$ esetén a C_i Boole-hálózat minimális méretű.
- **Boole-hálózatok** egy $\mathcal{C} = (C_1, \ldots)$ **családja minimális mélységű**, ha minden $i \in \mathbb{N}$ esetén a C_i Boole-hálózat minimális mélységű.

Boole-hálózatok méret- és mélységbonyolultsága

Definíció

- Boole-hálózatok egy $\mathcal{C} = (C_1, \ldots)$ családjának **méretbonyolultságán** azt az $f : \mathbb{N} \to \mathbb{N}$ függvényt értjük, melyre $f(n)$ a C_n mérete ($n \in \mathbb{N}$).
- Boole-hálózatok egy $\mathcal{C} = (C_1, \ldots)$ családjának **mélységbonyolultságán** azt az $f : \mathbb{N} \to \mathbb{N}$ függvényt értjük, melyre $f(n)$ a C_n mélysége ($n \in \mathbb{N}$).

Definíció

- Egy $L \subseteq \{0,1\}^*$ **nyelv hálózatméret-bonyolultságán** egy minimális méretű, őt eldöntő Boole-hálózat család méretbonyolultságát értjük.
- Egy $L \subseteq \{0,1\}^*$ **nyelv hálózatmélység-bonyolultságán** egy minimális, őt eldöntő Boole-hálózat család mélységbonyolultságát értjük.

Boole-hálózatok méret- és mélységbonyolultsága

Példa:

Legyen $L_{\text{parity}} = \left\{a_1 \cdots a_n \in \{0,1\}^* \mid \sum_{i=1}^{n} a_i \equiv 1 (\text{mod } 2)\right\}$.

Könnyen készíthető olyan Boole-hálózat család, amelynek az elemei rendre a

$\text{parity}_1 = X_1 \quad \text{parity}_2 = X_1 \otimes X_2$
$\text{parity}_n = \text{parity}_{n-2} \otimes (X_{n-1} \otimes X_n)$

Boole-függvényeket számítják ki ($n \in \mathbb{N}$).

Ehhez csak $n-1$ darab $\otimes$ kapu kell. Az előző példában látott konstrukció alapján egy $\otimes$ kapu összesen 5 darab $\neg, \wedge, \vee$ kapuval szimulálható, így L_{parity} hálózatméret-bonyolultsága legfeljebb $5(n-1) = O(n)$.

TG-ek szimulálása Boole-hálózatok családjával

Tétel

Ha $L \in \text{TIME}(f(n))$, és $f(n) \geqslant n$, akkor L hálózatméret-bonyolultsága $O(f(n)^2)$.

Tehát „kis” időbonyolultságú problémák hálózatméret-bonyolultsága is „kicsi”.

Következmény

Ha egy $L \in \text{NP}$ nyelv hálózatméret-bonyolultsága polinomiálisnál nagyobb aszimptotikus nagyágrendű, akkor $\text{P} \neq \text{NP}$.

Így ez a tétel egy lehetséges támadási felületet ad a $\text{P} \neq \text{NP}$ sejtés bizonyításához.

TG-ek szimulálása Boole-hálózatok családjával

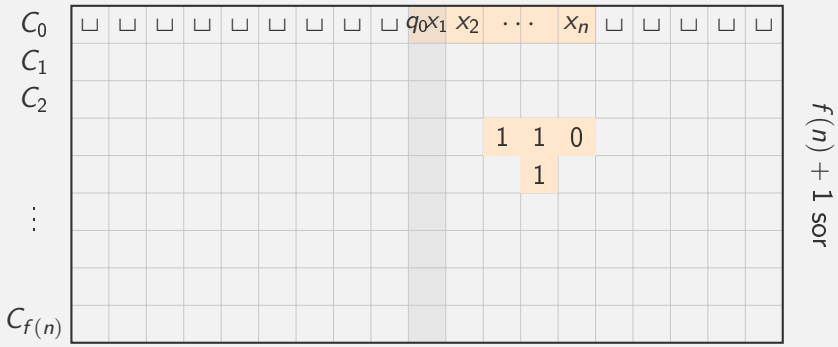
Bizonyítás: Legyen $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n \rangle$ egy $f(n)$ időkorlátos, L-et eldöntő determinisztikus TG és $w = x_1 \cdots x_n$ egy n hosszú inputja. Azaz egyszerűség kedvéért feltesszük, hogy a TIME$(f(n))$ definíciójában lévő $O(f(n))$ konstans faktora 1 (különben mindent ezzel a faktorral kéne szorozni). Feltehető továbbá, hogy $\Sigma = \{0, 1\}$.

Ekkor M számítása w-re a konfigurációi által leírhatók, melyeket ha egymás alá írunk, akkor ez a számítás egy $(f(n)+1) \times (2f(n)+1)$-es T táblázattal leírható. A kezdőkonfiguráció az első sor közepén, az $f(n)+1$-edik cellán kezdődik. ($f(n) \geqslant n$ esetén w belefér T-be.)

A konfigurációk minden betűje saját cellába kerül, kivétel a fej alatt lévő betű, mely kerüljön az állapottal egy cellába az állapot után konkatenálva.

Például, ha a konfiguráció $01q_6101$, akkor a $\sqcup$-eket tároló celláktól eltekintve sorra $0,1,\boxed{q_6 1},0,1$ a cellák tartalma.

TG-ek szimulálása Boole-hálózatok családjával



Ha esetleg $f(n)$-nél rövidebb idő alatt jut M megállási konfigurációba, akkor a táblázat utolsó sorait identikusan, ezen utolsó konfigurációval töltjük ki.

$T(i,j)$ értéke a konfigurációátmenet definíciója szerint csak a $T(i-1,j-1)$, $T(i-1,j)$, $T(i-1,j+1)$ értékeitől függ.

TG-ek szimulálása Boole-hálózatok családjával

A C_n Boole-hálózat konstrukciója:

$k := |\Gamma \cup Q \times \Gamma|$, ennyi fajta tartalma lehet a celláknak. Képzeljük azt, hogy minden (i,j) cellához tartozik k darab villanykörte (light$[i,j,1]$,...,light$[i,j,k]$) a cella minden lehetséges tartalmához pontosan egy. A C_n hálózatban a light$[i,j,s]$ villanykörte akkor és csak akkor fog égni, ha az (i,j) cella tartalma éppen az s-edik $\Gamma \cup Q \times \Gamma$-beli szimbólum. (Rögzítjük $\Gamma \cup Q \times \Gamma$ elemeinek egy sorrendjét.)

A villanykörtéknek nincsen hatásuk a kimeneti értékre, a bizonyítás megértését segítik, és a bizonyítás végén ki is iktatjuk őket.

TG-ek szimulálása Boole-hálózatok családjával

Legyenek $(a_1,b_1,c_1),(a_2,b_2,c_2),\ldots,(a_r,b_r,c_r)$ azok a (light$[i-1,j-1,s_1]$, light$[i-1,j,s_2]$, light$[i-1,j+1,s_3]$) rendezett 3-asok, melyeknek megfelelő égők, ha világítanak akkor az (i,j) cella tartalma az s-edik $\Gamma \cup Q \times \Gamma$-beli szimbólum lesz.

Vegyük észre, hogy r értéke egy csak M-től függő konstans, n-től (és $f(n)$-től) nem függ.

Kössük össze ezen 3-asoknak megfelelő villanykörtéket egy-egy $\wedge$-kapuval, majd ezt az r $\wedge$-kaput egy $\vee$-kapuval, amit vezessünk light$[i,j,s]$-be. (Mivel bináris kapukat használunk ezt persze több bináris $\wedge$ illetve $\vee$ kapuval kell megoldani.)

A táblázat szélein az értékek persze csak 2 fölötte lévő cella értékétől függnek, ezekre értelemszerűen módosítjuk a konstrukciót.

TG-ek szimulálása Boole-hálózatok családjával

Az első sorban lévő celláknak nincs megelőzője, villanykörtéi a bemeneti változókhoz kapcsolódnak. Tehát a light$[1,f(n)+1,\ q_0 1]$ villanykörte az x_1 inputhoz kapcsolódik, mivel a kezdőkonfiguráció a kezdőállapottal kezdődik, az író-olvasó fej pedig x_1-re mutat.

Hasonlóképpen light$[1,\ f(n)+1,\ q_0 0]$ egy $\neg$ kapun keresztül a x_1 inputhoz kapcsolódik.

light$[1,f(n)+i,\ 1]$ az x_i bemenethez, light$[1,f(n)+i,\ 0]$ egy $\neg$ kapun keresztül szintén az x_i bemenethez van bedrótozva.

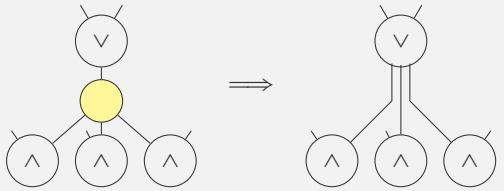
Minden más j-re a light$[1,j,\sqcup]$ villanykörték felkapcsolt állapotban vannak, a többi lámpa az első sorban lekapcsolt állapotban van.

Ez utóbbi megoldható: egy tetszóleges rögzített x_i bemenethez drótozzuk be simán és $\neg$-kapun keresztül is a villanykörtét és ezt a két drótot aszerint kapcsoljuk $\vee$- vagy $\wedge$-kapun keresztül a villanykörtéhez, hogy melyik konstans Boole-értéket szeretnénk hozzárendelni. $(*)$

TG-ek szimulálása Boole-hálózatok családjával

M akkor és csak akkor fogadja el w-t, ha T utolsó, $f(n)+1$-edik sorában megtalálható q_i. Ez $(2f(n)+1)|\Gamma|$ darab villanykörtének felel meg, kössük össze őket egy $\vee$-kapuval, és ez legyen a kimeneti kapu. (ezt persze több bináris $\vee$ kapuval kell megvalósítani).

A villanykörték kiiktatása (1 a be-fokuk):



Meggondoltuk tehát, hogy $(C_0, C_1, \ldots)$ eldönti L-et.

C_n konstukciójában $O(f(n)^2)$ villanykörtét használtunk. Egy villanykörte előző sortól való függőségét egy csak M-től függő konstans darab kapuval szimuláltuk. Az elején és a végén kell még $O(f(n))$ kapu, tehát C_n összesen $O(f(n)^2)$ kaput használ. □

Circuit-Sat

Definíció

Egy C n bemenetű Boole-hálózatot **kielégíthetőnek** nevezünk, ha van olyan $w \in \{0,1\}^n$, melyre $C(w) = 1$.

Circuit-Sat $:= \{\langle C\rangle \mid C$ kielégíthető Boole-hálózat $\}$.

(Szokásos módon $\langle C\rangle$ a C valamely legalább bináris ábécé feletti tömör kódját jelöli.)

Circuit-Sat NP-teljes

Tétel

Circuit-Sat NP-teljes.

Bizonyítás: Könnyen meggondolható, hogy Circuit-Sat NP-beli. (Egy konkrét bemenetre O(hálózat méret) időben kiszámítható a kimenet.)

Legyen $L \in$ NP, kell, hogy $L \leqslant_p$Circuit-Sat, azaz meg kell adnunk egy olyan polinom időben kiszámítható φ függvényt, melyre $w \in L \Leftrightarrow \varphi(w)$ kielégíthető Boole-hálózat.

Mivel $L \in$ NP ezért van olyan $p(n)$ polinom időkorlátos M NTG, mely eldönti L-et. Legyen w egy n hosszúságú szó.

Az előző tétel C_n konstrukciója szimulált egy determinisztikus TG-et bármely n hosszú inputra. Vegyük észre a következőket:

CIRCUIT-SAT NP-teljes

- A konstrukció működik NTG-re is.
- Az előző bizonyításban $(*)$-al jelölt ötlet alapján könnyen módosíthatjuk a konstrukciót úgy, hogy minden bemenetre w-t szimulálja (a w bitjeinek megfelelő villanykörtéket konstans igazra, minden mást konstans hamisra állítva), jelölje ezt C_w, így C_w minden bemenetre w működését szimulálja M-en, így a C_w hálózat akkor és csak akkor kielégíthető, ha $w \in L$. Tehát $\varphi(w) := C_w$ visszavezetés.
- $|C_w| = O(p(n)^2)$, mely $O(p(n)^2)$ idő alatt el is készíthető, így φ poilnomiális. □

3SAT NP-teljes – bizonyítás Boole-hálózatokkal

Tétel

3SAT NP-teljes.

2. Bizonyítás: [1. Bizonyítás : lásd BSc-n]

3SAT NP-beli. Elég: CIRCUIT-SAT $\leqslant_p$ 3SAT.

Legyenek $x_1, \ldots x_n$ a C Boole-hálózat bemeneti változói és rendeljük hozzá az $x_{n+1}, \ldots x_m$ változókat a kapukhoz. Készítünk egy 3KNF-et, melynek $x_1, \ldots x_m$ lesznek a változói.

Minden kapuhoz tartozik 1 vagy 2 bemeneti változó a be-élek alapján és egy kapuhoz tartozó kimeneti változó.

$\neg$-kapu:
$(\bar{x}_i \to x_j) \wedge (\bar{x}_j \to x_i)$ (x_i bemeneti x_j kimeneti változó)

$\wedge$-kapu:
$(\bar{x}_i \wedge \bar{x}_j \to \bar{x}_k) \wedge (\bar{x}_i \wedge x_j \to \bar{x}_k) \wedge (x_i \wedge \bar{x}_j \to \bar{x}_k) \wedge (x_i \wedge x_j \to x_k)$
(x_i,x_j bemeneti x_k kimeneti változó)

3SAT NP-teljes – bizonyítás Boole-hálózatokkal

$\vee$-kapu:
$(\bar{x}_i \wedge \bar{x}_j \to \bar{x}_k) \wedge (\bar{x}_i \wedge x_j \to x_k) \wedge (x_i \wedge \bar{x}_j \to x_k) \wedge (x_i \wedge x_j \to x_k)$
(x_i,x_j bemeneti x_k kimeneti változó)

Az $\wedge$-sel összekötött részformulákat alakítsuk klózokká.

Még egy klóz: x_m önmagában, ha x_m a kimeneti kapu.

A hiányos klózokat egészítsük ki 3 hosszúvá.

A klózok konjunkciója lesz a C-nek megfelelő φ formula.

Meggondolható, hogy C kielégíthető akkor és csak akkor, ha φ kielégíthető és hogy a konstrukció polinomiális idejű valamint polinomiális méretű φ -t ad. □

Számítási modellek

6. előadás

RAM gép

A RAM gép egy speciális **regiszter gép**. A regiszter gépek olyan gépek, ahol

- adott vagy előre meghatározott számú vagy korlátlan számú regiszter (memóriarekesz), amit együtt tekinthetünk a gép memóriájának.
- a regiszterek tartalma egy korlátlan méretű természetes szám lehet (úgy is gondolhatunk rá, hogy a regiszterekben zsetonok vannak) [mi: kényelmi szempontból negatív számok is],
- a gép egy limitált utasításkészlettel rendelkezik, ami tipikusan tartalmaz minimum 1-2 aritmetikai és egy kontrol (feltételes ugrás) utasítást,
- a címkézett utasítások listája lehet külön programtárban vagy valamely regiszterben tárolva,
- az utasításkészlet egyes modelleknél tartalmazhat indirekt címzésű utasítást is,
- az aritmetikai utasításokat vagy minden vagy csak egy speciális regiszter tudja elvégezni.

RAM gép (informális kép)

Példa:

$x[-1]$	$x[0]$	$x[1]$	$x[2]$	$x[3]$	$x[4]$	$x[5]$	$x[6]$	$x[7]$
-1	0	1	2	3	4	5	6	7
	5	12	-15	33	3	-2		

Memória

0	$x[5] := x[5] + 1;$
1	IF $x[5] \leqslant 0$ THEN GOTO 0;
2	$x[x[4]] := x[1];$
3	IF $x[2] \leqslant 0$ THEN GOTO 2;
4	
5	
6	
7	

Programtár

RAM gép (informális kép)

- A RAM gép működése a **programtárjának** és a **memóriájának** tartalmától függ.
- A memória végtelen sok regiszterből (memóriarekeszből) áll, melyek az egész számokkal ($\mathbb{Z}$ elemivel) vannak címezve.
- Minden $i \in \mathbb{Z}$ esetén az i-edik memóriarekesz egy $x[i] \in \mathbb{Z}$ egész számot tartalmaz. A memóriarekeszek közül mindig csak véges soknak nem 0 a tartalma. Az egyes memóriarekeszek tartalma a működés során változhat.
- A programtár potenciálisan végtelen sok, $0, 1, 2, \ldots$ címkékkel ellátott rekeszből áll; ebbe egy adott gépi kódszerű programnyelven egy véges hosszúságú programot írhatunk (rekeszenként 1 utasítással). A RAM gép programja a gép működése során állandó.
- Két fontos különbség a **számláló gépekhez** képest (amik a legegyszerűbb regiszter gépek): a korlátlan számban rendelkezésre álló memóriarekesz és az indirekt címzésű utasítások használata.

A RAM gép

Definíció

Egy $M = \langle x, P \rangle$ **RAM gép** két komponensből áll: az $\langle x[i] \rangle_{i \in \mathbb{Z}}$ memóriarekeszekből és a P programból. $P = \langle (0, \omega_0), \dots (n, \omega_n) \rangle$ egy véges sorozat ($n \in \mathbb{N}$), ahol minden $1 \leqslant k \leqslant n$ esetén az ω_k az alábbiak valamelyike

- $x[i] := 0;$ $(i \in \mathbb{Z})$
- $x[i] := x[i] + 1;$ $(i \in \mathbb{Z})$
- $x[i] := x[i] - 1;$ $(i \in \mathbb{Z})$
- $x[i] := x[i] + x[j];$ $(i, j \in \mathbb{Z})$
- $x[i] := x[i] - x[j];$ $(i, j \in \mathbb{Z})$
- $x[x[i]] := x[j];$ $(i, j \in \mathbb{Z})$
- $x[i] := x[x[j]];$ $(i, j \in \mathbb{Z})$
- IF $x[i] \leqslant 0$ THEN GOTO $p;$ $(i \in \mathbb{Z},\ 0 \leqslant p \leqslant n)$

RAM gép bemenete

Mivel az első komponens, a memória szerkezete minden RAM gépre azonos ezért a RAM gépet azonosíthatjuk a P programjával.

Definíció

Egy **RAM-gép bemenete** egy természetes számokból álló véges sorozat, amelynek a hosszát az $x[0]$ memóriarekeszbe, elemeit pedig rendre az $x[1], x[2], \dots x[x[0]]$ memóriarekeszekbe írjuk be. A többi memóriarekesz tartalma 0.

Megjegyzés: A bemenet hosszát azért tároljuk el $x[0]$-ban, hogy lehessen 0 is egy bemenethez tartozó rekesz tartalma.

Teljes indukcióval könnyen látható, hogy a program végrehajtása során a nem-0 tartalmú memóriarekeszek száma mindig véges.

Észrevétel: Az értékadások egész szám(ok)ból aritmetikai műveletekkel egész értéket adnak vissza, így (mivel a rekeszek tartalma kezdetben egész) a működés során $x[i] \in \mathbb{Z}$ mindvégig fennáll ($i \in \mathbb{Z}$).

RAM gép működése és kimenete

Definíció

- A RAM gép programja a feltételes ugrás kivételével szekvenciálisan hajtódik végre, $(i-1, \omega_{i-1})$-t (i, ω_i) követi $(1 \leqslant i \leqslant n)$.
- Ha ω_k IF $x[i] \leqslant 0$ THEN GOTO $p;$ $(i \in \mathbb{Z},\ 0 \leqslant p \leqslant n)$ típusú utasítás, akkor a feltétel teljesülése esetén a program (p, ω_p)-vel, egyébként pedig $(k+1, \omega_{k+1})$-el folytatódik.
- Egy RAM gép program működése **végetér**, ha egy olyan programsorhoz ér, amiben nincs utasítás.
- **A RAM gép kimenete** alatt a program megállásakor az $\langle x[i] \rangle_{i \in \mathbb{Z}}$ rekeszek tartalmát értjük.

A fentiek alapján a kimenet mindig végesen reprezentálható.

Alternatív kimenet: kijelölhetünk bizonyos rekesz(eke)t kimeneti rekesszé.

RAM gép programjának futási ideje

A végrehajtott utasítások száma, nem a legjobb mérőszáma annak, hogy mennyit dolgozik a RAM gép.

Két nagyon nagy természetes szám összeadása ne legyen egységnyi művelet.

Definíció

A **logaritmikus költségű RAM gép modellben** a egyes utasítások végrehajtásának költsége a benne szereplő természetes számok (rekeszcímek és tartalmak) kettes számrendszerbeli jegyeinek száma. A P RAM gép program **futási ideje** egy I inputra a P által I-re végrehajtott utasítások költségeinek összege.

Néha e helyett két paraméterrel jellemezzük a futási időt, pl.

„a gép legfeljebb $O(f(n))$ lépést végez kettes számrendszerben legfeljebb $O(g(n))$ jegyű számokon".

Ekkor persze a gép $O(f(n)g(n))$ futási idejű.

Alternatív input fogalom

Feladat: Írjunk programot RAM gépre, mely egy N pozitív egészre meghatározza a legnagyobb olyan m-t melyre $2^m \leqslant N$(azaz $m = \lfloor \log_2 N \rfloor$). A program $O(m)$ lépést tegyen m jegyű számokkal.

Megoldás: A bemenet az $x[0]$ rekeszben van, minden más rekesz tartalma 0, a kimenet az $x[1]$ rekeszben lesz.

```
0  x[2] := x[2] + 1;
1  x[3] := x[2];
2  x[2] := x[2] + x[3];
3  x[4] := x[0] − x[2];
4  x[4] := x[4] + 1;
5  IF x[4] ⩽ 0 THEN GOTO 8;
6  x[1] := x[1] + 1;
7  IF x[5] ⩽ 0 THEN GOTO 1;
8
```

Alternatív input fogalom

Állítás: Kiszámítható $O(\log N)$ lépésben, $O(\log N)$ jegyű számokkal N kettes számrendszerbeli alakja.

Ötlet: Az előző feladat alapján állítsuk elő $m = \lfloor \log_2 N \rfloor$-t. Majd $0, \ldots m$-re állítsuk elő a 2-hatványokat $m + 1$ egymást követő rekeszben (2 utasítás kettőhatványonként). Majd N-ből a legnagyobbtól a legkisebb felé haladva kivonva sorra a 2-hatványokat a rekeszek feltölthetők a 2-es számrendszerbeli alak bitjeivel.

Alternatív input fogalom: Az input mindig egyetlen, de méretében nem korlátozott N természetes szám az $x[0]$ rekeszben. A fentiek alapján hatékonyan, azaz $O(\log N)$ lépésben, $O(\log N)$ jegyű számokkal átkonvertálható az eredeti alakra.

Turing ekvivalencia

Tétel

Ha egy függvény $f(n)$ időkorlátos TG-pel kiszámítható, akkor konstruálható $O(f(n))$ lépésszámú $O(\log f(n))$ jegyű számokkal dolgozó program RAM gépre ami ugyanazt a függvényt számítja ki.

Bizonyítás: Mivel a TG egy függvényt számít ki, feltehetjük, hogy a TG-nek egyetlen megállási állapota van.
Legyen $M = \langle Q, \Sigma, \Gamma, \delta, q_1, q_r \rangle$ TG, ahol $r \geqslant 1$, $Q = \{q_1, \ldots, q_r\}$, $\Gamma = \{0, 1, \ldots |\Gamma| - 1\}$, q_r a megállási, q_1 a kezdőállapot.

A Turing gép számolásának szimulálása során a RAM gép $2i$-edik memóriarekeszében ugyanaz a Γ-beli szám fog állni, mint a Turing gép szalagjának i-edik cellájában. Feltehető, hogy az input M nulladik cellájánál kezdődik, így egy n hosszú input első betűje a RAM gép $x[0]$ rekeszében, utolsó betűje $x[2n-2]$-ben található.

A kis, páratlan indexű rekeszeket használjuk minden másra. $x[1]$ tartalma a működés során mindig a fej helyzetének megfelelő cella sorszáma.

Turing ekvivalencia

Programunk P_i $(1 \leqslant i \leqslant r)$ és $Q_{i,j}$ $(1 \leqslant i \leqslant r-1, 0 \leqslant j \leqslant |\Gamma| - 1)$ részekből fog állni. A P_i programrész $1 \leqslant i \leqslant r-1$ akkor kerül meghívásra, ha a Turing gép vezérlőegysége egy átmenet után az i-edik állapotába kerül, ekkor a RAM program a szalagon olvasott betű (j) függvényében elágazik a $Q_{i,j}$ programrészekre.

A P_i programrész $(1 \leqslant i \leqslant r-1)$:

```
x[3] := x[x[1]];
IF x[3] ⩽ 0 THEN GOTO [Q_{i,0} címe];
x[3] := x[3] − 1;
IF x[3] ⩽ 0 THEN GOTO [Q_{i,1} címe];
⋮
x[3] := x[3] − 1;
IF x[3] ⩽ 0 THEN GOTO [Q_{i,|Γ|−1} címe];
```

A P_r programrész álljon egyetlen üres programsorból.

Turing ekvivalencia

$Q_{i,j}$ programrész akkor kerül meghívásra, ha az i-edik állapotban j-t olvas a fej. Ekkor a TG átírja az $x[1]$-edik rekeszt $\delta(q_i, j) = (q_{\alpha(i,j)}, \beta(i,j), \gamma(i,j))$ szerint $\beta(i,j)$-re, módosítja $x[1]$-et $\gamma(i,j)$ szerint, és az új, $q_{\alpha(i,j)}$ állapotának megfelelő $P_{\alpha(i,j)}$ programrészre ugrik. A $Q_{i,j}$ programrész:

$x[3] := 0;$
$\left.\begin{array}{l} x[3] := x[3] + 1; \\ \vdots \\ x[3] := x[3] + 1; \end{array}\right\}$ $\beta(i,j)$ -szer
$x[x[1]] := x[3];$
$x[1] := x[1] + 1;$ ha $\gamma(i,j) = R$
$x[1] := x[1] + 1;$ ha $\gamma(i,j) = R$
$x[1] := x[1] - 1;$ ha $\gamma(i,j) = L$
$x[1] := x[1] - 1;$ ha $\gamma(i,j) = L$
$x[3] := 0;$
IF $x[3] \leqslant 0$ THEN GOTO [$P_{\alpha(i,j)}$ címe];

Turing ekvivalencia

A főprogram:

$x[1] := 0;$
P_1
$\vdots$
P_r
$Q_{0,0}$
$\vdots$
$Q_{r-1,|\Gamma|-1}$

Ezzel a Turing gép szimulálását befejeztük.

A futási időre vonatkozó állítás abból következik, hogy a Turing gép minden lépésének szimulálása legfeljebb $3|\Gamma| + 6 = O(1)$ RAM gép utasítással történik, így a RAM gép $O(f(n))$ utasítást hajt végre. $f(n)$ lépésben a Turing gép legfeljebb egy $-f(n)$ és $f(n)$ közötti sorszámú cellába írhat bármit is. Így a program egyes lépéseiben $O(\log f(n))$ hosszúságú számokkal dolgozunk. □

Turing ekvivalencia

Tétel

Minden RAM gépre írt programhoz van olyan Turing gép, amely minden bemenetre ugyanazt a kimenetet számítja ki, mint a RAM gép. Ha a RAM-gép futási ideje $f(n)$, akkor a Turing gép $O(f(n)^2)$ időkorlátos.

Bizonyítás: A RAM-gép számolását egy négyszalagos Turing géppel fogjuk szimulálni.

Alapötlet: A Turing gép egyik szalagjára írjuk fel sorban az $x[i]$ memóriarekeszek tartalmát (kettes számrendszerben, ha negatív, előjellel ellátva, # jelekkel elválasztva), minden rekesz tartalmát sorra feltüntetve.

Turing ekvivalencia

Gond: Ha a RAM-gép a 2^x sorszámú rekeszbe ír, akkor ez a logaritmikus költség modell szerint x költséget jelent. Ilyenkor a Turing gépnek a 2^x-edik (vagy akár még nagyobb sorszámú) cellájába kell írnia, és ha a fej aktuális helyzete távol esik ettől a cellától, akkor az akár $\Omega(2^x)$ lépést (és így költséget) is jelenthet a Turing gép számára.

Ilyen költséges reprezentációt nem engedhetünk meg magunknak, mert ez összességében akár a futási idő exponenciális növekedését eredményezheti.

Vegyük azonban észre, hogy túl nagy indexű rekeszek használata $f(n)$ összköltség esetén csak úgy lehetséges, ha a kisebb indexű rekeszek túlnyomó többségének a tartalma 0 marad az egész számítás folyamán.

Turing ekvivalencia

Javítás: Csak azoknak a rekeszeknek a tartalmát tároljuk a Turing gép szalagján, melyekbe ténylegesen ír a RAM-gép. Ekkor azt is fel kell tüntetni, hogy mi a szóban forgó rekesz sorszáma.

Azt tesszük tehát, hogy valahányszor a RAM-gép egy $x[z]$ rekeszbe egy y számot ír, a Turing gép ezt úgy szimulálja, hogy az első szalagja végére a $\#\#y\#z$ jelsorozatot írja, y-t és z-t kettes számrendszerben. (Korábbi tartalmat sosem ír felül, csak hozzáadja ezt az információt a szalag tartalmához!)

$\#\#y\#z$ hossza így 3-mal hosszabb, mint a logaritmikus költség modellben a RAM gép értékadásának a költsége. Mivel y és z leírásához legalább 1 bit kell, így legfeljebb $5/2$-szerese a hossznövekmény a szimulált RAM gép utasítás logaritmikus költségének.

Turing ekvivalencia

Ha a RAM-gép egy $x[z]$ rekesz tartalmát olvassa ki, akkor a Turing gép első szalagján a fej hátulról indulva megkeresi az első $\#\#u\#z$ alakú részszót; ez az u érték a rekesz tartalma, hiszen ez volt utoljára a z-edik rekeszbe írva.

Ha ilyen alakú részszót nem talál, akkor $x[z]$-t 0-nak tekinthetjük.

A RAM-gép programnyelvének minden egyes utasítását könnyű szimulálni egy-egy alkalmas Turing géppel, amely csak a másik három szalagot használja.

A Turing gép egy olyan „szupergép" lesz, amelyben minden programsornak megfelel egy rész-Turing gép, és az ehhez tartozó állapotok egy halmaza.

Ez a rész-Turing gép az illető utasítást végrehajtja, az első szalag fejét a végére (utolsó nemüres cellájára) állítja, a többi szalagot üresen adja vissza.

Turing ekvivalencia

Minden ilyen Turing gép végállapota azonosítva van a következő sornak megfelelő Turing gép kezdőállapotával. (A feltételes ugrás esetén, ha $x[i] \leqslant 0$ teljesül, a p sornak megfelelő Turing gép kezdőállapotába megy át a „szupergép".)

A 0-dik programsornak megfelelő Turing gép kezdőállapota lesz a szupergép kezdőállapota is.

Ezenkívül lesz még egy végállapot; ez felel meg minden üres programsornak.

Belátható, hogy az így megkonstruált Turing gép lépésről lépésre szimulálja a RAM-gép működését.

Turing ekvivalencia

A legtöbb programsort a Turing gép a benne szereplő számok számjegyeinek számával, vagyis éppen a RAM-gépen erre fordított idejével arányos lépésszámban hajtja végre.

Kivétel egy $x[i]$ érték kiolvasása, amelyhez esetleg (egy lépésnél legfeljebb kétszer) végig kell keresni az egész szalagot.

Mivel az első szalag tartalmának a hossza egy K költségű RAM gép utasítás esetén $O(K)$-val nő lépésenként, ezért $f(n)$ lépés alatt $O(f(n))$-re duzzadhat. Így a TG futási ideje $f(n)O(f(n)) = O(f(n)^2)$. □

PRAM gép

PRAM gép: párhuzamos RAM gép – párhuzamos számítást végez. Ez $p \geqslant 1$ azonos RAM-gépből (processzorból) áll.

A gépek programtára közös, és van egy közös memóriaterületük is, amely mondjuk az $x[i]$ rekeszekből áll (ahol i az egész számokon fut végig).

Feltehető, hogy minden processzornak van még végtelen sok $u[i]$ ($i \in \mathbb{Z}$) saját memóriarekesze.

Induláskor minden processzor $u[0]$ rekeszében a processzor saját sorszáma áll.

A processzor írhat és olvashat a saját $u[i]$ rekeszeibe, ill. rekeszeiből és a közös $x[i]$ memóriarekeszekbe, ill. memóriarekeszekből.

PRAM gép

A bemenetet az $x[0], x[1], \ldots, x[n]$ rekeszekbe írjuk, ahol $x[0] = n$ a bemenet hossza.

A bemenet és a (közös) program mellé meg kell adni azt is, hogy hány processzorral dolgozunk; ezt írhatjuk pl. az $x[-1]$ rekeszbe.

A processzorok párhuzamosan, de ütemre hajtják végre a programot.

További utasítások az alapmodell utasításain felül:

- $u[i] := x[u[j]];$ $\quad (i, j \in \mathbb{Z})$
- $x[u[i]] := u[j];$ $\quad (i, j \in \mathbb{Z})$
- $u[i] := u[i] \cdot u[j];$ $\quad (i, j \in \mathbb{Z})$
- $u[i] := u[i] : u[j];$ $\quad (i, j \in \mathbb{Z})$

Az utóbbi a maradékos osztás, $a : b$ eredménye a legnagyobb c természetes szám, amelyre $|a| \geqslant |b|c$.

PRAM gép

A szorzást és maradékos osztást azért célszerű hozzávenni, hogy egy processzor a saját sorszámából mindig ki tudja 1 lépésben számolni, hogy melyik $x[f(x[-1], x[0], u[0])]$ inputcellát kell először olvasnia, legalábbis elég egyszerű f függvény esetén.

Példa: $f(x[-1], x[0], u[0]) = (u[0] : x[0]) (\mathrm{mod}\ x[-1])$.

PRAM gép

Logaritmikus költségfüggvényt használunk: pl. egy k egész számnak az $x[t]$ memóriarekeszbe való beírásának vagy kiolvasásának a költsége $\log |k| + \log |t|$.

A szorzásnál és maradékos osztásnál ehhez még hozzáadjuk a két operandus bitszámának szorzatát.

A következő ütem akkor kezdődik, amikor az előző műveletet minden processzor végrehajtotta.

Egy ütem idejét tehát a legtöbb időt igénybevevő processzor határozza meg.

A gép akkor áll meg, ha minden processzor olyan programsorhoz ér, amelyben nincsen utasítás.

A kimenet az $x[i]$ rekeszek tartalma.

PRAM gép konfliktuskezelés

Konvenciók az I/O konfliktusok kezelésére:

EREW (Exclusive Read Exclusive Write) – Teljesen konfliktusmentes modell

Nem szabad két processzornak ugyanabból a rekeszből olvasni vagy ugyanabba a rekeszbe írni. A programozónak kell gondoskodnia arról, hogy ez ne következzen be; ha mégis megtörténne, a gép programhibát jelez.

CREW (Concurrent Read Exclusive Write) – Félig konfliktusmentes modell

Talán legtermészetesebb az a modell, melyben megengedjük, hogy több processzor is ugyanazt a rekeszt olvassa, de ha ugyanoda akarnak írni, az már programhiba.

PRAM gép konfliktuskezelés

CRCW (Concurrent Read Concurrent Write) – Konfliktus-korlátozó modell

Szabad több processzornak ugyanabból a rekeszből olvasni és ugyanabba a rekeszbe írni is, de csak akkor, ha ugyanazt akarják beírni. A gép akkor áll le programhibával, ha két processzor ugyanabba a rekeszbe más-más számot akar beírni.

P-CRCW (Priority Concurrent Read Concurrent Write) – Elsőbbségi modell

Szabad több processzornak ugyanabból a rekeszből olvasni és ugyanabba a rekeszbe írni. Ha többen akarnak ugyanoda írni, a legkisebb sorszámúnak sikerül.

PRAM gép

A PRAM-gépnél a processzorok számát nem csak azért szokás megadni, mert ettől függ a számítás, hanem azért is, mert ez – az idő és tár mellett – a számítás igen fontos bonyolultsági mértéke.

Példa: Ha ezt nem korlátozzuk,akkor igen nehéz feladatokat is nagyon röviden meg tudunk oldani, pl. egy gráf 3 színnel való színezhetőségét el tudjuk dönteni úgy, hogy az alaphalmaz minden színezéséhez és a gráf minden éléhez rendelünk egy processzort (n csúcsú, e élű gráf esetén $e3^n \leqslant n^2 3^n$ darab processzor). Adott színezéshez tartozó élprocesszorok megnézik, hogy a színezésben az adott él két végpontja különböző színű-e.

Az eredmények összesítése a CRCW modellben $O(1)$ lépésben megoldható. (Az élprocesszorok a színezésüknek „jelentenek" egy közös memóriacímen, majd a színezések egyetlen egy közös memóriacímen „jelentik", hogy jó-színezések-e.)

Az EREW modellben az összesítés színezésenként $O(\log n)$ lépésben, mindösszesen $O(n \log n)$ lépésben elvégezhető.

PRAM gép – Brent tétele

Feltehetjük, hogy egy párhuzamos számítási modell processzorai, amennyiben egy adott lépésben aktívak, 1 egységnyi munkát végeznek, míg ha inaktívak, akkor 0 egységnyit.

Definíció

A P (akármelyik PRAM-gép modellben írt) program adott bemenetre t lépésben végzett **összmunkája** $w = \sum_{i=1}^{t} w_i$, ahol w_i az i-edik lépésben aktív processzorok száma.

Brent tétele

Ha egy feladatot egy konkrét inputra a P program (akármelyik PRAM-gép modellben) valahány processzorral t lépésben és w összmunkával megoldja, akkor minden p pozitív egész számra megoldható p processzorral is $(w - t)/p + t$ lépésben (ugyanabban a modellben).

PRAM gép – Brent tétele

Bizonyítás: Tegyük fel, hogy a P program i-edik lépésében w_i darab processzor aktív, ekkor az összmunka $w = \sum_{i=1}^{t} w_i$. Az eredeti algoritmus i-edik lépése során elvégzett feladatot nyilván el lehet végezni p processzorral $\lceil w_i/p \rceil$ lépésben, így a szükséges idő $\sum_{i=1}^{t} \lceil w_i/p \rceil \leqslant \sum_{i=1}^{t} (w_i + p - 1)/p \leqslant (w - t)/p + t$. □

Példa: ha van egy feladatra párhuzamos algoritmusunk, mely n^2 processzort használ és $\log_2 n$ lépést tesz, akkor olyan is van, mely

- 32 processzort használ $\leqslant n^2 \log_2 n/32 + \log_2 n = O(n^2 \log n)$ lépést tesz
- $\sqrt{n}$ processzort használ és $\leqslant n^2 \log_2 n/\sqrt{n} + \log_2 n = O(n^{3/2} \log n)$ lépést tesz

Valóban, $t = \log_2 n$ a w összmunkára $w \leqslant n^2 \log_2 n$, alkalmazzuk Brent tételét.

PRAM gép – P-CRCW EREW szimulációja

A P-CRCW számítások hatékonyan szimulálhatók az EREW modellben.

Tétel

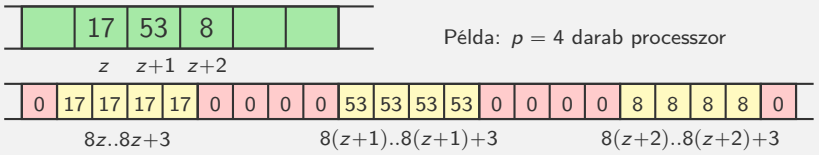
Minden P programhoz létezik olyan Q program, hogy ha P egy bemenetből p processzorral t időben kiszámít egy kimenetet az elsőbbségi (P-CRCW) modellben, akkor a Q program $O(p^2)$ processzorral $O(t \log^2 p)$ időben a teljesen konfliktusmentes (EREW) modellben számítja ki ugyanazt.

Bizonyítás:
A P programot végrehajtó minden processzornak meg fog felelni egy processzor a Q program végrehajtása során. Ezeket főnökprocesszoroknak nevezzük.

Ezenfelül a Q programban minden főnökprocesszornak p további „segédprocesszora" is lesz.

PRAM gép – P-CRCW EREW szimulációja

Az a konstrukció alapgondolata, hogy ami a P program futása során egy adott ütem után a z című memóriarekeszben van, az a Q program futása során a megfelelő ütemben a $2pz, 2pz + 1, \ldots, 2pz + p - 1$ című rekeszek mindegyikében meglegyen.



Példa: $p = 4$ darab processzor

Ha a P következő ütemében az i-edik processzor z című rekeszből kell, hogy olvasson, akkor a Q program következő ütemében az ennek megfelelő processzor olvashatja a $2pz + i$ című rekeszt.

Ha pedig ebben az ütemben (P szerint) a z című rekeszbe kell, hogy írjon, akkor Q szerint a $2pz + i$ rekeszbe írhat.

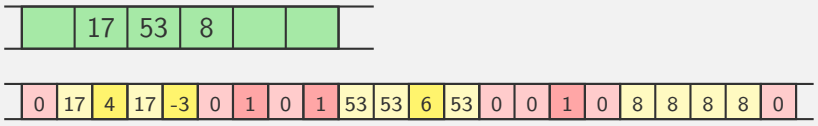
Ez biztosan elkerüli az összes konfliktust, mert a különböző processzorok modulo p különböző rekeszeket használnak.

PRAM gép – P-CRCW EREW szimulációja

Gondoskodni kell azonban arról, hogy a Q program futása során a $2pz, 2pz + 1, \ldots, 2pz + p - 1$ rekeszek mindegyikébe az a szám kerüljön, amit az elsőbbségi szabály a P program futásának megfelelő ütemében z-be ír.

Ehhez a P futását szimuláló minden ütem után beiktatunk egy $O(\log p)$ lépésből álló fázist, amely ezt megvalósítja.

Először is minden olyan processzor, mely az előző (P-t szimuláló) ütemben írt, mondjuk a $2pz + i$ című rekeszbe, ír egy 1-est a $2pz + p + i$ című rekeszbe.



proc2: $x[z] := 4$
proc4: $x[z] := -3$
proc3: $x[z + 1] := 6$

PRAM gép – P-CRCW EREW szimulációja

A következő lépésben megnézi, hogy a $2pz + p + i - 1$ rekeszben 1-es áll-e.

Ha igen, elalszik ennek a fázisnak a hátralevő idejére.

Ha nem, akkor ő ír oda 1-est, majd „felébreszti" egy segédjét.

Általában a k-adik lépésben az i-edik processzornak legfeljebb 2^{k-1} segédje lesz ébren (magát is beleértve), és ezek rendre, amennyiben aktívak, a $2pz + p + i - 2^{k-1}, \ldots, 2pz + p + i - (2^k - 1)$ című rekeszeket olvassák le.

Amelyik 1-est talál, elalszik.

Amelyik nem talál 1-est, az 1-est ír, felébreszt egy új segédet, azt 2^{k-1} hellyel „balra" küldi, míg maga 2^k hellyel „balra" lép.

Amelyik segéd kiér a $[2pz + p, 2pz + 2p - 1]$ intervallumból, elalszik; ha egy főnök kiér ebből az intervallumból, akkor már tudja, hogy ő „győzött".

PRAM gép – P-CRCW EREW szimulációja

Könnyű meggondolni, hogy ha a P program megfelelő ütemében több processzor is írni akart a z rekeszbe, akkor az ezeknek megfelelő főnökök és segédeik nem kerülnek konfliktusba, mialatt a $[2pz + p, 2pz + 2p - 1]$ intervallumban mozognak.

A k-adik lépésre ugyanis az i-edik processzor és segédei $2pz + p + i$-től lefelé 2^{k-1} egymás utáni helyre 1-est írtak.

Minden tőlük jobbra induló processzor és annak minden segédje ebbe szükségképpen bele fog lépni, és ezért elalszik, mielőtt konfliktusba kerülhetne az i-edik processzor segédeivel.

Ha nem ért még célba főnök és legalább 2 főnök aktív még, akkor a hátrébbról induló épp akkorát lép, mint az előbbről induló által eddig készített 1-esekből álló blokk mérete, mivel ez a blokk teljes egészében előtte van, így nem érhet a következő lépésben célba.

Ez mutatja azt is, hogy mindig egyetlenegy főnök fog győzni, éspedig a legkisebb sorszámú.

PRAM gép – P-CRCW EREW szimulációja

Tehát $O(\log p)$ lépésben eldől, melyik processzor a „győztes". Ennek a processzornak még van egy dolga. Amit a $2pz + i$. rekeszbe írt, azt most az egész $[2pz, 2pz + p - 1]$ intervallum minden rekeszébe be kell írnia.

Ezt könnyű megtenni $O(\log p)$ lépésben az előzőhöz nagyon hasonló eljárással: ő maga beírja a kívánt értéket a $2pz$ című rekeszbe, majd felébreszt egy segédet; beírják a kívánt értéket a $2pz + 1$ és $2pz + 2$ rekeszekbe, majd felébresztenek egy-egy segédet stb.

Ha mindannyian kiértek a $[2pz, 2pz + p - 1]$ intervallumból, jöhet a következő szimulálandó P-CRCW lépés.

PRAM gép – P-CRCW EREW szimulációja

Egy m költségű P-CRCW utasítás szimulálásához használt minden egyes EREW utasítás költsége $O(\log p + m)$, mivel a z című rekesz helyett $O(pz)$ méretűeket használunk, így $\log z$ helyett $O(\log p + \log z)$ lesz a címből adódó költség, így a nagyobb című rekeszek használata csak egy $O(\log p)$-s additív taggal növeli a költséget.

A bizonyítás során egy P-CRCW lépést $O(\log p)$ darab EREW lépéssel szimuláltunk, így összességében az elsőbbségi modell egy m költségű lépésének szimulációja $O((\log p)(\log p + m))$ költségű.

Mivel minden $p \geqslant 2$-re $(\log p)(\log p + m) \leqslant (\log^2 p)m$, ezért az elsőbbségi modellbeli lépések szimulálása a konfliktusmentes modellben a költséget egy $O(\log^2 p)$-s szorzóval növeli.

Mivel ez minden lépésről elmondható, ezért a tétel futási időre vonatkozó állítását beláttuk. □

Számítási modellek

7. előadás

A sejtautomata koncepciója

A sejtautomata egy természet inspirálta párhuzamos számítási modell, melyet alkalmazhatunk fizikai, kémiai, biológiai vagy közgazdasági természetes folyamatok modellezésére. Stanislaw Ulam és Neumann János vezették be a 40-es években. Jellemzői:

- egy lokálisan uniform architektúra szerint egymással összeköttetésben levő automaták, ún. „sejtek" alkotják, bármely két sejtnek lokálisan ugyanolyan a szomszédsága (legtöbbször az automaták egy térkitöltő rács rácspontjaiban helyezkednek el),
- a sejtek egyformák és egyszerű felépítésűek, egyetlen jellemzőjük az állapotuk,
- a sejtek csak a szomszédaikkal kommunikálnak, saját és szomszédaik aktuális állapota alapján frissítik az állapotukat,
- a gép szinkron módon, diszkrét időskálán működik.

Rengeteg érdekes anyag található interneten (Wikipedián, YouTubeon, Jarkko J Kari (Turku) honlapján, ...). (angolul: cellular automata)

Sejtautomata

Definíció

$A = \langle X, S, N, f \rangle$ rendezett négyes egy (homogén) **sejtautomata**, ahol

- X egy vektortér végtelen részhalmaza, a **sejttér**, elemeit **sejteknek** nevezzük,
- S egy nemüres, véges halmaz, a **sejtállapotok** halmaza,
- $N = (\mathbf{n_1}, \ldots, \mathbf{n_m})$ rendezett vektor m-es, a **szomszédságvektor**, úgy hogy $\forall \mathbf{x} \in X, 1 \leqslant i \leqslant m$ esetén $\mathbf{x} + \mathbf{n_i} \in X$
- $f : S^m \to S$ a **lokális frissítési szabály**

- Legtöbbször a sejt aktuális állapotától is függ A új állapota, ilyenkor legyen $\mathbf{0} \in N$.

Sejtautomata

- Ha $X = \mathbb{Z}^d$ és $\mathbf{n_i} \in \mathbb{Z}^d$ $(1 \leqslant i \leqslant m)$ akkor a szomszédságvektorra vonatkozó feltétel automatikusan teljesül. Mivel ez egy gyakran előforduló sejttér, ilyenkor röviden $\langle d, S, N, f \rangle$-t fogunk írni.
- Bár legtöbbször az $X = \mathbb{Z}^d$ esettel foglalkozunk, de elképzelhető más rács is. Például az euklideszi síkon egy hatszög- vagy háromszögrács, de tekinthetünk rácsokat a tóruszon, hiperbolikus síkon is.
- Az egyszerűség kedvéért feltettük, hogy a sejtautomata **homogén**, ekkor ugyanis elég egyelten közös lokális frissítési szabályt megadni, amely alapján egy globális frissítést könnyen megadhatunk. A sejtautomata fogalma általánosítható **inhomogén** sejtterekre is, ekkor a lokális frissítési szabályok nem feltétlen egyformák minden sejtre. Ilyenkor persze az se szükséges, hogy a sejttér vektortér legyen, lehet egy tetszőleges gráf.

Sejtautomata

Definíció

Legyen $N = (\mathbf{n_1}, \ldots, \mathbf{n_m})$. Egy $\mathbf{x} \in X$ sejt **szomszédainak** halmaza $N(\mathbf{x}) = \{\mathbf{x} + \mathbf{n_i} \mid 1 \leqslant i \leqslant m\}$

Definíció

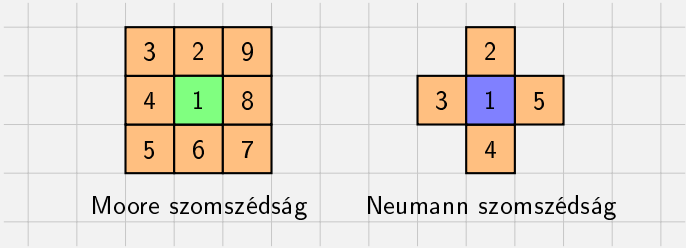
Egy $c : X \to S$ leképezést **konfigurációnak** nevezünk.

Definíció

Legyen $A = \langle X, S, N, f \rangle$ egy sejtautomata ahol $N = (\mathbf{n_1}, \ldots, \mathbf{n_m})$. Ekkor definiálhatunk egy $G : S^X \to S^X$ **globális átmenetfüggvényt**. Legyen $c : X \to S$ egy konfiguráció és $\mathbf{x} \in X$ egy tetszőleges sejt. Ekkor

$$G(c)(\mathbf{x}) := f(c(\mathbf{x} + \mathbf{n_1}), \ldots, c(\mathbf{x} + \mathbf{n_m})).$$

Neumann és Moore szomszédság



Ha $X = \mathbb{Z}^2$, akkor úgy gondolhatunk a sejtekre, mint a fenti ábra celláira. A zöld sejt Moore-szomszédai: 2-9, a kék sejt Neumann-szomszédai 2-5.

Egy sejt új állapota legtöbbször saját maga előző állapotától is függ, ezért a Moore- illetve Neumann szomszédságot így érdemes definiálni:

$N_{\text{Moore}} = ((0,0),(0,1),(-1,1),(-1,0),(-1,-1),(0,-1),$
$(1,-1),(1,0),(1,1))$.

$N_{\text{Neumann}} = ((0,0),(0,1),(-1,0),(0,-1),(1,0))$.

Életjáték

Definíció(Conway)

Életjátéknak (game of life, Life, B3S23) nevezzük azt a $\text{GOL} = \langle 2, \{0,1\}, N_{\text{Moore}}, f \rangle$ sejtautomatát, ahol

1. $f(1, b_2, \ldots, b_9) = 1$, ha $\sum_{i=2}^{9} b_i \in \{2,3\}$
2. $f(0, b_2, \ldots, b_9) = 1$, ha $\sum_{i=2}^{9} b_i = 3$
3. $f(b_1, b_2, \ldots, b_9) = 0$, minden más esetben

Tehát egy élő sejt életben marad, ha 2 vagy 3 élő Moore-szomszédja van, minden más esetben elszigetelődés (0-1 szomszéd) vagy túlnépesedés (4-8 szomszéd) miatt meghal.

Egy halott sejtből akkor és csak akkor lesz élő, ha pontosan 3 élő Moore-szomszédja van.

A B3S23 jelölésben a B=birth, S=stay alive, azaz a születéshez 3, az életben maradáshoz 2 vagy 3 Moore-szomszéd kell.

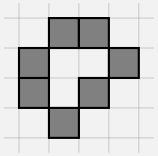
Életjáték – konfigurációtípusok

Definíció

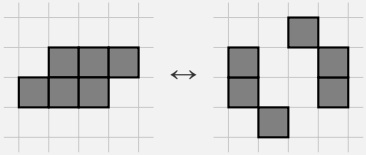
Egy c konfiguráció **orbitja** az $orb(c) = c, G(c), G^2(c), \ldots$ sorozat.

- c **csendélet**, ha $G(c) = c$. A legkisebb csendélet egy 2x2-es blokk.
- c **oszcillátor**, ha $\exists i \geqslant 2$, hogy $G^i(c) = c$ (c időben periodikus). Példa: 3 egymás melletti élő sejt. Ez a legkisebb (2 periódusú) oszcillátor, neve Villogó.
- c **űrhajó**, ha $\exists i \geqslant 1$, hogy $\{\mathbf{x} \in \mathbb{Z}^2 \mid G^i(c)(\mathbf{x}) = 1\} = \{\mathbf{x} \in \mathbb{Z}^2 \mid c(\mathbf{x}) = 1\} + \mathbf{y}$ valamely $\mathbf{y} \in X$-re. A legkisebb méretű űrhajó neve Sikló.
- c **ágyú**, ha $\exists \ell \geqslant 0, k \geqslant 1$ és c' űrhajó, hogy $\forall i \in \mathbb{N}$ $\{\mathbf{x} \in \mathbb{Z}^2 \mid G^{(i+1)k+\ell}(c)(\mathbf{x}) = 1\} \supset \{\mathbf{x} \in \mathbb{Z}^2 \mid G^{ik+\ell}(c)(\mathbf{x}) = 1\}$ és $\{\mathbf{x} \in \mathbb{Z}^2 \mid G^{(i+1)k+\ell}(c)(\mathbf{x}) = 1\} \setminus \{\mathbf{x} \in \mathbb{Z}^2 \mid G^{ik+\ell}(c)(\mathbf{x}) = 1\} = \{\mathbf{x} \in \mathbb{Z}^2 \mid c'(\mathbf{x}) = 1\} + \mathbf{y}_k$ valamely $\mathbf{y}_k \in X$-re.
  (azaz periódikusan c' egy-egy eltoltjával nő a konfiguráció)

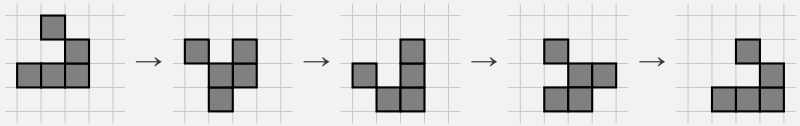
Életjáték – konfigurációtípusok



A Cipó nevű csendélet



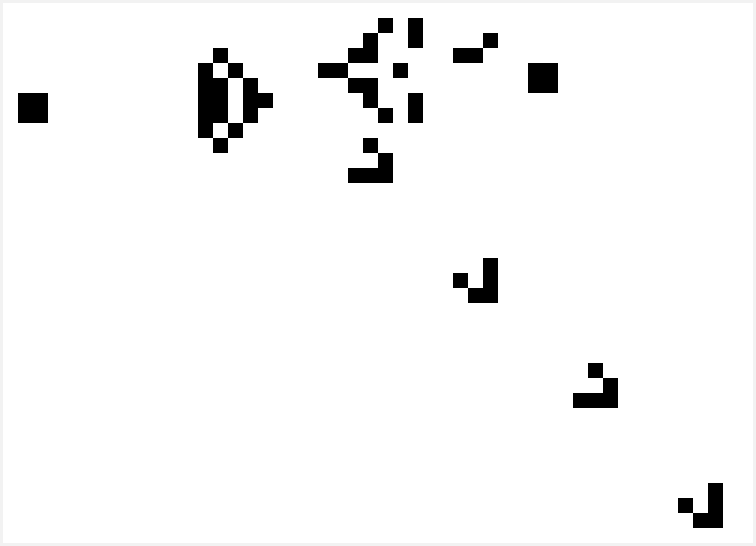
A Varangy nevű oszcillátor



A Sikló nevű űrhajó

Életjáték

Bill Gosper siklóágyúja



Egydimenziós, kétállapotú sejtautomata

Wolfram megvizsgálta az $A = \langle 1, \{0,1\}, (-1,0,+1), f \rangle$ alakú sejtautomatákat.

$f : \{0,1\}^3 \to \{0,1\}$ függvényből 256 lehetőség van. Ezek megfeleltethetők azon 8 hosszúságú bitsorozatoknak, ahol az első bit $f(111)$, a második bit $f(110)$,..., a nyolcadik bit $f(000)$ Ezen a megfeleltetésnek megfelelő decimális számot a sejtautomata **Wolfram-kódjának** nevezzük. A megfelelő sejtautomatára a Wolfram-kódja alapján hivatkozunk (0-tól 255-ig).

Példa: Mik a 30-as számú egydimenziós, kétállapotú sejtautomata szabályai? Írjuk fel binárisan!

111	110	101	100	011	010	001	000
0	0	0	1	1	1	1	0

Tehát például $f(0,1,1) = 1$, azaz ha egy élő sejt baloldali szomszédja halott, jobboldali élő, akkor a sejt életben marad.

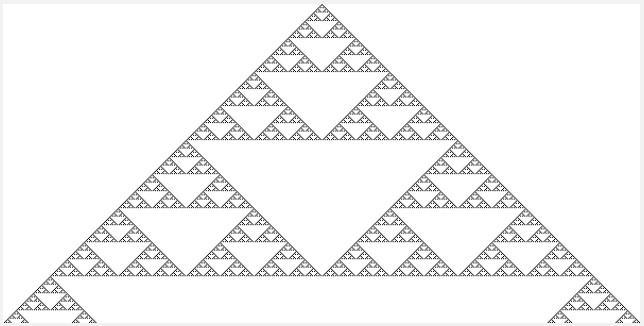
Egydimenziós, kétállapotú sejtautomata

Tér-idő diagram

Az egydimenziós, kétállapotú sejtautomaták orbitját egy 2 dimenziós képpel, az ún. **tér-idő diagram** segítségével ábrázolhatjuk: a kép első sorában ábrázoljuk a c kezdőkonfigurációt, az i-edikben $G^i(c)$-t.

Példa: 90-es szabály

111	110	101	100	011	010	001	000
0	1	0	1	1	0	1	0



Egydimenziós, kétállapotú sejtautomata

Wolfram osztályozása

(W1) Majdnem minden kezdőkonfiguráció orbitja egyazon konstans konfigurációban (minden sejt állapota ugyanaz) stabilizálódik.

Pl. 160	111	110	101	100	011	010	001	000
	1	0	1	0	0	0	0	0

(W2) Majdnem minden kezdőkonfiguráció orbitja periodikus.

Pl. 108	111	110	101	100	011	010	001	000
	0	1	1	0	1	1	0	0

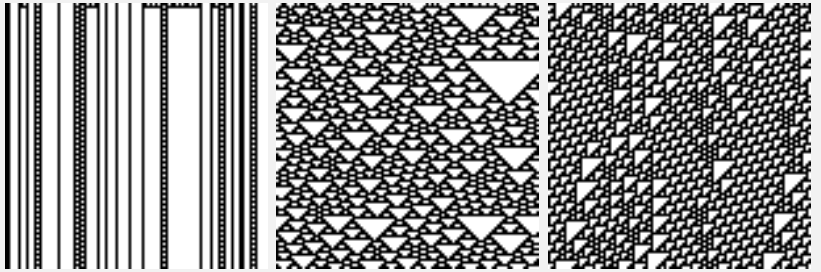
(W3) Majdnem minden kezdőkonfiguráció kaotikus, véletlenszerű viselkedéshez vezet.

Pl. 126	111	110	101	100	011	010	001	000
	0	1	1	1	1	1	1	0

(W4) Lokális struktúrák alakulnak ki komplex kapcsolattal.

Pl. 110	111	110	101	100	011	010	001	000
	0	1	1	0	1	1	1	0

Egydimenziós, kétállapotú sejtautomata

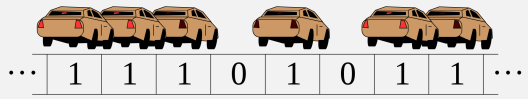


108-as (W2) 126-os (W3) 110-es (W4)

Egydimenziós, kétállapotú sejtautomata

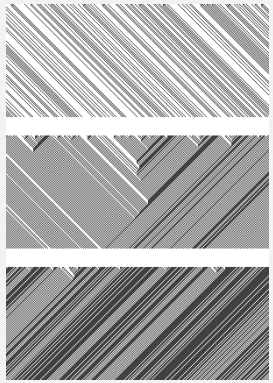
Példa: 184-es szabály

184	111	110	101	100	011	010	001	000
	1	0	1	1	1	0	0	0



⋯	1	1	1	0	1	0	1	1	⋯

- **Közlekedés áramlása.** Az 1-esek autók, melyek akkor lépnek egyet jobbra, ha van előttük egy szabad hely (0). Meglepően jól modellezi a valóságot (folyamatos haladás, stop-go-stop-go, dugó).



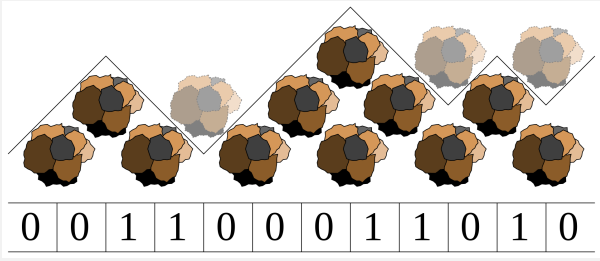
25,50, illetve 75 százalékos autósűrűség

Egydimenziós, kétállapotú sejtautomata

Példa: 184-es szabály

184	111	110	101	100	011	010	001	000
	1	0	1	1	1	0	0	0

- **Részecskék lerakódása szabálytalan felületre** Tekintsünk egy a gravitációval 45 fokos szöget bezáró rácsot. Egy felületet modellezhetünk úgy, hogy minden részecske alatt balra lent és jobbra lent részecske kell legyen. Ennek felülete egy +1 és -1 meredekségű darabokból álló határvonal. A következő iterációban 1-1 új részecske rakódik le a lokális minimum pontokban (10-k fölé)



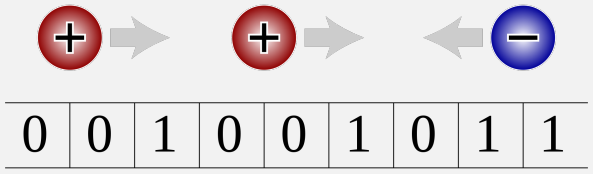
0	0	1	1	0	0	0	1	1	0	1	0

Egydimenziós, kétállapotú sejtautomata

Példa: 184-es szabály

184	111	110	101	100	011	010	001	000
	1	0	1	1	1	0	0	0

- **Ballisztikus kioltás.** A 00 minta egy balról jobbra haladó pozitív töltésű részecskét, míg az 11 minta ennek egy jobbról balra mozgó antirészecske párját reprezentálja. 01 és 10 a köztes térnek felel meg. Az ellentétes töltésű részecskepárok kioltják egymást.



0	0	1	0	0	1	0	1	1

00100101**1**,
1**00**1**00**11**0**,
0**1**00**1**0101,
101**00**1010.

Injektív/szürjektív/bijektív sejtautomata

Egy sejtautomata **injektív/szürjektív/bijektív**, ha G injektív/szürjektív/bijektív.

Azaz egy sejtautomata bijektív, ha minden lehetséges c konfigurációjára teljesül, hogy pontosan egy olyan c' konfiguráció van, melyre $G(c') = c$.

A $p \in S$ sejtállapot **nyugalmi** állapot, ha $f(p, \dots p) = p$. Ha egy c konfigurációban véges sok sejt van a p nyugalmi állapottól különböző állapotban, akkor azt mondjuk, hogy c a sejtautomata egy **véges konfigurációja**.

Észrevétel: Ha c véges, akkor $G(c)$ is az.

Ha ugyanis c-nek k nyugalmitól különböző állapota van, akkor $G(c)$-nek legfeljebb $k|N|$.

Jelölje G_F G-nek a **véges** konfigurációkra való megszorítását.

Reverzibilis sejtautomata

Definíció

Az G globális átmenetfüggvényű sejtautomatát **reverzibilisnek** nevezzük, ha van inverze, azaz egy olyan F globális átmenetfüggvényű sejtautomata, amelyre $F \circ G = G \circ F = id$.

Példa: 1 dimenziós sejtautomata 9 állapottal

Észrevétel: Ha egy sejtautomata revervibilis akkor nyilván bijektív. A fordított állítás nem nyilvánvaló.

Édenkert

Definíció

Egy sejtautomata valamely konfigurációját **édenkert** konfigurációnak nevezzük, ha nincs megelőzője.

Azaz édenkert csak kezdőkonfigurációként fordulhat elő.

Példa:

110	111	110	101	100	011	010	001	000
	0	1	1	0	1	1	1	0

Minden 01010-t tartalmazó c konfiguráció édenkert. Indirekt, tegyük fel, hogy van olyan c', hogy $G(c') = c$. Tekintsük a középső 0-t, három eset van aszerint, hogy mi c'-ben ennek a sejtnek a szomszédsága. (1) 000, ekkor az előző egyes szomszédsága c'-ben $*00$, ami nem lehet. (2) 111, ekkor az előző 1-es miatt c'-ben ez előtt 0 áll, de akkor a minta első 0-ja $*01$-ből lett, ami nem lehet. (3) 100, ekkor a következő 1-es miatt c'-ben 100 után 1-es áll, de akkor a minta utolsó 0-ja $01*$-ból lett, ami nem lehet.

Édenkert

Definíció

Egy véges mintát **árvának** nevezzük, ha minden őt tartalmazó konfiguráció édenkert.

Tehát a fentiek szerint 01010 árva a 110-es szabályú egydimenziós, kétállapotú sejtautomatában.

Tétel

Bármely édenkert tartalmaz árvát.

(nem bizonyítjuk)

Édenkert tétel

Édenkert tétel (Moore (1962), Myhill (1963))

Legyen $A = \langle X, S, N, f \rangle$ egy sejtautomata, ahol X euklideszi tér. Ekkor G szürjektív akkor és csak akkor, ha G_F injektív.

Azaz akkor és csak akkor van árva (édenkert), ha van két véges konfiguráció, melynek ugyanaz a képe. Az ilyen párokat **ikreknek** nevezzük.

Bizonyítás (vázlat): Legyen $|S| = s$. Csak az euklideszi síkon bizonyítjuk.

Tegyük fel, hogy vannak P és Q ikrek és tegyük fel, hogy befoglalhatók egy-egy $n \times n$-es négyzetbe. Tegyük fel továbbá, hogy X elemei szomszédságának legfeljebb n a sugara.

Édenkert tétel

Tekintsünk most egy nagy, $mn \times mn$-es négyzetet. A megelőző konfiguráció egy legfeljebb $(m+2)n \times (m+2)n$-es területének lehet hatása ezen sejtek állapotára. Ez $(m+2) \times (m+2)$ darab $n \times n$-es cellára osztható.

Mivel léteznek ikrek, ezen $n \times n$-es részek $s^{n \times n}$ lehetséges konfigurációja közül legalább 2-nek ugyanaz a hatása. Így legfeljebb $(s^{n \times n} - 1)^{(m+2)(m+2)}$ különböző kép állhat elő a lehetséges $s^{mn \times mn}$ közül.

Utóbbi a nagyobb szám, ha m elég nagy, mivel $\log(s^{n^2})m^2 > \log(s^{n^2} - 1)(m^2 + 4m + 4)$ teljesül, ha m elég nagy.

Tehát van olyan minta az $mn \times mn$-es négyzet lehetséges kitöltései közül, amely nem áll elő képként, azaz van legfeljebb $mn \times mn$ méretű árva.

Édenkert tétel

Fordítva, tegyük fel, hogy létezik egy R árva. Ekkor létezik $n \in \mathbb{N}$, melyre R befoglalható egy $n \times n$-es négyzetbe.

Tekintsük az egy nagy, $mn \times mn$-es négyzetbe foglalható véges konfigurációkat. Ezen konfigurációk következő generációja egy $(m+2)n \times (m+2)n$-es négyzetbe foglalható, melyet osszunk fel $(m+2) \times (m+2)$ darab $n \times n$-es négyzetre.

Mivel egyik ilyenben se kaphatjuk R-et, ezért a potenciálisan $s^{mn \times mn}$ konfigurációnak legfeljebb $(s^{n \times n} - 1)^{(m+2)(m+2)}$ fajta képe lehet. Az előző számítás miatt a skatulyelv szerint van két konfiguráció, amelyeknek ugyanaz a képe, azaz vannak ikrek. □

Megjegyzés: Azt, hogy X euklideszi tér ott használtuk ki, hogy egy nagy térfogatú kockának a felszíne hozzá képest aszimptotikusan kisebb nagyságrendű sejtet tartalmaz (síkban: terület/kerület). Van olyan sejtatomata pl. a hiperbolikus síkon (ami nem euklideszi tér), amelynek van édenkertje, de nincsenek ikrei és olyan is, amelynek vannak ikrei, de nincs édenkertje.

Édenkert tétel

Következmény

Legyen az A sejtautomata sejttere euklideszi tér és globális átmenetfüggvénye G.

- Ha A injektív sejtautomata, akkor szürjektív is.
- Ekvivalens az, hogy A injektív és az, hogy A bijektív.

Bizonyítás: Ha G injektív, akkor G_F is, tehát az Éden-tétel miatt szürjektív is. Minden bijektív leképezés szürjektív. □

Tétel

Legyen az A sejtautomata sejttere euklideszi tér és globális átmenetfüggvénye G. Ekkor

$\boxed{G \text{ injektív} \Leftrightarrow G \text{ bijektív} \Leftrightarrow G \text{ reverzibilis}} \Longrightarrow$

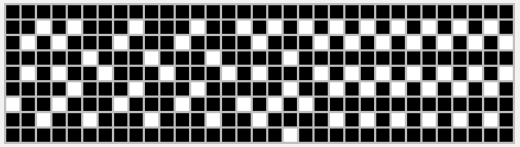
$\boxed{G_F \text{ szürjektív} \Leftrightarrow G_F \text{ bijektív}} \Longrightarrow \boxed{G \text{ szürjektív} \Leftrightarrow G_F \text{ injektív}}$

(nem bizonyítjuk)

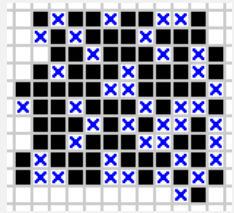
Édenkert tétel

Például GOL-re alkalmazható az édenkert tétel. Egy csupa halott cellából és egy egyetlen élő sejtből álló konfiguráció könnyen láthatóan iker. Ekkor az édenkert-tétel szerint létezik édenkert. GOL-ben azonban nincs kis méretű édenkert.

R. Banks konstrukciója:



A. Flammenkamp árvája: (kék x: kötelezően halott)



Sejtautomaták számítási ereje

Sejtautomata, mint nyelvanalizáló eszköz:

Definíció

Legyen $A = \langle 1, S, (-1, 0, +1), f \rangle$ egy egydimenziós sejtautomata, $T \subset S$, $F \subset S \backslash T$ továbbá $\sqcup \in S \backslash (T \cup F)$ az A nyugalmi állapota. A felismeri az $L \subseteq T^*$ nyelvet, ha $w \in L$ akkor és csak akkor ha az automatát w-vel indítva (értsd: egymás utáni celláin w olvasható, minden más cella nyugalmi állapotában van) a w első betűjének megfelelő cella F-beli állapotba jut.

Tétel

Legyen adott egy M Turing gép n állapottal és m-elemű szalagábécével. Ekkor van olyan egydimenziós sejtautomata, ami

- háromelemű szomszédsággal és $(m+1)n$ állapottal
- háromelemű szomszédsággal és $m+n+2$ állapottal
- hatelemű szomszédsággal és $\max\{n, m\}+1$ állapottal

szimulálni tudja M-et.

Sejtautomaták számítási ereje

Bizonyítás (csak az első):
Legyen $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n \rangle$ egy TG. Megkonstruálunk egy $A = \langle 1, \Gamma \cup Q \times \Gamma, (-1, 0, +1), f \rangle$ egydimenziós sejtautomatát, ami ugyanazt a nyelvet ismeri fel, mint M.

Ha $\delta(q, a) = (r, b, R)$, valamely $q, r \in Q, a, b \in \Gamma$-ra, akkor
$f((q, a), x, y) := (r, x) \quad (\forall x, y \in \Gamma)$
$f(x, (q, a), y) := b \quad (\forall x, y \in \Gamma)$

Ha $\delta(q, a) = (r, b, S)$, valamely $q, r \in Q, a, b \in \Gamma$-ra, akkor
$f(x, (q, a), y) := (r, b) \quad (\forall x, y \in \Gamma)$

Ha $\delta(q, a) = (r, b, L)$, valamely $q, r \in Q, a, b \in \Gamma$-ra, akkor
$f(x, y, (q, a)) := (r, y) \quad (\forall x, y \in \Gamma)$
$f(x, (q, a), y) := b \quad (\forall x, y \in \Gamma)$

Minden egyéb esetben $(x, y, z \in \Gamma \cup Q \times \Gamma)$
$f(x, y, z) := y$.

Sejtautomaták számítási ereje

$T = \Sigma$ és $F = \{q_i\} \times \Gamma$ választással könnyen látható, hogy A-nak éppen az $L(M)$-beli szavakra tartalmaz az orbitja F-beli állapotot.

Ennek a kezdőcellába másolása és a többi cella nyugalmi állapotba vitele technikai részletkérdés, amit itt nem részletezünk. □

Tétel

Tegyük fel, hogy az A egydimenziós sejtautomata által felismert nyelv L. Ekkor készíthető olyan M TG, amelyre $L(M) = L$.

Ennek a bizonyítása egyszerű és természetes, amit a hallgatóságra bízok.

Megjegyzés: Több dimenziós sejtautomaták Turing-univerzalitása szintén bizonyítható.

Életjáték – eldönthetetlen problémák

Végül bizonyítás nélkül megemlítünk néhány algoritmikus probléma megoldhatóságával kapcsolatos eredményt.

Az Univerzális Turing gép szimulálható az Életjátékban.

Tétel

Minden $\langle M, w \rangle$ (Turing gép,szó) párhoz megkonstruálható egy olyan $c_{\langle M,w \rangle}$ kezdőkonfiguráció GOL-ben, melyre $c_{\langle M,w \rangle}$ kihal $\Leftrightarrow \ w \in L(M)$.

Tétel

Adott egy c és egy c' konfiguráció GOL-ben. Eldönthetetlen, hogy létezik-e olyan $i \geqslant 0$, hogy $c' = G^i(c)$.

Tétel

Eldönthetetlen, hogy GOL egy véges mintája kihal-e.

További eldönthetetlen problémák

Tétel (M. Cook, 2004)

A 110-es Wolfram kódú egydimenziós kétálllapotú sejtautomata univerzális, azaz bármely kiszámítható függvény kiszámítható vele.

Tétel

(1) Eldönthető, hogy egy egydimenziós sejtautomata injektív-e.
(2) Eldönthető, hogy egy egydimenziós sejtautomata szürjektív-e.

Tétel

(1) Eldönthetetlen, hogy egy kétdimenziós sejtautomata injektív-e.
(2) Eldönthetetlen, hogy egy kétdimenziós sejtautomata szürjektív-e.
(3) Egy kétdimenziós sejtautomata injektivitása rekurzíve felsorolható.
(4) Egy kétdimenziós sejtautomata nem-szürjektivitása rekurzíve felsorolható.

További eldönthetetlen problémák

Egy konfiguráció **nyugalmi**, ha minden sejt nyugalmi állapotban van. Egy sejtautomata **nilpotens**, ha minden kezdőkonfigurációra nyugalmi konfigurációba jut.

Tétel

(1) Eldönthetetlen, hogy egy egydimenziós sejtautomata nilpotens-e.
(2) Eldönthetetlen, hogy egy kétdimenziós sejtautomata nilpotens-e.
(3) Rekurzíve felsorolható, hogy egy egydimenziós sejtautomata nilpotens-e.
(4) Rekurzíve felsorolható, hogy egy kétdimenziós sejtautomata nilpotens-e.

További eldönthetetlen problémák

Tétel

(1) Eldönthetetlen, hogy egy kétdimenziós sejtautomata reverzibilis-e.
(2) Rekurzíve felsorolható, hogy egy kétdimenziós sejtautomata reverzibilis-e.

Egy G globális átmenetfüggvényű sejtautomata **periodikus**, ha minden kezdőkonfigurációra G időben periodikus.

Tétel

(1) Eldönthetetlen, hogy egy kétdimenziós sejtautomata periodikus-e.
(2) Eldönthetetlen, hogy egy egydimenziós sejtautomata periodikus-e.

Számítási modellek

8. előadás

Membránrendszerek

G. Paun 2000-ben vezette be a később róla elnevezett biológialag inspirált számítási modelljét, a P-rendszereket (membránrendszereket).

Az eukarióta sejtek sejtplazmája több, membránnal határolt sejtalkotót tartalmaz, így belső terekre, ún. régiókra különül. Általánosabban, a többsejtű organizmusok sejtek közötti tere is tekinthető régiónak. A membránok (bizonyos) kémiai molekulák számára átjárhatóak. Az egyes régiókban kémiai reakciók mehetnek végbe, melyek eredményéül kapott (bizonyos) molekulák a régiót határoló membránon áthaladhatnak.

Olyan absztrakt számítási modellt szeretnénk adni, amely alkalmas a molekulák által közvetített sejten belüli és sejtközi információáramlás folyamatának leírására, jobb megértésére.

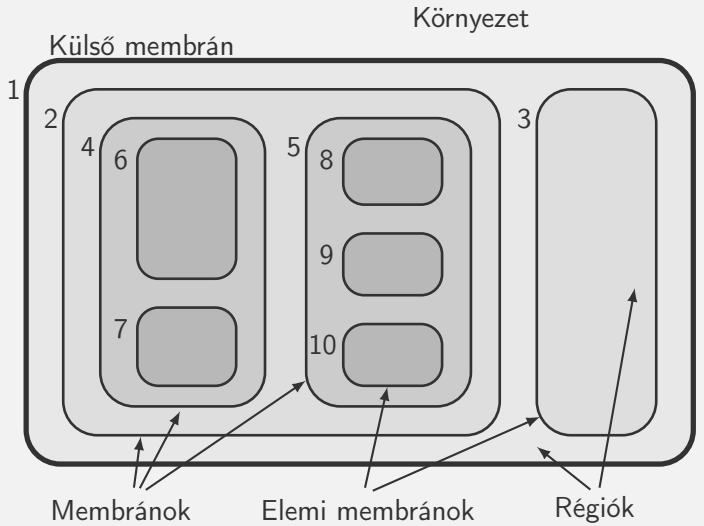
Modellünk ugyanakkor alkalmas lehet absztrakt (nem biokémiai) számítások hatékony elvégzésére is a modellben rejlő masszív párhuzamosság révén.

Membránrendszerek

A membránrendszerek tehát biokémiai folyamatok alapvető törvényszerűségeit szeretnénk modellezni. Ezek közül néhány:

- Az egyes kémiai reakciókhoz szükséges a reakció bemeneti molekuláinak kellő számú jelenléte a régióban.
- Ha a megfelelő mennyiségű nyersanyag rendelkezésre áll, a reakció (vagy ha több reakció lehetséges, ezek bármelyike) végbe is megy.
- Egyszerre több reakció (és/vagy egy reakció többször) is végbemehet ha van mindhez elegendő nyersanyag.
- Bizonyos reakciókhoz szükséges katalizátor molekulák jelenléte a régióban.
- A sejtmembránok feloldódhatnak, ilyenkor a régió tartalma az őt övező régióba kerül.
- Néha két reakció közül biokémiai okok miatt mindig az egyik hajtódik végre, holott az erőforrások mindkét reakcióhoz rendelkezésre állnának.

Hierarchikus membránstruktúra

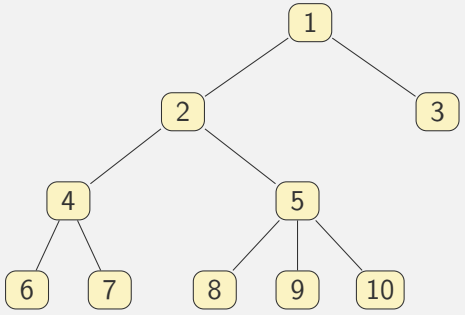


A membránstruktúra:
$[_1\,[_2\,[_4\,[_6\,]_6\,[_7\,]_7\,]_4\,[_5\,[_8\,]_8\,[_9\,]_9\,[_{10}\,]_{10}\,]_5\,]_2\,[_3\,]_3\,]_1$.

Hierarchikus membránstruktúra

Észrevétel: A régiókat azonosíthatjuk a legszűkebb őt tartalmazó membránnal. Így például ha azt mondjuk, hogy egy objektum a 4-es membránban van az úgy értendő, hogy abban a régióban, amit a 4-es membrán, és a 4-es membrán gyerek membránjai határolnak.

A membránok (régiók) hierarchiáját fa alakban is ábrázolhatjuk. Ekkor a levelek az elemi membránok (régiók).



$[_1\,[_2\,[_4\,[_6\,]_6\,[_7\,]_7\,]_4\,[_5\,[_8\,]_8\,[_9\,]_9\,[_{10}\,]_{10}\,]_5\,]_2\,[_3\,]_3\,]_1$.

Multihalmazok

Definíció

Legyen O egy ábécé. Elemeit **objektumoknak** nevezzük. Egy $M : O \to \mathbb{N}$ leképezést az objektumok egy **multihalmazának** nevezünk. Ha $a \in O$, akkor $M(a)$ az a objektum **multiplicitása**.

Egy multihalmaz üres, ha $\forall a \in O : M(a) = 0$. Egy M multihalmazt egy olyan w szóval reprezentálhatunk, melyre $\forall a \in O : |w|_a = M(a)$. Az üres multihalmazt ε reprezentálja.

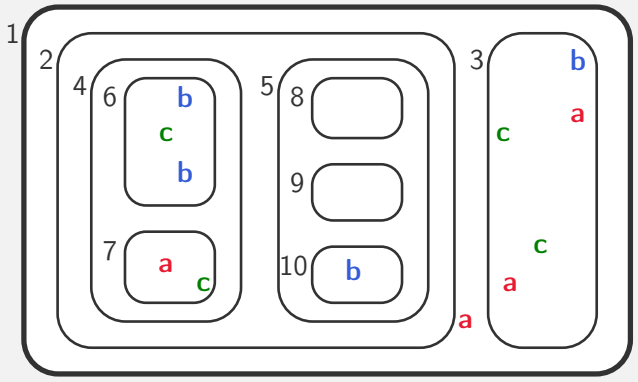
Észrevétel: $aaabab$ és a^4b^2 ugyanazt a multihalmazt reprezentálja.

Legyenek $M_1, M_2 : O \to \mathbb{N}$ két multihalmaz, azt mondjuk, hogy $M_1 \subseteq M_2$, ha $\forall a \in O : M_1(a) \leqslant M_2(a)$. M_1 és M_2 uniója: $\forall a \in O : (M_1 \cup M_2)(a) := M_1(a) + M_2(a)$. Ha $M_1 \subseteq M_2$, akkor M_1 és M_2 különbsége: $\forall a \in O : (M_2 - M_1)(a) := M_2(a) - M_1(a)$.

Példa: $b^4c^2 \subseteq a^3b^5c^3$, de $a^2b \not\subseteq ab^7c^3$.
$b^4c^2 \cup a^3b^5c^3 = a^3b^9c^5$.
$a^3b^5c^3 - b^4c^2 = a^3bc$, viszont $ab^7c^3 - a^2b$ nem értelmezett.

A membránrendszerek konfigurációi

Az egyes régiók objektumok multihalmazait tartalmazhatják.



Az ábrához tartozó konfiguráció:
$[_1\,\mathbf{a}\,[_2\,[_4\,[_6\,\mathbf{bbc}\,]_6\,[_7\,\mathbf{ac}\,]_7\,]_4\,[_5\,[_8\,]_8\,[_9\,]_9\,[_{10}\,\mathbf{b}\,]_{10}\,]_5\,]_2\,[_3\,\mathbf{aabcc}\,]_3\,]_1$.

Alternatív reprezentáció: $(a, \varepsilon, a^2bc^2, \varepsilon, \varepsilon, b^2c, ac, \varepsilon, \varepsilon, b)$, ahol az i. komponens az i. membrán által határolt régió tartalma.

Evolúciós szabályok alkalmazása

Legyenek $\{1, \ldots, m\}$ a régiók címkéi. Minden régióhoz tartoznak $u \to v$ alakú **evolúciós szabályok**, ahol $u \in O^+$ és $v \in (O \times \mathrm{TAR})^*$, ahol TAR=$\{\mathrm{here}, \mathrm{out}\} \cup \{\mathrm{in}_j \mid 1 \leqslant j \leqslant m\}$.

Egy $u \to v$ evolúciós szabályt akkor lehet alkalmazni, ha az u által reprezentált M_1 multihalmazra és a régió aktuális, M multihalmazzal adott tartalmára $M_1 \subseteq M$ teljesül. A szabály alkalmazása a következőt jelenti:

- Vegyük azt a konfigurációt, melyre a szabály régiójának tartalma $M - M_1$, a többi régió tartalma változatlan.
- v minden $(a, \mathrm{tar}) \in O \times$ TAR betűjére adjuk egy a-t
  - tar=here esetén a régióhoz,
  - tar=out esetén a szülő régióhoz (ha gyökér: környezethez),
  - tar=in_j esetén a j címkéjű gyerek régióhoz.

Előfordulhat, hogy in-nek nincs indexe, ilyenkor nemdeterminisztikusan választunk egy gyereket.

Evolúciós szabályok konfliktusai

Észrevétel: A környezetnek nincsenek szabályai, így a környezetbe kijutó objektumok a további számítások számára elvesznek.

Észrevétel: a szabályok párhuzamos alkalmazása csak akkor okozhat konfliktust, ha mindkét szabály ugyanahhoz a régióhoz tartozik és a szabályok baloldalának együttes kielégítéséhez nincs elég nyersanyag.

A maximális párhuzamosság elve:

Motiváció: ha egy kémiai reakció számára minden nyersanyag és környezeti feltétel adott, akkor az a reakció végbe is megy.

Membránrendszerekben a szabályokat mindig **maximális párhuzamossággal** kell alkalmazni, ez azt jelenti, hogy az egy ütemben végrehajtott szabályok multihalmaza egy tartalmazásra nézve maximális halmaz kell legyen, tehát ne legyen hozzáadható a végrehajtott szabályokhoz olyan további szabály, hogy ez a bővebb szabályhalmaz is konfliktusmentesen végrehajtható lett volna.

A maximális párhuzamosság elve

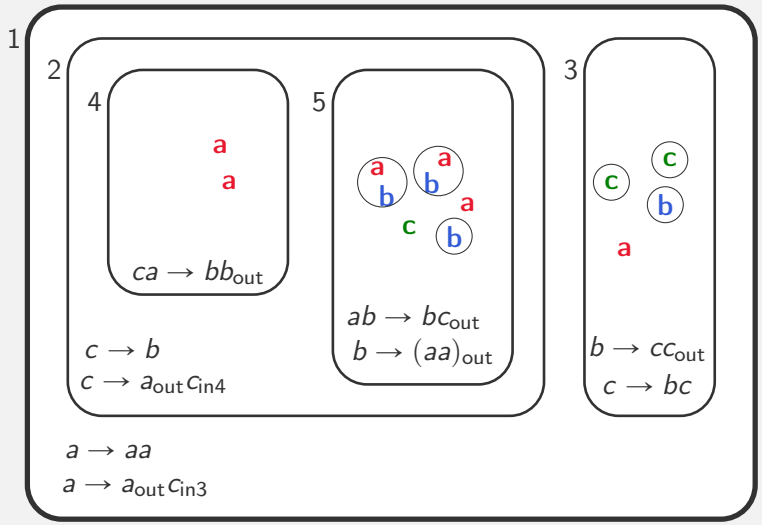
Az evolúció egy lépése alatt azt értjük, hogy minden régióban nemdeterminisztikusan kiválasztjuk az adott régióhoz tartozó szabályok egy olyan multihalmazát, hogy a szabálybaloldalak által reprezentált $M_1, \ldots, M_r$ multihalmazokra és a régió aktuális M multihalmazzal adott tartalmára a következő két feltétel teljesül:

(1) $\bigcup_{i=1}^{r} M_i \subseteq M$, (azaz a kiválasztott szabálymultihalmaz baloldalainak együttes nyersanyagigénye kielégíthető)

(2) ha M' a régió egy tetszőleges további szabályának baloldala által reprezentált multihalmaz, akkor $M' \not\subseteq M - \bigcup_{i=1}^{r} M_i$. (Azaz a kiválasztott szabályokon felül további szabály nyersanyagigénye már nem elégíthető ki a régióból)

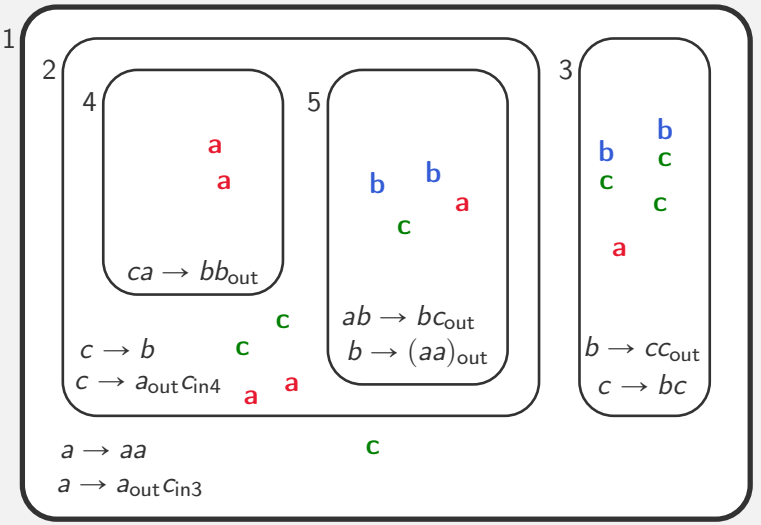
Ezek után minden kiválasztott szabály baloldalát kivonjuk a szabályhoz tartozó régióból, majd a szabályok jobboldalán szereplő objektumokat a szabályok utasítása szerinti régiókhoz hozzáadjuk.

Konfigurációátmenet – példa



$(\varepsilon, \varepsilon, abc^2, a^2, a^3b^3c)$

Konfigurációátmenet – példa



$(\varepsilon, \varepsilon, abc^2, a^2, a^3b^3c) \Rightarrow (c, a^2c^2, ab^2c^3, a^2, ab^2c)$

P-rendszer (membránrendszer – alapmodell)

Definíció

Egy $\Pi = \langle O, \mu, \omega_1, \ldots, \omega_m, R_1, \ldots, R_m, i_o \rangle$ rendezett $(2m+3)$-ast **P-rendszernek** (membránrendszernek) nevezünk, ha

- O egy ábécé (elemeit **objektumoknak** nevezzük).
- μ egy m membránból álló hierarchikus membránstruktúra. A membránok (és így a régiók is) $\{1, 2, \ldots, m\}$ elemeivel injektív módon vannak címkézve. m-et Π **fokának** nevezzük.
- $\omega_1, \ldots, \omega_m$ O feletti multihalmazokat reprezentáló sztringek, ezek rendre az $1, 2, \ldots, m$ címkéjű régióhoz vannak rendelve.
- $R_i, 1 \leqslant i \leqslant m$ μ i-edik membránjához rendelt O feletti **evolúciós szabályok** véges halmaza. A szabályok $u \to v$ alakúak, $u \in O^+$, $v \in (O \times \mathrm{TAR})^*$, ahol $\mathrm{TAR}=\{\mathrm{here}, \mathrm{out}\} \cup \{\mathrm{in}_j \mid 1 \leqslant j \leqslant m\}$.
- $i_o \in \{1, 2, \ldots, m\}$ egy elemi membrán címkéje (**kimeneti membrán**)

A P-rendszer konfigurációi

$C = (v_1, \ldots v_m)$ a $\Pi = \langle O, \mu, \omega_1, \ldots, \omega_m, R_1, \ldots, R_m, i_o \rangle$ P-rendszer **konfigurációja**, ha $v_i \in O^*$ és v_i az i-edik régióban lévő objektum-multihalmaz sztring reprezentációja.

C **kezdőkonfiguráció**, ha $\forall\ 1 \leqslant i \leqslant m$ esetén $v_i = \omega_i$. (Az i-edik régió kezdeti tartalma az ω_i által reprezentált multihalmaz.)

Megállási konfiguráció: Olyan konfiguráció, melyre nem lehet már evolúciós szabályt alkalmazni.

(Egylépéses) **konfigurációátmenet:** $C_1 \Rightarrow_\Pi C_2$, ha C_1-ből egy ütemben megkapható C_2 a korábban definiált maximális párhuzamos evolúciós átírással.

Megjegyzés: Mivel az átírás nemdeterminisztikus, ezért egy C_1-hez több ilyen C_2 is létezhet.

A többlépéses **konfigurációátmenet** a szokásos módon definiáluk, a $\Rightarrow_\Pi$ reláció $\Rightarrow_\Pi^*$-al jelölt reflexív tranzitív lezártjaként.

A P-rendszer számításának eredménye

A P-rendszer egy **számítása** alatt egy a kezdőkonfigurációból egy megállási konfigurációba történő (többlépéses) konfigurációátmenet-sorozatot értünk.

Ha a P-rendszer megállási konfigurációba kerül a **számítás eredménye** a modell különféle változataiban lehet:

- a kimeneti régióban a megálláskor jelen lévő objektumok száma (**alapértelmezett**, mi is ezt tekintjük)
- a kimeneti régióban a megálláskor jelen lévő objektumok vektora. Ha például $O = \{a, b, c\}$ és 2 a, 5 b és 1 c jutott ki, akkor az eredmény (2,5,1), míg az előbb 8 lett volna.
- a rendszert elhagyó objektumok száma
- a rendszert elhagyó objektumok vektora

A Π által **generált nyelv**: A számítások lehetséges eredményeinek a halmaza. Jelölés: $N(\Pi)$.

Alapértelmezett modellben tehát $N(\Pi) \subseteq \mathbb{N}$.

Szabálytípusok

A szabályok kompaktabb reprezentációi:

Példa: $a^2 b \to (a, \mathrm{in})(a, \mathrm{here})(b, \mathrm{here})(b, \mathrm{here})(a, \mathrm{out})(c, \mathrm{out})$ helyett:

$a^2 b \to a b^2 (a, \mathrm{in})(ac, \mathrm{out})$ vagy

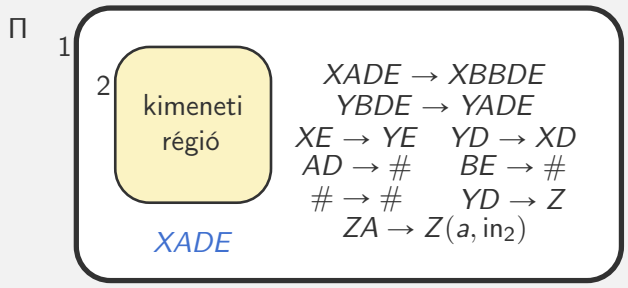
$a^2 b \to a b^2 a_{\mathrm{in}} (ac)_{\mathrm{out}}$

Definíció

Egy $u \to v$ szabály **súlya** alatt $|u|$-t értjük.

Az 1 súlyú szabályokat **nemkooperatívaknak**, a legalább kettő súlyú szabályokat **kooperatívnak** hívjuk.

A P-rendszer számításának eredménye – példa



A 2-es membrán határolja a kimeneti régiót.

Az A-k megduplázása (kezdetben $m = 1$):
$(XA^mDE, \varepsilon) \Rightarrow^* (XB^{2m}DE, \varepsilon) \Rightarrow (YB^{2m}DE, \varepsilon) \Rightarrow^*$
$(YA^{2m}DE, \varepsilon) \Rightarrow (XA^{2m}DE, \varepsilon)$

a-k kiküldése a 2-es régióba:
$(YA^{2m}DE, \varepsilon) \Rightarrow (ZA^{2m}E, \varepsilon) \Rightarrow^* (ZE, a^{2m})$ (megállási konfiguráció)

Tehát $N(\Pi) \supseteq \{2^n \mid n \geqslant 0\}$.

A P-rendszer számításának eredménye – példa

$\#$ egy „csapdajel”. Mivel $\#$ baloldalon egyedül a $\# \to \#$ szabályban szerepel, ezért $\#$-et ha egyszer behozzuk nem tudunk tőle megszabadulni, ráadásul a $\# \to \#$ szabály végtelen ciklusba küldi a membránrendszert, ilyen számítás nem eredményezhet $N(\Pi)$-beli szót.

Ha az XA^mDE $XB^{2m}DE$-re való átírásánál az $XE \to YE$ szabályt még A jelenléténél alkalmaznánk, akkor a maximális párhuzamosság elve és az $AD \to \#$ szabály miatt keletkezne $\#$ objektum.

Ha az $YB^{2m}DE$ $YA^{2m}DE$-re való átírásánál az $YD \to Z$ szabályt még B jelenléténél alkalmaznánk, akkor a maximális párhuzamosság elve és a $BE \to \#$ szabály miatt keletkezne $\#$ objektum.

Tehát csak teljes duplázási ütemek után tudjuk az a-kat úgy a 2-es membránba küldeni, hogy véget is érjen a számítás. Így $N(\Pi) \subseteq \{2^n \mid n \geqslant 0\}$, azaz Π a 2-hatványokat generálja.

Katalitikus P-rendszerek

Definíció

$\Pi = \langle O, K, \mu, \omega_1, \ldots, \omega_m, R_1, \ldots, R_m, i_o \rangle$ **katalitikus P-rendszer** ha

- $\varnothing \neq K \subset O$ az ún. *katalizátorok* halmaza
- $\Pi' := \langle O, \mu, \omega_1, \ldots, \omega_m, R_1, \ldots, R_m, i_o \rangle$ P-rendszer
- Π szabályainak alakja
  - (a) vagy $a \to v$ *(nemkooperatív szabályok)*,
  - (b) vagy $ca \to cv$ *(katalitikus szabályok)*,

  ahol $c \in K, a \in O \backslash K, v \in ((O \backslash K) \times \mathrm{TAR})^*$.

A P-rendszerek számítási ereje

Definíció

$\mathrm{NOP}_m(\alpha) = \{A \subseteq \mathbb{N} \mid A = N(\Pi)$ valamely Π m-edfokú P-rendszerre α típusú szabályokkal$\}$.

α=ncoo: minden szabály nemkooperatív,

α=cat: a P-rendszer katalitikus

α=coo: kooperatív szabályok is megengedettek,

$\mathrm{NOP}_*(\alpha) := \bigcup_{m=1}^{\infty} \mathrm{NOP}_m(\alpha)$

A P-rendszerek számítási ereje

Jelölés: Jelölje a NRE, NCS, illetve NCF azon $A \subseteq \mathbb{N}$ számhalmazok osztályát, melyre A rendre RE, CS illetve CF-beli.

Észrevételek:

- $\mathrm{NOP}_m(\alpha) \subseteq \mathrm{NOP}_{m+1}(\alpha)$, minden $\alpha \in \{\mathrm{coo}, \mathrm{ncoo}, \mathrm{cat}\}$-ra és $m \geqslant 1$-re.
- $\mathrm{NOP}_m(\mathrm{ncoo}) \subseteq \mathrm{NOP}_m(\mathrm{cat}) \subseteq \mathrm{NOP}_m(\mathrm{coo})$, minden $m \geqslant 1$-re.
- $\mathrm{NOP}_*(\mathrm{ncoo}) \subseteq \mathrm{NOP}_*(\mathrm{cat}) \subseteq \mathrm{NOP}_*(\mathrm{coo})$.
- $\mathrm{NOP}_*(\mathrm{coo}) \subseteq \mathrm{NRE}$.

A P-rendszerek számítási ereje

Tétel

$\mathrm{NOP}_m(\alpha) = \mathrm{NOP}_*(\alpha)$ minden $\alpha \in \{\mathrm{coo}, \mathrm{ncoo}, \mathrm{cat}\}$-ra és $m \geqslant 2$-re.

Bizonyítás: (vázlat)

Legyen $\Pi = \langle O, \mu, \omega_1, \ldots, \omega_m, R_1, \ldots, R_m, i_o \rangle$, ahol $R_i = \{r_{i,1}, \ldots r_{i,t_i}\}$ és ha $\alpha = \mathrm{cat}$, akkor K a katalizátorok halmaza.

Konstruálunk egy $\Pi' = \langle O', [_1\, [_{i_o}\,]_{i_o}]_1, \omega, \omega_{i_o}, \bigcup_{i=1, i\neq i_o}^{m} R'_i, R'_{i_o}, i_o \rangle$ 2-fokú P-rendszert (ha $\alpha = \mathrm{cat}$, akkor K' katalizátor halmazzal), melyre $N(\Pi') = N(\Pi)$.

$O' := O \cup \{a_i \mid a \in O, 1 \leqslant i \leqslant m\}$

$K' = K \cup \{c_i \mid c \in K, 1 \leqslant i \leqslant m\}$. (Katalitikus rendszer esetén.)

$h_i : O^* \to (O')^*$ homomorfizmus, melyre $h_i(a) := a_i$ $(a \in O, 1 \leqslant i \leqslant m)$.

$\omega := h_1(\omega_1) \cdots h_{i_o-1}(\omega_{i_o-1}) \omega_{i_o} h_{i_o+1}(\omega_{i_o+1}) \cdots h_m(\omega_m)$.

A P-rendszerek számítási ereje

$i \neq i_o$-ra $R'_i = \{r'_{i,j} : h_i(u) \to v' \mid r_{i,j} : u \to v \in R_i, 1 \leqslant j \leqslant t_i\}$, ahol v'-t úgy kapjuk, hogy minden v-beli

- (b, here)-t b_i-vel helyettesítünk,
- (b, out)-ot b_j-vel helyettesítjük, ahol j közvetlenül tartalmazza i-t,
- (b, in_s)-t b_s-sel helyettesítjük, ha $s \neq i_o$,
- (b, in_{i_o}) változatlan.

$R'_{i_o} = \{r'_{i_o,j} : u \to v' \mid r_{i_o,j} : u \to v \in R_{i_o}, 1 \leqslant j \leqslant t_{i_o}\}$, ahol v'-t úgy kapjuk, hogy minden v-beli

- (b, here) változatlan,
- (b, out)-ot $(b_j$, out)-tal helyettesítjük, ahol j i_o szülő membránja.

Meggondolható, hogy $N(\Pi') = N(\Pi)$ és nemkooperatív rendszer képe nemkooperatív, katalitikus képe katalitikus.

A P-rendszerek számítási ereje

Tétel

$\mathrm{NOP}_*(\mathrm{ncoo}) = \mathrm{NOP}_m(\mathrm{ncoo}) = \mathrm{NCF}$, minden $m \geqslant 1$-re.

Tétel

$\mathrm{NOP}_*(\mathrm{coo}) = \mathrm{NOP}_m(\mathrm{coo}) = \mathrm{NRE}$, minden $m \geqslant 1$-re.

(Bizonyítások nélkül.)

Feloldódás

Megengedünk a szabályok között $u \to v\delta$ alakú szabályokat, ahol $u \to v$ evolúciós szabály és δ egy speciális szimbólum. Egy evolúciós lépés két részből áll. Először az esetleges δ-kat nem figyelembe véve a szokásos evolúciós lépés (a kiválasztott szabályokkal és objektumokkal) végrehajtódik. Ezután, ha egy vagy több alkalmazott szabályban szerepel a δ, akkor ezen szabályokhoz tartozó membránok feloldódnak. Ilyenkor a határolt régió tartalma (**objektumok és membránok igen**, **a szabályok nem!**) a membránt közvetlenül tartalmazó (szülő) régióba kerül.

Ilyenkor i_o nem feltétlenül kell elemi membrán legyen.
A külső membrán (skin) sosem oldódhat fel.

A konfigurációk első komponense legyen az aktuális membránstruktúra (mivel ez változhat). Példa: $([_1[_4]_4]_1, a^2b, ca)$

Alternatív jelölés: maradjon az eredeti rendezett m-es, ahol m a P-rendszer foka. Az eredeti membránstruktúra hiányzó membránjai helyére írjunk egy speciális szimbólumot, például δ-t.

Feloldásos P-rendszerek számítási ereje

Jelölje $\mathrm{NOP}_m(\alpha, \delta)$ a természetes számok halmazainak családját, amelyeket egy legfeljebb m-edfokú ($m \geqslant 1$) feloldással kiegészített P-rendszer generál, $\alpha \in \{coo, cat, ncoo\}$ típusú szabályokat használva.

A feloldással kiegészített P-rendszernek megnő a számítási ereje a nemkooperatív esetben.

Tétel

$\mathrm{NCF} = \mathrm{NOP}_*(\mathrm{ncoo}) \subset \mathrm{NOP}_2(\mathrm{ncoo},\delta) \subseteq \mathrm{NOP}_*(\mathrm{ncoo},\delta) \subset \mathrm{NCS}.$

(Bizonyítás nélkül.)

P-rendszer prioritással

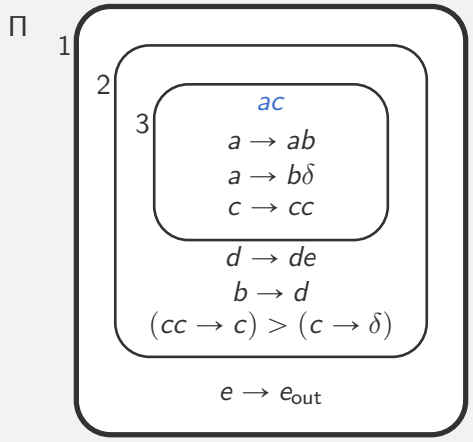
Definíció

$\Pi = \langle O, \mu, \omega_1, \ldots, \omega_m, (R_1, \varrho_1), \ldots, (R_m, \varrho_m), i_o \rangle$ egy **P-rendszer prioritással**, ha $\langle O, \mu, \omega_1, \ldots, \omega_m, R_1, \ldots, R_m, i_o \rangle$ P-rendszer és ϱ_i részbenrendezés R_i-n ($1 \leqslant i \leqslant m$).

Megjegyzés: Mivel az egyes részbenrendezéseknek nincs egymásra közvetlen hatása, használhatjuk a közös $<$ jelet. Azaz $(r_1, r_2) \in \varrho_i$, helyett írhatunk $r_1 < r_2$-t.

A szabályok alkalmazása: Ha $r_1 < r_2$ akkor az r_1 szabály csak abban az esetben alkalmazható, ha r_2 már nem. (Egy objektumot csak akkor lehet az r_1 szabályhoz rendelni, ha r_2-höz már nem.)

P-rendszer feloldódással és prioritással– példa



Négyzetszámokat a környezetbe generáló P-rendszer $N(\Pi) = \{n^2 \mid n \geqslant 1\}$

$(\varepsilon, \varepsilon, ac) \Rightarrow (\varepsilon, \varepsilon, abc^2) \Rightarrow (\varepsilon, b^2c^4, \delta) \Rightarrow (\varepsilon, d^2c^2, \delta) \Rightarrow$
$(\varepsilon, d^2e^2c, \delta) \Rightarrow (d^2e^4, \delta, \delta) \Rightarrow (d^2, \delta, \delta)$, környezetbe: 4 objektum.

$(\varepsilon, \varepsilon, ac) \Rightarrow (\varepsilon, \varepsilon, abc^2) \Rightarrow (\varepsilon, \varepsilon, ab^2c^4) \Rightarrow (\varepsilon, b^3c^8, \delta) \Rightarrow$
$(\varepsilon, d^3c^4, \delta) \Rightarrow (\varepsilon, d^3e^3c^2, \delta) \Rightarrow (\varepsilon, d^3e^6c, \delta) \Rightarrow (d^3e^9, \delta, \delta) \Rightarrow$
(d^3, δ, δ), környezetbe: 9 objektum.

Prioritásos P-rendszerek számítási ereje

Jelölje $\mathrm{NOP}_m(\alpha, \mathrm{pri})$ a természetes számok halmazainak családját, amelyeket egy legfeljebb m-edfokú $(m \geqslant 1)$ prioritással kiegészített P-rendszer generál, $\alpha \in \{coo, cat, ncoo\}$ típusú szabályokat használva.

Ha a prioritásos szabályok mellett feloldódás is megengedett, akkor $\mathrm{NOP}_m(\alpha, \delta, \mathrm{pri})$ jelöli az ilyen P-rendszerekkel generálható természetes szám halmazok családját.

Tétel

$\mathrm{NOP}_2(\mathrm{cat}, \mathrm{pri}) = \mathrm{NRE}$.

(Bizonyítás nélkül.)

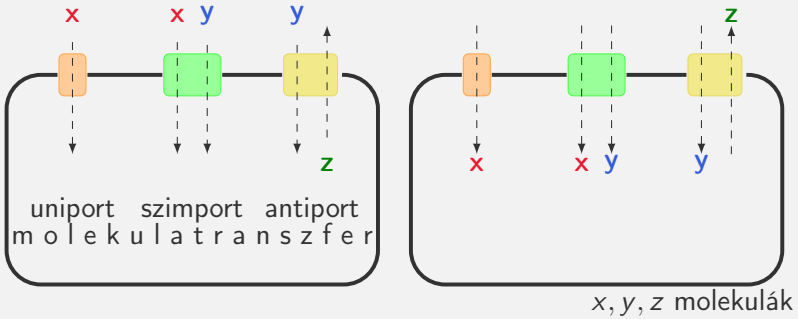
Számítási modellek

9. előadás

Szimport/antiport P-rendszerek

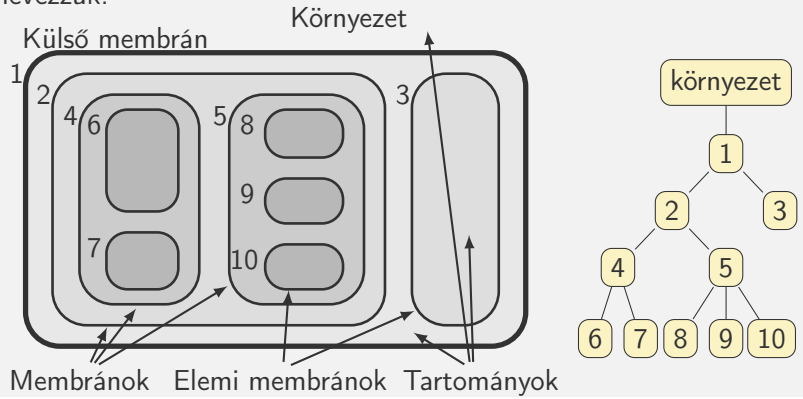
A szimport/antiport P-rendszer egy olyan bio-inspirált számítási modell, melyben az objektumok nem íródhatnak át, csak mozognak a hierarchikus membránstruktúra membránjai által elválasztott különböző régiók között.

Motiváció: Egy élő sejt membránja gátolja a molekulák szabad mozgását. Ugyanakkor lehetőség van szállítóproteinek segítségével különféle fehérjetranszferekre. A különféle transzferek lehetnek uniport, szimport, antiport típusúak.



Szimport/antiport P-rendszerek – alapfogalmak

A **membránok** fa struktúrájú hierarchiát alkotnak. Egyrészt minden membránt – a **külső membrán** kivételével – egyetlen másik membrán által határolt **régió** vesz körül, másrészt őmaga több membránt is tartalmazhat. Amennyiben egyet se tartalmaz **elemi membránnak** nezezzük. A külső membránt a **környezet** veszi körül. A régiókat és a környezetet együtt **tartományoknak** nevezzük.



Szimport/antiport P-rendszerek – szabályok

Legyenek $x, y \in O^+$ multihalmazokat reprezentáló objektumokból álló sztringek. Az objektumok tartományok közötti mozgása az alábbi szabályok szerint történhet:

- (x, in): az x multihalmaz a membránt körülvevő tartományból a membrán által határolt tartományba mozog (szimport szabály). A szabály súlya $|x|$.
- (x, out): az x multihalmaz a membrán által határolt tartományból a membránt körülvevő tartományba mozog (szimport szabály). A szabály súlya $|x|$.
- $(x, \text{in}; y, \text{out})$: az x multihalmaz a membránt körülvevő tartományból a membrán által határolt tartományba, míg az y multihalmaz a membrán által határolt tartományból a membránt körülvevő tartományba mozog (antiport szabály). A szabály súlya $\max\{|x|, |y|\}$.

Megjegyzés: Az uniport molekulatranszferek leírhatók a szimport szabályok speciális eseteként. (x, in), ahol $|x| = 1$.

Szimport/antiport P-rendszerek

Definíció

A **szimport/antiport P-rendszer** egy rendezett $\Pi = \langle O, \mu, E, \omega_1, \ldots, \omega_m, R_1, \ldots, R_m, i_o \rangle$ $(2m+4)$-es, ahol

- O egy ábécé (elemeit **objektumoknak** nevezzük).
- μ egy m membránból álló hierarchikus membránstruktúra. A membránok (és így a régiók is) $\{1, 2, \ldots, m\}$ elemeivel injektív módon vannak címkézve. m-et Π **fokának** nevezzük.
- $\omega_1, \ldots, \omega_m$ O feletti multihalmazokat reprezentáló sztringek, ezek rendre az $1, 2, \ldots, m$ címkéjű régióhoz vannak rendelve.
- $E \subseteq O$ a környezetben **korlátlanul** rendelkezésre álló objektumok halmaza.
- $R_i, 1 \leqslant i \leqslant m$ μ i-edik membránjához rendelt szimport/antiport szabályok véges halmaza.
- $i_o \in \{1, 2, \ldots, m\}$ egy elemi membrán címkéje (**kimeneti membrán**)

Szimport/antiport P-rendszerek konfigurációi

Definíció

$(w_0, w_1, \ldots, w_m)$ a $\Pi = \langle O, \mu, E, \omega_1, \ldots, \omega_m, R_1, \ldots, R_m, i_o \rangle$ szimport/antiport rendszer **konfigurációja**, ahol $w_0 \in O^*$ a környezet $(O - E)$-beli objektumainak multihalmazát, $w_1, \ldots, w_m \in O^*$ pedig rendre az $1, \ldots, m$ régióbeli objektumok multihalmazait reprezentáló sztringek.

$(w_0, w_1, \ldots, w_m)$ **kezdőkonfiguráció**, ha $w_0 = \varepsilon$ és $w_i = \omega_i$ $(1 \leqslant i \leqslant m)$.

A szabályokat **maximálisan párhuzamos** módon kell alkalmazni. Ez pontosabban a következőket jelenti.

Szimport/antiport P-rendszer egylépéses konfigurációátmenete

Legyen M_i a w_i által reprezentált multihalmaz $(0 \leqslant i \leqslant m)$, M_0-hoz adjuk hozzá az E-beli objektumokat végtelen multiplicitással.

Egylépéses konfigurációátmenet: nemdeterminisztikusan válasszuk ki az alkalmazandó szabályok egy olyan véges $\mathcal{R}$ multihalmazát, amelyre a következő 2 feltétel teljesül.

(1) Az $\mathcal{R}$-beli szabályok együttesen alkalmazhatók, azaz az $\mathcal{R}$-beli szabályokra egymás után elvégezhetők az alábbi multihalmaz-kivonások:

- ha $(y,\text{out}) \in R_i \cap \mathcal{R}$ vagy $(x, \text{in}\ ;\ y,\text{out}) \in R_i \cap \mathcal{R}$: vonjuk ki az y által reprezentált multihalmazt M_i-ből
- ha $(x,\text{in}) \in R_j \cap \mathcal{R}$ vagy $(x, \text{in}\ ;\ y,\text{out}) \in R_j \cap \mathcal{R}$ ahol j az i. tartomány μ szerinti egyik gyereke: vonjuk ki az x által reprezentált multihalmazt M_i-ből

  [álljon a környezet a hierarchia csúcsán és legyen a külső membrán által határolt régió az ő egyetlen gyereke]

Szimport/antiport P-rendszer egylépéses konfigurációátmenete

(2) $\mathcal{R}$ nem bővíthető egyetlen szabály hozzáadásával sem, azaz ha M_i' $(0 \leqslant i \leqslant m)$ az iménti kivonások után kapott multihalmazok, akkor

- ha $(y,\text{out}) \in R_i \backslash \mathcal{R}$ akkor az y által reprezentált multihalmaz nem vonható ki M_i'-ből
- ha $(x,\text{in}) \in R_j \backslash \mathcal{R}$ és j az i. tartomány μ szerinti egyik gyereke akkor az x által reprezentált multihalmaz nem vonható ki M_i'-ből
- ha $(x,\text{in};\ y,\ \text{out}) \in R_j \backslash \mathcal{R}$ és j az i. tartomány μ szerinti gyereke akkor vagy az x által reprezentált multihalmaz nem vonható ki M_i'-ből vagy az y által reprezentált multihalmaz nem vonható ki M_j'-ből

Szimport/antiport P-rendszer egylépéses konfigurációátmenete

Az egylépéses konfigurációátmenet eredménye.

- ha $(y,\text{out}) \in R_i \cap \mathcal{R}$ akkor az y által reprezentált multihalmazt uniózzuk hozzá az i. régiót közvetlenül tartalmazó tartományban lévő multihalmazhoz
- ha $(x,\text{in}) \in R_j \cap \mathcal{R}$ és j az i. tartomány μ szerinti gyereke akkor az x által reprezentált multihalmazt uniózzuk hozzá M'_j-höz
- ha $(x,\text{in};\ y,\ \text{out}) \in R_j \cap \mathcal{R}$ és j az i. tartomány μ szerinti gyereke akkor az x által reprezentált multihalmazt uniózzuk hozzá M'_j-höz, míg az y által reprezentált multihalmazt uniózzuk hozzá M'_i-höz.

Észrevétel: Ha a külső membrán egy (x,in) szimport szabályára $x \in E^+$ teljesül, akkor nem választható ki a feltételeknek eleget tevő véges maximális szabály-multihalmaz. Tehát feltehető, hogy ilyen szabályok nincsenek.

Szimport/antiport P-rendszerek számításai

Többlépéses konfigurációátmenet: A egylépéses konfigurációátmenet reflexív, tranzitív lezártja.

Jelölés: $\Rightarrow_\Pi$ jelöli az egylépéses, $\Rightarrow_\Pi^*$ a többlépéses konfigurációátmenet relációt.

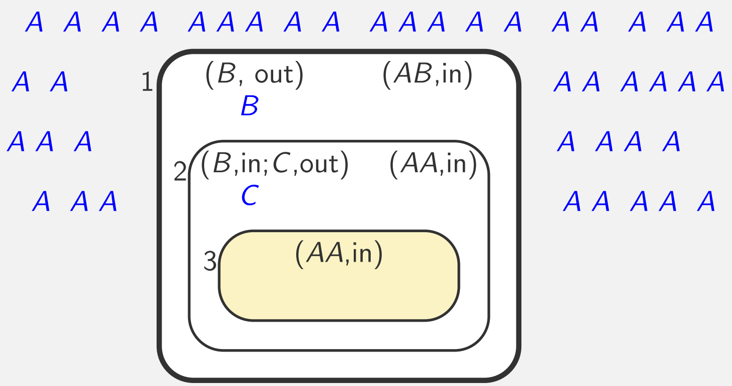
Megállási konfiguráció: olyan konfigurácó, amelyre további szabály nem alkalmazható.

A Π szimport/antiport rendszer egy (termináló) **számítása** egy többlépéses konfigurációátmenet sorozat a kezdőkonfigurációból valamely megállási konfigurációba.

A Π szimport/antiport rendszer egy számításának **eredménye** bármely megállási konfiguráció esetén az i_o kimeneti membránban lévő objektumok száma. (*Alternatív eredményszámítás:* az objektumok (objektumtípus szerinti) számvektora.)

A Π által **generált nyelv** a lehetséges termináló számítások eredményeinek $N(\Pi)$ halmaza. Nyilván $N(\Pi) \subseteq \mathbb{N}$.

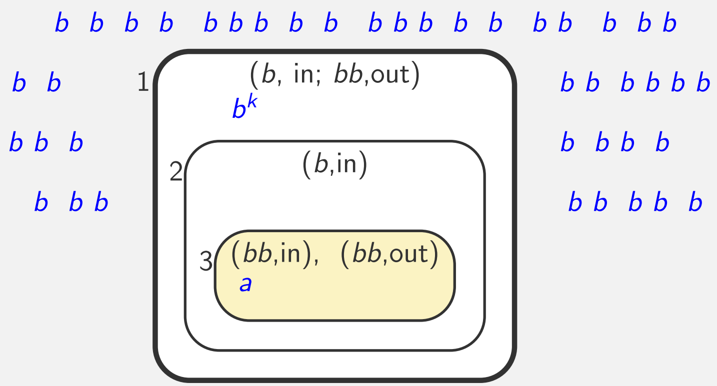
Szimport/antiport P-rendszer – 1. példa



Egy lehetséges számítás:

$(\varepsilon, B, C, \varepsilon) \vdash (B, \varepsilon, C, \varepsilon) \vdash (\varepsilon, AB, C, \varepsilon) \vdash (B, A, C, \varepsilon) \vdash$
$(\varepsilon, AAB, C, \varepsilon) \vdash (B, \varepsilon, AAC, \varepsilon) \vdash (\varepsilon, AB, C, AA) \vdash$
$(B, A, C, AA) \vdash (\varepsilon, AAB, C, AA) \vdash (\varepsilon, C, AAB, AA) \vdash$
$(\varepsilon, C, B, AAAA)$
$N(\Pi) = \{2n \mid n \in \mathbb{N}\}$.

Szimport/antiport P-rendszer – 2. példa



Példa: $(\varepsilon, b^{12}, \varepsilon, a) \vdash (\varepsilon, b^6, \varepsilon, a) \vdash (\varepsilon, b^3, \varepsilon, a) \vdash (\varepsilon, b, b, a) \vdash$
$(\varepsilon, \varepsilon, b^2, a) \vdash (\varepsilon, \varepsilon, \varepsilon, ab^2) \vdash (\varepsilon, \varepsilon, b^2, a) \vdash \ldots$ (nem áll meg)

Pontosan akkor van leálló számítás, ha k 2-hatvány. Más k-ra a maximális párhuzamosság elve miatt legalább két b mindig bekerül a 2-es membránba, ezek ki-be fognak közlekedni a 3-as membránon.

$N(\Pi) = \{1\}$, ha k 2-hatvány, $N(\Pi) = \emptyset$, ha k nem 2-hatvány.

Szimport/antiport P-rendszerek foka és súlya

A **szimport/antiport rendszer foka** a régióinak száma.

A **szimport/antiport rendszer súlya** a rendszerben szereplő szimport és antiport szabályok maximális súlyai által adott számpár.

Példa: Az előző példa egy (2,1) súlyú 3-adfokú rendszer.

Ha a rendszer nem tartalmaz szimport illetve antiport szabályt, akkor a megfelelő maximumot 0-nak értelmezzük.

$\mathrm{NOP}_m(\mathrm{sym}\ p, \mathrm{anti}\ q)$ jelöli a természetes számok azon halmazainak családját, amelyeket egy legfeljebb m-edfokú ($m \geqslant 1$) legfeljebb p súlyú szimport és legfeljebb q súlyú antiport szabályokat használó szimport/antiport P-rendszer generál.

Ha az m, p, q paraméterek valamelyike nem korlátos, akkor a megfelelő paraméter helyére $*$-ot írunk.

Szimport/antiport P-rendszerek univerzalitása

NFIN:=$\{S \subseteq \mathbb{N} \mid S$ véges$\}$.

Tétel

$\mathrm{NOP}_1(\mathrm{sym}\ 1, \mathrm{anti}\ 1) \subseteq \mathrm{NFIN}$

Tétel

$\mathrm{NOP}_1(\mathrm{sym}\ 2, \mathrm{anti}\ 0) \subseteq \mathrm{NFIN}$

Tétel (Frisco, Hoogeboom,2004)

$\mathrm{NOP}_1(\mathrm{sym}\ 1, \mathrm{anti}\ 2) = \mathrm{NRE}$

Tétel (Frisco, Hoogeboom,2004)

$\mathrm{NOP}_3(\mathrm{sym}\ 2, \mathrm{anti}\ 0) = \mathrm{NRE}$

Tétel (Martín-Vide, A. Păun, Gh. Păun, 2002)

$\mathrm{NOP}_2(\mathrm{sym}\ 3, \mathrm{anti}\ 0) = \mathrm{NRE}$

Szimport/antiport P-rendszerek univerzalitása

NRE':=$\{S \backslash \{0\} \mid S \in$ NRE$\}$

Tétel (Freund, A. Păun, 2002)

$\mathrm{NOP}_3(\mathrm{sym}\ 0, \mathrm{anti}\ 2) = \mathrm{NRE}'$.

Megjegyzés: Egyedül antiport szabályokkal nem kapható meg a 0, kivéve ha a kimeneti membránnak nincs szabálya, de akkor meg csak a 0 kapható meg.

Tétel (Vaszil Gy., 2004)

$\mathrm{NOP}_3(\mathrm{sym}\ 1, \mathrm{anti}\ 1) = \mathrm{NRE}$

(ezeket a tételeket nem bizonyítjuk)

Aktív membránrendszerek szabályai

Ebben a számítási modellben a membránoknak lehet egy $\alpha \in \{+, 0, -\}$ **töltése**. A h. membrán α töltöttségét $[_h]_h^\alpha$ jelöli, a semleges, 0 töltést gyakran nem írjuk ki. A szabályok típusai a következők ($a, b \in V, v \in V^*, \alpha, \alpha_1, \ldots \in \{+, 0, -\}, h, h_1, \ldots$ membráncímkék)

(a) $[_h\, a \to v\,]_h^\alpha$

Membránon belüli evolúciós szabály. Nincs hatása a membránra.

(b) $a\, [_h\,]_h^{\alpha_1} \to [_h\, b\,]_h^{\alpha_2}$

Befelé irányuló kommunikációs szabály. Objektum bevitele a h. membránon át, megváltozhat maga az objektum illetve a membrán töltése, a membrán címkéje változatlan.

(c) $[_h\, a\,]_h^{\alpha_1} \to [_h\,]_h^{\alpha_2}\, b$

Kifelé irányuló kommunikációs szabály. Objektum kivitele a h. membránon át, megváltozhat maga az objektum illetve a membrán töltése, a membrán címkéje változatlan.

Aktív membránrendszerek szabályai

(d) $[_h\, a\,]^{\alpha}_h \to b$

Feloldó szabály. Az objektummal való reakció következtében a membrán feloldódhat, ilyenkor a reakció hatására esetleg megváltozott objektum minden egyéb feloldott membránbeli objektummal a szülő membránba kerül.

(e) $[_h\, a\,]^{\alpha_1}_h \to [_h\, b\,]^{\alpha_2}_h [_h\, c\,]^{\alpha_3}_h$

Elemi membránok kettéosztására vonatkozó szabály. Az objektummal való reakció eredményeképpen a membrán két membránná válik szét, a membránok címkéi nem változnak, a polarizációjuk viszont módosulhat.

A membrán minden más objektuma 1-1 példányban átmásolódik az új membránokba. (A szabályban specifikált objektumot lehetőleg új objektumokkal helyettesítjük a két új membránban.)

Megjegyzés: Néha elemi membránok d-felé ($d \geqslant 2$) osztódására vonatkozó szabályokat is megengedünk.

Aktív membránrendszerek szabályai

(f) $[_h\ [_{h_1}]^{+}_{h_1} \cdots [_{h_k}]^{+}_{h_k}\ [_{h_{k+1}}]^{-}_{h_{k+1}} \cdots [_{h_\ell}]^{-}_{h_\ell}\ ]^{\alpha_1}_h \to$
$[_h\ [_{h_1}]^{\alpha}_{h_1} \cdots [_{h_k}]^{\alpha}_{h_k}\ ]^{\alpha_2}_h\quad [_h\ [_{h_{k+1}}]^{\beta}_{h_{k+1}} \cdots [_{h_\ell}]^{\beta}_{h_\ell}\ ]^{\alpha_3}_h$
$(0 < k < \ell, \alpha, \beta, \alpha_1, \alpha_2, \alpha_3 \in \{+, 0, -\}, h, h_1, \ldots, h_\ell \in H.)$

Nem elemi membránok polarizáció alapján történő kettéosztására vonatkozó szabály.

A szabály alkalmazása: h-ban lehetnek $[_{h_{\ell+1}}]^{0}_{h_{\ell+1}} \cdots [_{h_n}]^{0}_{h_n}$ további 0 polaritású membránok és objektumok, ilyenkor ezek az osztódásnál **mindkét** új membránba bekerülnek.

Az ellentétes polarizációjú membránok két új membránba kerülnek, polarizációjuk megváltozhat. Ellentétes polarizációjú membránokat **(f)** típusú szabály alkalmazásával lehet elkülöníteni.

Észrevétel: Vegyük észre, hogy ebben a számítási modellben a membránstruktúra nem állandó és a membránok címkéi nem egyediek (lásd **(e)** és **(f)**), előfordulhat, hogy egy adott szabályt az aktuális membránstruktúra több pontján is alkalmazni lehet.

Aktív membránrendszerek

Definíció

Aktív P rendszernek nevezzük a $\Pi = \langle V, H, \mu, \omega_1, \ldots, \omega_m, R \rangle$ ($m \geqslant 1$) konstrukciót, ahol

- V objektumok nemüres, véges halmaza, a rendszer ábécéje;
- H a membránok címkéinek véges halmaza;
- μ a membránstruktúra, amely m membránból áll, ahol a membránok $1, 2, \ldots, m$ elemeivel nem feltétlenül injektív módon vannak címkézve; feltesszük, hogy μ minden membránja semleges polarizációjú (töltésű);
- $\omega_i, 1 \leqslant i \leqslant m$, olyan V feletti sztringek, amelyek objektumokból álló multihalmazokat reprezentálnak és μ m darab régiójához vannak rendelve;
- R a fenti (a)-(f) típusú szabályokból álló szabályrendszer.

Aktív P-rendszerek – maximális párhuzamosság

A szabályok alkalmazása nemdeterminisztikus módon maximálisan párhuzamos működési módban történik, melyet a következőképpen értelmezünk:

Minden egyes membrán és minden egyes objektum konfigurációátmenetenként **csak egyetlen** szabályban vehet részt.

Tehát membránon belül az objektumok szokásos módon, maximális párhuzamossággal evolválódnak az **(a)** típusú szabályok szerint, ezen szabályok alkalmazásával ugyanis nem halad át egyetlen objektum sem membránon.

Minden egyes membrán a **(b)**-**(f)** típusú szabályok közül összesen csak egyszer lehet érintett.

Egy ilyen egylépéses konfigurációátmenetben a kijelölt szabályok végrehajtását az **(a)** típusú evolúciós szabályokkal kell kezdeni. Ez a feloldódó membránok miatt fontos. Bottom-up végrehajtás van, a membránhierarchiában a levelektől a gyökérig.

Aktív P-rendszerek – maximális párhuzamosság

Nem létezhet olyan szabály, melyet az adott lépésben alkalmazott szabály-multihalmazhoz hozzávéve a fentieket nem sértő szabály-multihalmazt kapnánk. Azaz, ha további objektumok tudnának még membránon belül evolválódni, vagy egy újabb membránnal történhetne (b)-(f) típusú lépés, akkor annak végre is kell hajtódnia.

Amikor egy membrán feloldódik, akkor a megszűnő membrán szabályai nem öröklődnek a szülő membránra. Az R szabályrendszer a működés során nem változik.

A legkülső membrán se osztódni, se feloldódni nem tud, de töltése lehet.

Aktív P-rendszerek – maximális párhuzamosság

1. Példa: A h membránon belül három a objektum van, ezek közül kettő az $[_h a \to b]^0_h$ szabállyal előbb b-vé evolválódhatnak, majd ugyanezen lépésben a $[_h a]^0_h \to c$ szabállyal a h membrán még feloldódhat (a két b és a c a szülő membránba kerül), vagy a $[_h a]^0_h \to [_h b]^0_h [_h c]^0_h$ szabállyal kettéosztódhat (az egyik példányban három b, a másik példányban két b és egy c lesz).

2. példa: Egy h címkéjű membránban egyetlen a objektum van. Az $[_h a \to b]^0_h$ és $[_h b]^0_h \to [_h]^+_h c$ szabályokkal csak 2 lépésben lehet kivinni a végül c-vé alakuló objektumot (az objektumok és membránok szabályokhoz rendelése a lépések elején történik).

3. példa: A h címkéjű membránokra 4 szabály vonatkozik: $[_h a \to b]^0_h$, $[_h b]^0_h \to [_h]^+_h c$, $a\,[_h]^0_h \to [_h b]^-_h$ és $[_h a]^0_h \to c$. Megfelelő mennyiségű és fajtájú objektum rendelkezésre állása esetén egy h címkéjű membránra az első (akár több példányban) a másik 3 közül legfeljebb az egyikkel alkalmazható együtt. A második 3 szabály közül semmelyik 2 nem alkalmazható egyszerre ugyanazon membránra ugyanazon lépésben.

Aktív membránrendszerek konfigurációi

Definíció

$(\mu, w_1, \ldots, w_r)$ a $\Pi = \langle V, T, H, \mu, \omega_1, \ldots, \omega_m, R \rangle$ aktív P-rendszer **konfigurációja**, ahol μ az aktuális membránstruktúra r membránnal, $w_1, \ldots, w_r \in O^*$ pedig rendre az egyes membránok által határolt régiókban levő objektumok multihalmazát reprezentáló sztring.

Definíció

$(\mu, w_1, \ldots, w_m)$ **kezdőkonfiguráció**, ha μ a kezdeti membránstruktúra és $w_i = \omega_i$ $(1 \leqslant i \leqslant m)$.

Definíció

Az **egylépéses konfigurációátmenet** relációt a fent ismertetettek alapján definiáljuk. Egy a kezdőkonfigurációból induló konfigurációátmenet sorozatot Π egy **számításnak** nevezzük.

Aktív membránrendszerek által generált nyelv

Definíció

Egy számítás **megállási konfigurációba** jutott, ha a számítás nem folytatható (nem alkalmazható további szabály).

Definíció

Π egy termináló számításának eredménye a megállásig környezetbe jutott szimbólumok száma. A Π által **generált nyelv** az így generált számok halmaza, melyet $N(\Pi)$-vel jelölünk.

Alternatívák:

- a környezetbe jutott objektumok típusonkénti vektora
- figyelembe vesszük a környezetbe jutás sorrendjét, így generálhatunk sztringeket is
- megkülönböztethetünk terminális objektumokat és csak ezeket számoljuk

Aktív membránrendszerek számítási ereje

NOP_m(aktív, (a), (b), (c)) jelöli a természetes számok halmazainak családját, amelyek $N(\Pi)$ alakúak és amelyeket egy legfeljebb m-edfokú ($m \geqslant 1$) aktív membránrendszer generál, a következő szabályokat használva: (a) objektumok evolúciós szabályai, (b) befelé irányuló kommunikációs szabályok és (c) kifelé irányuló kommunikációs szabályok.

Általában (a), (b), (c) helyett a megengedett szabálytípusok listája áll.

Az aktív membránrendszerek is Turing univerzálisak:

Tétel

NOP_3 (aktív, (a), (b), (c)) = NRE

SAT lineáris időben

Tétel

A SAT probléma a változók és a klózok számának függvényében lineáris időben megoldható aktív P-rendszerrel a következő típusú szabályokat használva: (a) objektumok evolúciós szabályai, (b) (befelé irányuló) kommunikációs szabályok, (c) (kifelé irányuló) kommunikációs szabályok és (e) elemi membránok osztódását leíró szabályok (csak 2 részre osztódás).

Bizonyítás: Tekintsük az $\varphi = C_1 \wedge \cdots \wedge C_m$ n ítéletváltozót tartalmazó KNF-et, ahol $C_i = L_{i,1} \vee \cdots \vee L_{i,p_i}$ és $L_{i,j} \in \{X_k, \neg X_k \mid 1 \leqslant k \leqslant n\}$ $(1 \leqslant i \leqslant m, 1 \leqslant j \leqslant p_i)$.

Lineáris időben készítünk egy $O(n+m)$ lépésszámú $\Pi = \langle V, \{1,2\}, [_1[_2]_2^0]_1^0, \varepsilon, a_1 \cdots a_n d_0, R\rangle$ aktív P-rendszert, ahol $V = \{a_i t_i, f_i \mid 1 \leqslant i \leqslant n\} \cup \{r_i \mid 0 \leqslant i \leqslant m\} \cup \{c_i \mid 1 \leqslant i \leqslant m+1\} \cup \{d_i \mid 0 \leqslant i \leqslant n\} \cup \{\mathrm{yes}\}$

SAT lineáris időben

R szabályai:

(1) $[_2 a_i]_2^0 \to [_2 t_i]_2^0 [_2 f_i]_2^0$, $1 \leqslant i \leqslant n$.

(2) $[_2 d_k \to d_{k+1}]_2^0$, $0 \leqslant k \leqslant n-2$.

(3) $[_2 d_{n-1} \to d_n c_1]_2^0$.

(4) $[_2 d_n]_2^0 \to [_2]_2^+ d_n$.

(5) $[_2 t_i \to r_{j_1} \cdots r_{j_{k(i)}}]_2^+$, ha az X_i literált épp a $C_{j_1}, \ldots C_{j_{k(i)}}$ klózok tartalmazzák $(1 \leqslant i \leqslant n)$.

(6) $[_2 f_i \to r_{j_1} \cdots r_{j_{k(i)}}]_2^+$, ha a $\neg X_i$ literált épp a $C_{j_1}, \ldots C_{j_{k(i)}}$ klózok tartalmazzák $(1 \leqslant i \leqslant n)$.

(7) $[_2 r_1]_2^+ \to [_2]_2^- r_1$.

(8) $[_2 c_i \to c_{i+1}]_2^-$, $(1 \leqslant i \leqslant m)$.

(9) $[_2 r_k \to r_{k-1}]_2^-$, $(1 \leqslant k \leqslant m)$.

(10) $r_1 [_2]_2^- \to [_2 r_0]_2^+$

(11) $[_2 c_{m+1}]_2^+ \to [_2]_2^+$ yes.

(12) $[_1$ yes $]_1^0 \to [_1]_1^0$ yes.

SAT lineáris időben

Belátjuk, hogy Π akkor és csak akkor küld legalább egy „yes”-t a környezetbe, ha φ kielégíthető.

d_i és c_i számlálók, az indexük révén történik a lépések számlálása.

(1)-(4): n lépésben létrejön 2^n membrán, minden változókiértékeléshez egy, d_i-nek az i indexe azt mutaja, hogy hány változó kapott értéket. Ha $i = n$, akkor egy c_1 is bekerül a membránokba.

Példa: ha $n = 2$, akkor a következőt kapjuk: $[_1[_2 t_1 t_2 d_2 c_1]_2 [_2 t_1 f_2 d_2 c_1]_2 [_2 f_1 t_2 d_2 c_1]_2 [_2 f_1 f_2 d_2 c_1]_2]_1$.

Ezek után d_n kikerül a 2-es membránokból és a polaritásuk + lesz.

(5)-(6) után minden változókiértékelés membránja pontosan azon klózok r_i jelét tartalmazza, melyeket igazra értékel. Minden i-re r_i annyi példányban lesz jelen ahány literált igazzá tesz C_i-ben.

φ akkor és csak akkor kielégíthető, ha valamelyik membránban az összes r_i-ből van legalább egy példány.

SAT lineáris időben

(7)-(10): ellenőrzi a membránokra, hogy minden r_i benne van-e:

- Azok a membránok, amelyekben nincsen benne minden egyes r_i-nek legalább egy példánya előbb-utóbb blokkolódni fog (+ lesz a polaritása, és nem fog r_1-et tartalmazni).
- A még szóba jöhető membránok mindegyikéből pontosan 1 darab r_1 kerül ki az 1-es membránba.
- Ekkor minden r_i indexe eggyel csökken (így az i. lépésben az eredeti r_i-k jelenlétét ellenőrizzük). Lenullázza a többlet r_1 -ek indexét. c_i indexe eggyel nő.
- Mivel épp annyi negatív polaritású membrán van, mint r_1 az 1-es membránban, ezért ezek visszakerülnek a 2-es membránokba r_0-ként, újra +-ra állítva a polaritást.

(11)-(12): Ha a c számláló $m+1$-hez ér valamelyik 2-es membránban, az azt jelent, hogy minden r_i benne volt a membránban, a membránhoz tartozó interpretáció kielégíti φ-t. Ekkor a rendszer kiküld egy "yes" üzenetet a környezetbe.

Összesen lineáris, $n+2m+4$ iteráció van. □

Hamilton út lineáris időben

Tétel

A Hamilton-út probléma a gráf csúcsai számának függvényében lineáris időben megoldható aktív P-rendszerrel a következő típusú szabályokat használva: **(a)** objektumok evolúciós szabályai, **(b)** (befelé irányuló) kommunikációs szabályok, **(c)** (kifelé irányuló) kommunikációs szabályok és **(e)** elemi membránok osztódását leíró szabályok (többfelé osztódás is).

Bizonyítás: Legyen $G=(N,E)$ irányított gráf, ahol $n \geqslant 2$, és $N=\{1,2,\dots n\}$.
Konstruálunk egy $\Pi=(V,H,\mu,\varepsilon,dd_0,R)$, P-rendszert aktív membránokkal, ahol

$$V=\{a_i,b_i \mid 1\leqslant i\leqslant n\}\cup\{r_i \mid 0\leqslant i\leqslant n+1\}\cup\{c_i \mid 0\leqslant i\leqslant n+1\}\cup \{d_i \mid 0\leqslant i\leqslant 2n\}\cup\{d,\text{yes}\},$$

$H=\{1,2\}$, $\mu=[_1\,[_2\,]_2^0\,]_1^0$, és R a következő szabályokat tartalmazza:

Hamilton út lineáris időben

(1) $[_2 d]_2^0 \to [_2 a_1]_2^0 \cdots [_2 a_n]_2^0, \quad (1\leqslant i\leqslant n).$
(2) $[_2 d_k \to d_{k+1}]_2^0, \quad (0\leqslant k\leqslant 2n-2).$
(3) $[_2 d_{2n-1} \to d_{2n} c_1]_2^0,$
(4) $[_2 d_{2n}]_2^0 \to [_2\,]_2^+ d_{2n}.$
(5) $[_2 a_i \to r_i b_i]_2^0, \quad (1\leqslant i\leqslant n).$
(6) $[_2 b_i]_2^0 \to [_2 a_{j_1}]_2^0 \cdots [_2 a_{j_k}]_2^0, \quad (1\leqslant i\leqslant n, 1\leqslant j_1,\dots,j_k\leqslant n),$
úgy, hogy éppen $(i,j_1),\dots,(i,j_k)$ az i-ből kiinduló E-beli élek.
(7) $[_2 r_1]_2^+ \to [_2\,]_2^- r_1\ .$
(8) $[_2 c_i \to c_{i+1}]_2^-, \quad (1\leqslant i\leqslant n).$
(9) $[_2 r_i \to r_{i-1}]_2^-, \quad (1\leqslant i\leqslant n).$
(10) $r_1\,[_2\,]_2^- \to [_2\, r_0\,]_2^+$
(11) $[_2 c_{n+1}]_2^+ \to [_2\,]_2^+$ yes.
(12) $[_1$ yes $]_1^0 \to [_1\,]_1^0$ yes.

Hamilton út lineáris időben

A d_i és c_i objektumok számlálók, a számítás végességét garantálják. Minden 2-es címkéjű objektumban a számítás során pontosan egy található belőlük és szinkronban számolnak.

Az (1)-es szabály létrehoz n darab 2-es membránt. Az a_i és b_i objektumok az i. csúcsot reprezentálják. Ha a_i b_i-re változik, akkor a következő ütemben i-ből kiinduló éleket keresünk.

Az alapötlet az, hogy (5)-(6) segítségével n hosszú sétákat állítunk elő minden lehetséges módon. Minden sétához tartozni fog egy saját 2-es membrán. Az r_i objektumok a séta korábbi csúcsai.

Hamilton út lineáris időben

Például ha $2 \to 3 \to 2 \to 5 \to 3$ egy séta és $n = 5$, akkor lesz olyan 2-es membrán, mely sorra a $dd_0, a_2d_1, r_2b_2d_2, r_2a_3d_3, r_2r_3b_3d_4, r_2r_3a_2d_5, r_2^2r_3b_2d_6, \ldots r_2^2r_3^2r_5b_3d_{10}c_1$ által reprezentált objektum multihalmazt tartalmazza. A következő lépésben lesz a membrán polaritása $+$. Ezzel leáll a séta hosszabbítása és jön az ellenőrző fázis.

Végül a (7)-(12) szabályok a SAT-nál látott módon ellenőrzik, hogy van-e olyan 2-es membrán, amelyik mindegyik r_i-t tartalmazza. Ha van, akkor egy „yes" üzenetet kap a környezet legfeljebb $4n + 3$ ütem után. □

Megjegyzés: Ha membránokat akárhány helyett csak 2 részre osztó szabályokat használhatunk, akkor a konstrukció módosítása $O(n^2)$ lépésben dönti el a Hamilton út problémát.

Számítási modellek

10. előadás

A DNS számítások

DNS-számítások alatt olyan számítási modelleket és módszereket értünk, amelyeket a DNS szálak struktúrája, tulajdonságai, a szálakon végezhető műveletek, valamint a szálak viselkedése motivál.

Tom Head (1987): a DNS szálak rekombináns viselkedésének formális nyelvi modellezése (elméleti megközelítés)

Leonard M. Adleman (1994): kísérlet a Hamilton-út probléma egy példányának megoldására DNS szálakat manipuláló kémcsőrendszerek segítségével (gyakorlati megközelítés)

A DNS szerkezete

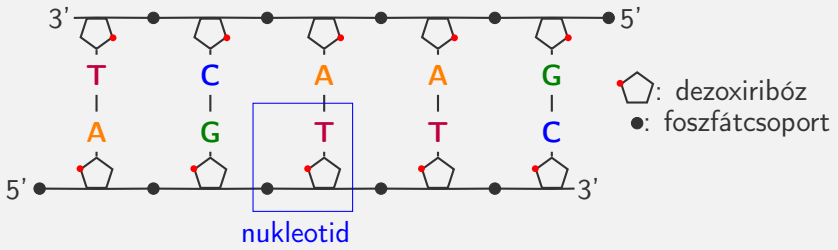
A DNS egy nukleinsav. A nukleinsavak ismétlődő nukleotid egységekből álló nagy méretű molekulák (polimerek). Minden nukleotid három egymáshoz kapcsolódó elemből áll:

- egy nitrogéntartalmú szerves bázisból (ez lehet adenin(A), timin(T), citozin(C), guanin(G))
- egy pentózcukorból (dezoxiribóz)
- egy foszfátcsoportból

A polimer váza a nukleotidok foszfodiészter kötéssel egymáshoz kapcsolódó dezoxiribóz részeiből áll. A foszfodiészter kötés az egyik nukleotid cukor komponensének 3'OH-csoportja és a következő cukorkomponensének 5'OH-ja között található egy foszfátcsoport közbeiktatásával.

A szerkezet változó része az egymást követő nukleotidok bázisainak a sorrendje, ez a bázissorrend határozza meg az információt.

A DNS szerkezete



A 4 nukleotidbázis közül kettő 9 atomos (adenin és guanin), ezek a **purinok**, kettő 6 atomos (timin és citozin), ezek a **pirimidinek**.

A DNS-ben két hosszú nukleotidszál kapcsolódik egymáshoz a bázispárok segítségével. A párok mindig A-T és G-C. A DNS a térben kettős spirál szerkezetű.

Az egyik szál egyik végén a cukor 5. szénatomjához végéhez még csatlakozik egy foszfátcsoport (röviden a nukleotidlánc 5' vége), a másik végén a cukor 3. szénatomjához egy OH csoport csatlakozik (a nukleotidlánc 3' vagy 3'OH vége). A komplemens szálon éppen fordítva vannak az 5' és 3' végek.

DNS rekombináció

A restrikciós enzimek (vagy restrikciós endonuklázok) képesek felismerni egy rövid nukleotidszekvenciát (részszót) a kétszálú DNS-en belül és a DNS-t adott helyeken elvágni.
Példa:

T C G A
A G C T

G C G C
C G C G

G C G C
C G C G

A létrejött fragmensek:

5'–C C C T C G A C C C–3'
3'–G G G A G C T G G G–5'

5'–C C C T
3'–G G G A G C

C G A C C C–3'
T G G G–5'

5'–A A A G C G C A A A–3'
3'–T T T C G C G T T T–5'

5'–A A A G
3'–T T T C G C

C G C A A A–3'
G T T T–5'

5'–T T T G C G C T T T–3'
3'–A A A C G C G A A A–5'

5'–T T T G C G
3'–A A A C

C T T T–3'
G C G A A A–5'

Leonard M. Adleman kísérlete

Az egyik első kísérlet biocomputer építésére. Ötlet: használjuk ki a DNS rekombináció jelenségében rejlő masszív párhuzamosságot egy nem hatékonyan kiszámítható probléma, például az NP-teljes Hamilton-út probléma egy példányának megoldására.

Adott néhány város (Adlemannál 7), közülük néhányat nonstop repülőjárat köt össze. Létezik-e olyan út Brisbane és Alice Springs között, amely mindegyik várost pontosan egyszer érinti?

Bemenet: G irányított gráf n csúccsal és két kitüntetett csúccsal, amelyeket v_{start}-tal és v_{end}-del jelölünk.
1. lépés: G -beli sétákat generálunk véletlenszerűen, nagy mennyiségben.
2. lépés: Csak azokat a sétákat tartjuk meg, amelyek v_{start}-tal kezdődnek és v_{end}-del végződnek.
3. lépés: Csak az n hosszú sétákat tartjuk meg.
4. lépés: Csak azokat a sétákat tartjuk meg, amik utak, azaz minden csúcsot érintenek.
Kimenet: YES, ha marad út, és NO egyébként.

DNS rekombináció

A különböző restrikciós (vágó) enzimek különféle alakú vágási végeket eredményeznek. Az egymáshoz illeszkedő vágási végek ún. ligáz (ragasztó) enzimek hatására a kémiai feltételek megléte esetén összeilleszthetők.

A egyik lehetőség persze, hogy az imént elvágott molekulát egy ligáz újra, ugyanúgy illeszti össze. Izgalmasabb azonban, ha az összeillő vágási végek összeragasztásával új, hibrid molekulák keletkeznek.

Az előző példában az első négy fragmens rekombinációja a következő új DNS-eket eredményezheti:

5'–C C C T C G C A A A–3'
3'–G G G A G C G T T T–5'

5'–A A A G C G A C C C–3'
3'–T T T C G C T G G G–5'

Ugyanakkor az utolsó 2 fragmens nem tud az első 4 fragmans egyikével se kombinálódni.

Adleman kísérlete

A program inputjának kódolásához a gráf mindegyik i csúcsához rendeljünk hozzá egy véletlen 20-mert (húsz nukleotid hosszúságú egyszeres DNS szálrészletet), amelyet s_i -vel, $(1 \leqslant i \leqslant n)$, jelölünk. s_1 legyen a v_{start}-hoz, s_n a v_{end}-hez tartozó 20-mer.

Pl. $i = 2, 3, 4$-re Adleman a következő 20 hosszúságú oligonukleotidokat (nukleotidlánc töredékeket) használta:

$s_2 = TATCGGATCGGTATATCCGA$,
$s_3 = GCTATTCGAGCTTAAAGCTA$,
$s_4 = GGCTAGGTACCAGCATGCTT$.

Ezen oligonukleotidok a 5' $\rightarrow$ 3' irányt követik.

Adleman kísérlete – Watson-Crick morfizmus

A h **Watson-Crick morfizmus** a 4 DNS bázist a Watson-Crick komplemensére képezi:

$h(A) = T, h(T) = A, h(C) = G, h(G) = C$.

h kiterjeszthető egy $\{C, T, A, G\}^* \to \{C, T, A, G\}^*$ homomorfizmussá. h-t betűnként alkalmazhatjuk a DNS szálra, pl.: $h(CATTAG) = GTAATC$.

Így h a DNS szálak Watson-Crick komplemensét adja vissza (h megváltoztatja az orientációt: az eredeti szál az 5' → 3' irányba mutat, míg a Watson-Crick komplemens a 3' → 5' irányba.)

Pl.

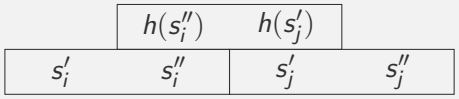
$s_2 = TATCGGATCGGTATATCCGA$,
$h(s_2) = ATAGCCTAGCCATATAGGCT$,

$s_3 = GCTATTCGAGCTTAAAGCTA$,
$h(s_3) = CGATAAGCTCGAATTTCGAT$.

Adleman kísérlete

Mindegyik s_i-t két 10-hosszú szálra bontjuk: $s_i = s_i's_i''$.
s_i'-t illetve s_i''-t úgy tekintjük, mint s_i első illetve második felét.

Vegyük észre, hogy a $e_{i\to j} := h(s_i''s_j')$ oligonukleotid illeszkedik az s_is_j 40-merhez.



Pl.
$e_{2\to 3} = CATATAGGCTCGATAAGCTC$,
$e_{3\to 2} = GAATTTCGATATAGCCTAGC$,
$e_{3\to 4} = GAATTTCGATCCGATCCATG$.

Vegyük észre, hogy ez a konstrukció megkülönbözteti az élek orientációját. $e_{2\to 3}$ és $e_{3\to 2}$ különböző 20-merek.

Adleman kísérlete

1. lépés: Minden i csúcsra és $i \to j$ élre a gráfban, s_i és $e_{i\to j}$ oligonukleotidokat keverünk össze nagy mennyiségben egy egyszerű ligációs reakcióban. Így a DNS molekulává kapcsolódás során ha $i \to j$ éle a gráfnak, akkor s_i és s_j következhet egymás után egy nukleotid láncban. Mivel az oligonukleotidok kellően hosszúak, ezért a ligációs reakció során kialakuló láncok megfelelnek a véletlen sétáknak.

2. lépés: Szálakra bontjuk az így kapott DNS darabokat. A polimeráz-láncreakció (PCR) nevű eljárás során kellő mennyiségben rendelkezésre álló építőkő (dezoxinukleotid-trifoszfát) és mesterséges kémiai feltételek mellett egy minta DNS szálat akár többmilló példányban lehet megsokszorozni. Ez a DNS szál két végéhez komplemensen illeszkedő ún. primer DNS szál darabok nagy számban való hozzáaadásával történik. Ha az s_1 és $h(s_n)$ primerekkel PCR-t alkalmazunk az 1. lépés eredményére akkor ezután a kémcső szinte csak v_{start}-ból v_{end}-be menő sétákat fog tartalmazni, igen nagy számban.

Adleman kísérlete

3. lépés: A gélelektroforézis alapelve, hogy a töltéssel rendelkező molekulák elektromos térben, össztöltésüknek megfelelően, az ellentétes töltésű elektróda felé vándorolnak. A rövidek gyorsabban, a hosszabbak lassabban. A gélelektroforézist a DNS szálak hossz alapján történő elválasztására használják. Csak a $20n$ nukleotidpárból álló kettős spirálokat hagyjuk meg, ezek n hosszú sétákat kódolnak.

4. lépés: Az ún. affinity purification (vonzalom alapú szűrés) olyan technikai eljárás, mely során kiszűrhetjük egy adott bázissorozatot tartalmazó nukleotidláncokat. A DNS negatív töltésű. Az x részszót tartalmazó szálak egy $h(x)$-et tartalmazó fémgolyóhoz tapadnak. Ezek után mágneses mező hatására az x-et nem tartalmazó szálak kiszűrhetők.

Ezt tegyük meg sorra $h(s_1), \ldots h(s_n)$-nel, rendre eldobva a nem illeszkedő láncokat. Akkor és csak akkor van Hamilton út v_{start}-ból v_{end}-be, ha az eljárás végén a kémcső legalább egy DNS-molekulát tartalmaz.

DNS számítások vs Neumann-elvű számítógép

- Adleman kísérlete 7 napig tartott egy 7 csúcsú gráfra, csekély volt a gyakorlati jelentősége.
- A DNS számításokkal potenciálisan nagyságrendekkel nagyobb párhuzamosság érhető el, mint Neumann típusú computerekkel.
- A molekulák kis mérete miatt potenciálisan nagyobb tárkapacitás érhető el.
- A kémiai reakciók lezajlását ki kell várni, így a DNS számítások jelenleg sokkal lassabbak, mint egy modern számítógép számításai. Ezt kéne a masszív párhuzamosságnak kompenzálnia.
- Egy Neumann típusú számítás eredménye sokkal könnyebben értelmezhető, mint egy kémiai kísérlet eredménye.
- Bár Adleman kísérlete óta történtek gyakorlati előrelépések, hatékony biocomputerek építése egyelőre gyerekcipőben jár.

A DNS-rekombináció matematikai modelljei

A DNS-rekombináció matematikai megfelelője a **splicing** (összeragasztás) művelete. Legyen V egy ábécé (például a DNS esetében $V = \{c, g, a, t\}$).

- Tom Head modellje (1987): Adott B és C (α, μ, β) alakú rendezett 3-asok halmaza, ahol $\alpha, \mu, \beta \in V^*$. Legyenek (α, μ, β) és (α', μ, β') vagy mindketten B-beli, vagy mindketten C-beli 3-asok továbbá $u\alpha\mu\beta v$ és $p\alpha'\mu\beta' q$ két V feletti szó. Ekkor a splicing művelet eredménye $u\alpha\mu\beta' q$.

  (Itt B és C a $3' \to 5'$ és $5' \to 3'$ orientációknak felel meg, míg a μ **kereszteződés** a ligációhoz szükséges illeszkedésének.)
- Gheorghe Păun modelljében (1996) a szabályok $u_1\#u_2\$u_3\#u_4$ alakúak és az $x_1u_1u_2x_2$ és $y_1u_3u_4y_2$ V feletti szavakra alkalmazva a művelet eredménye $x_1u_1u_4y_2$.
- Dennis Pixton modelljében (1996) a szabályok $(\alpha, \alpha'; \beta)$ alakúak és az $u\alpha v$ és $p\alpha' q$ V feletti szavakra $u\beta q$ a művelet eredménye.

A DNS-rekombináció matematikai modelljei

Megjegyzések:

- Păun modellje általánosabb Head modelljénél. Az (α, μ, β) és (α', μ, β') Head-féle szabálypárnak megfelel az $\alpha\mu\#\beta\$\alpha'\mu\#\beta'$ szabály.
- Pixton modellje általánosabb mindkét modellnél.

  Az (α, μ, β) és (α', μ, β') Head-féle szabálypárnak megfelel az $(\alpha\mu\beta, \alpha'\mu\beta'; \alpha\mu\beta')$ Pixton-szabály.

  A $u_1\#u_2\$u_3\#u_4$ Păun-féle szabálynak megfelel az $(u_1u_2, u_3u_4; u_1u_4)$ Pixton-szabály.
- Megmutatható, hogy van olyan reguláris nyelv, mely előállítható véges nyelvből Pixton-féle szabályokkal, de nem állítható elő Păun-féle szabályokkal, illetve olyan ami előállítható Păun-féle szabályokkal, de nem állítható elő Head-féle szabályokkal.
- Néha nem 1 hanem 2 szó a művelet eredménye. Például:
  $p\alpha'\mu\beta v$ is (Head modell)
  $y_1u_3u_2x_2$ is (Păun modell)

A splicing művelet

A továbbiakban ha külön nem említjük a Păun-féle változatra szorítkozunk.

Definíció

Egy (V ábécé feletti) $r = u_1\#u_2\$u_3\#u_4$ szabályt **splicing szabálynak** nevezünk, ha $u_i \in V^*$, $1 \leqslant i \leqslant 4$, valamint $\#$ és $\$$ speciális szimbólumok, amelyek bem V-beliek.

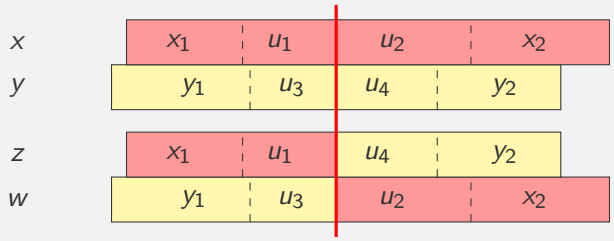
Definíció

Az $r = u_1\#u_2\$u_3\#u_4$ splicing szabályra és $x, y, z, w \in V^*$ sztringekre

- $(x, y) \vdash_r z$ akkor és csak akkor áll fenn, ha $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2$ és $z = x_1u_1u_4y_2$. valamely $x_1, x_2, y_1, y_2 \in V^*$ -ra (**1-splicing reláció**),
- $(x, y) \vdash_r (z, w)$ akkor és csak akkor áll fenn, ha $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2$, $z = x_1u_1u_4y_2$ és $w = y_1u_3u_2x_2$, valamely $x_1, x_2, y_1, y_2 \in V^*$-ra (**2-splicing reláció**).

A splicing művelet

Az $r = u_1\#u_2\$u_3\#u_4$ 2-splicing szabály illusztrálása az $x = x_1u_1u_2x_2$ és $y = y_1u_3u_4y_2$ szavakra:



- 1-splicing: a művelet eredménye egyetlen szó ($z = x_1u_1u_4y_2$), ilyenkor $(x, y) \vdash_r z$
- 2-splicing: a művelet eredménye két szó ($z = x_1u_1u_4y_2$ és $w = y_1u_3u_2x_2$)

Ha világos, hogy melyik szabályról van szó, akkor $\vdash_r$ helyett röviden $\vdash$ írható.

H-rendszer – nemiterált eset

Definíció

Splicing rendszernek (vagy **H-rendszernek**) nevezzük az $S = (V, A, R)$ rendezett 3-ast, ahol V egy ábécé, $A \subseteq V^*$ a kezdeti sztringek (axiómák) nyelve és R (V feletti) splicing szabályok halmaza.

Ekkor definiálhatjuk az egylépéses 1-splicing műveletet egy $L \subseteq V^*$ nyelvre, melynek eredménye az L-beli sztringeken minden lehetséges módon végrehajtott R szerinti 1-splicingok eredményeinek halmaza.

Definíció

Legyen $S = (V, A, R)$ egy H-rendszer és $L \subseteq V^*$. Ekkor az L nyelv **1-splicingja** $\sigma_1(L) = \{z \in V^* \mid (x, y) \vdash_r z,\ \text{ahol } x, y \in L, r \in R\}$.

(Az 1 alsó index az 1-splicingra utal.)

Megjegyzés: Ha $\sigma_1(x, y) = \{z \in V^* \mid (x, y) \vdash_r z,\ \text{ahol } r \in R\}$, akkor $\sigma_1(L) = \bigcup_{x,y \in L} \sigma_1(x, y)$.

H-rendszer számítási ereje – nemiterált eset

Ha az $S = (V, A, R)$ H-rendszerre és FL1,FL2 nyelvosztályokra $A \in$ FL1 és $R \in$ FL2, akkor S-et **(FL1,FL2) típusúnak** mondjuk.

Jelölés

Adott FL1 és FL2 nyelvcsaládokra használjuk a következő jelölést:

$S_1(\text{FL1, FL2}) := \{L \mid L = \sigma_1(A)$ valamely $S = (V, A, R)$ H-rendszerre ahol $A \in$ FL1 és $R \in$ FL2$\}$.

Jelölés:

FIN:	véges nyelvek nyelvcsaládja,
REG:	reguláris nyelvek nyelvcsaládja,
LIN:	lineáris nyelvek nyelvcsaládja (olyan CF grammatikával generálható, ahol a szabályok jobboldalán a terminálisokon kívül $\leqslant 1$ nemterminális áll).
CF:	környezetfüggetlen nyelvek nyelvcsaládja,
CS:	környezetfüggő nyelvek nyelvcsaládja,
RE:	rekurzíve felsorolható nyelvek nyelvcsaládja.

H-rendszer számítási ereje – nemiterált eset

Tételek S_1(FL1, FL2)-ről a következőket tudjuk:

FL1\FL2	FIN	REG	LIN	CF	CS	RE
FIN	FIN	FIN	FIN	FIN	FIN	FIN
REG	REG	REG	REG,LIN	REG,CF	REG,RE	REG,RE
LIN	LIN, CF	LIN,CF	RE	RE	RE	RE
CF	CF	CF	RE	RE	RE	RE
CS	RE	RE	RE	RE	RE	RE
RE	RE	RE	RE	RE	RE	RE

(Ha egy cellában két nyelvcsalád szerepel: szigorúan köztük van.)

H-rendszer – iterált eset

Definíció

Legyen $S = (V, A, R)$ egy H-rendszer és $L \subseteq V^*$. Ekkor az L nyelv $\sigma_1^*(L)$ **iterált 1-splicingját** a következőképpen definiáljuk:

$$\sigma_1^*(L) := \bigcup_{i \geqslant 0} \sigma_1^i(L),$$

ahol

$$\sigma_1^0(L) := L, \quad \sigma_1^{i+1}(L) := \sigma_1(\sigma_1^i(L)), \ i \geqslant 0.$$

Tehát $\sigma_1^*(L)$ a legkisebb nyelv, amely tartalmazza L-et és zárt az 1-splicing műveletére nézve.

Definíció

Az $S = (V, A, R)$ H-rendszer által **generált 1-splicing nyelv** alatt $L_1(S) := \sigma_1^*(A)$-t értjük.

H-rendszer által generált nyelv – példák

1. Legyen $S = (\{a\}, \{a\}, \{a\#\$\#a\})$ Könnyen látható, hogy $L_1(S) = a^+$.
2. Legyen $S = (V, A, R)$, ahol $\{a, b, c\}$, $A = \{cabb, cabab, cabaab\}$ és $R = \{caba\#a\$cab\#a\}$. Mivel cab mindig csak a szavak elejére illeszkedhet ezért $L_1(S) = caba^*b$ lesz.
3. Véges A axiómarendszerre $(aa)^* \neq \sigma_1^*(A)$.
   Indirekt, tegyük fel, hogy $L_1(S) = (aa)^*$, ahol $S = (V, A, R)$ és nyilván $V = \{a\}$, $A \subseteq V^*$ és $R \neq \varnothing$. Legyen $r \in R$, ekkor $r = a^k\#a^\ell\$a^m\#a^n$ valamely $k, \ell, m, n \in \mathbb{N}$-re. Tekintsük az $u = a^{k+\ell+x+1}$ és a $v = a^{m+n+y}$ szavakat $x, y \in \{0, 1\}$ értékeket úgy választva, hogy mindkét szó hossza (azaz $k + \ell + x + 1$ és $m + n + y$) páros legyen. Ekkor $u, v \in L_1(S)$, és az r szabály alkalmazható az (u, v) párra: (1) u-t a $k.$, v-t az $m + y.$ betű után vágjuk el, ekkor $(u, v) \vdash_r a^{k+n}$, (2) u-t a $k + 1.$, v-t az $m + y.$ betű után vágjuk el, ekkor $(u, v) \vdash_r a^{k+n+1}$ adódik. Egyik ptl. hosszú, ellentmondás.

H-rendszer számítási ereje – iterált eset

Jelölés

Adott FL1 és FL2 nyelvcsaládokra használjuk a következő jelölést:
$\mathrm{H}_1(\mathrm{FL1}, \mathrm{FL2}) := \{L \mid L = \sigma_1^*(A)$ valamely $S = (V, A, R)$ H-rendszerre ahol $A \in$ FL1 és $R \in$ FL2$\}$.

Tételek $\mathrm{H}_1(\mathrm{FL1}, \mathrm{FL2})$-ről a következőket tudjuk:

FL1\FL2	FIN	REG	LIN	CF	CS	RE
FIN	FIN,REG	FIN,RE	FIN,RE	FIN,RE	FIN,RE	FIN,RE
REG	REG	REG,RE	REG,RE	REG,RE	REG,RE	REG,RE
LIN	LIN, CF	LIN,RE	LIN,RE	LIN,RE	LIN,RE	LIN,RE
CF	CF	CF,RE	CF,RE	CF,RE	CF, RE	CF,RE
CS	CS,RE	CS,RE	CS,RE	CS,RE	CS,RE	CS,RE
RE	RE	RE	RE	RE	RE	RE

(Ha egy cellában két nyelvcsalád szerepel: szigorúan köztük van.)

2-splicing

Definíció

Legyen $S = (V, A, R)$ egy H-rendszer és $L \subseteq V^*$. Ekkor az L **2-splicingja** $\sigma(L) = \{w \in V^* \mid (x, y) \vdash_r (w, z)$ vagy $(x, y) \vdash_r (z, w)$, ahol $x, y \in L, r \in R\}$.

Definíció

Legyen $S = (V, A, R)$ egy H-rendszer és $L \subseteq V^*$. Ekkor az L nyelv $\sigma^*(L)$ **iterált 2-splicingját** a következőképpen definiáljuk:

$$\sigma^*(L) := \bigcup_{i \geqslant 0} \sigma^i(L),$$

ahol

$$\sigma^0(L) := L, \quad \sigma^{i+1}(L) := \sigma(\sigma^i(L)), \ i \geqslant 0.$$

Definíció

Az $S = (V, A, R)$ H-rendszer által **generált 2-splicing nyelv** alatt $L(S) := \sigma^*(A)$-t értjük.

2-splicing

Jelölés

Adott FL1 és FL2 nyelvcsaládokra vezessük be az alábbi jelöléseket:

S(FL1, FL2) := $\{L \mid L = \sigma(A)$ valamely $S = (V, A, R)$ H-rendszerre ahol $A \in$ FL1 és $R \in$ FL2$\}$.

H(FL1, FL2) := $\{L \mid L = \sigma^*(A)$ valamely $S = (V, A, R)$ H-rendszerre ahol $A \in$ FL1 és $R \in$ FL2$\}$.

Megjegyzés: Az 1-splicing erősebb, mint a 2-splicing. Ha egy (V, A, R) 1-splicing rendszer minden $u_1\#u_2\$u_3\#u_4 \in R$ szabálya esetén az $u_3\#u_4\$u_1\#u_2 \in R$ teljesül, akkor ez valójában ekvivalens egy 2-splicing rendszerrel.

Állítás: Van olyan nyelv, ami 1-splicing rendszerrel generálható, de 2-splicing rendszerrel nem.

Univerzalitási alaplemma

Tétel

Minden $L \in$ RE, $L \subseteq T^*$ nyelv felírható a következő alakban: $L = L_0 \cap T^*$, valamely $L_0 \in \mathrm{H}_1$(FIN, REG)-re.

Bizonyítás Tekintsük a $G = (N, T, P, S)$ 0-típusú grammatikát és alkalmazzuk az $U = N \cup T \cup \{B\}$ jelölést, ahol B új szimbólum. Megkonstruáljuk az $S = (V, A, R)$ H-rendszert, ahol $V = N \cup T \cup \{X, X', B, Y, Z\} \cup \{Y_\alpha \mid \alpha \in U\}$.

R a következő 1-splicing szabályokat tartalmazza.

Univerzalitási alaplemma

A grammatikai szabályok szimulálása:

1) $X\omega\#uY\$Z\#vY$, ahol $u \to v \in P, \omega \in U^*$,

A mondatforma elforgatása:

2) $X\omega\#\alpha Y\$Z\#Y_\alpha$, ahol $\alpha \in U, \omega \in U^*$,
3) $X'\alpha\#Z\$X\#\omega Y_\alpha$, ahol $\alpha \in U, \omega \in U^*$,
4) $X'\omega\#Y_\alpha\$Z\#Y$, ahol $\alpha \in U, \omega \in U^*$,
5) $X\#Z\$X'\#\omega Y$, ahol $\omega \in U^*$,

Terminálás:

6) $\#ZY\$XB\#\omega Y$, ahol $\omega \in T^*$,
7) $\#Y\$XZ\#$.

Legyen végül az axiómák halmaza

$A = \{XBSY, ZY, XZ\} \cup \{ZvY \mid u \to v \in P\} \cup \{ZY_\alpha, X'\alpha Z \mid \alpha \in U\}$.

Azt fogjuk belátni, hogy $L = \sigma_1^*(A) \cap T^*$.

Univerzalitási alaplemma

Az állítás bebizonyításához megvizsgáljuk, hogy S hogyan működik, vi. azt, hogy hogyan kaphatunk T^*-beli sztringeket.

Egyik A -beli sztring sem T^*-beli. Viszont minden R-beli szabály tartalmaz Z-t. Az 1-splicing szabályaink viszont olyanok, hogy nem keletkezik Z-t tartalmazó új sztring. Ezért minden lépésben egy $XBSY$-ból származtatott és $A - \{XBSY\}$-beli axiómát kell használnunk. (Két $A - \{XBSY\}$-beli se kombinálódhat.)

Állítás: minden új sztring $\beta w_1 B w_2 Y$ vagy $\beta w_1 B w_2 Y_\alpha$ alakú, ahol $\beta \in \{X, X'\}$ és $w_2 w_1$ illetve $w_2 w_1 \alpha$ G aktuális mondatformája és ezek a sztringek elő is állíthatók.

A 2)-5) típusú szabályokkal olyan új sztringeket kaphatunk, mely egy korábbi, X és Y által határolt γ szót ciklikusan elforgat. Ha γ-t addig forgatjuk, hogy egy G-beli szabály baloldala kerüljön a szó végére (Y elé), akkor az 1) típusú szabályokkal a mondatformából újabb (esetleg ciklikusan elforgatott) mondatformát kaphatunk.

Univerzalitási alaplemma

Kicsit részletesebben:

- A 2-es típusú szabály előállít olyan sztringet egy korábbiból, hogy az utolsó betűt (α-t) törli, a törölt betűt azonban megjegyzi Y indexében
- az előző sztringet csak 3-es típusú szabállyal lehet folytatni, ez az α-t beírja X elé (és X-ből X' lesz).
- az előző sztringet csak 4-es típusú szabállyal lehet folytatni, ez most törli az Y indexében feleslegessé vált információt
- az előző sztringet csak 5-ös típusú szabállyal lehet folytatni, ez X'-ből újra X-et csinál.

Összeségében egy betű a szó végéről (a közvetlenül Y előtti pozícióból) a szó elejére kerül át (közvetlenül X mögé).

Vegyük észre, hogy az aktuális ciklikus mondatformához mindig tartozik pontosan egy B marker, amelyik a ciklikus szóban a tényleges első betűt jelöli, így nem veszik el az az információ, hogy valójában hol kezdődik a mondatforma.

Univerzalitási alaplemma

Az egyetlen mód, hogy a nem T-beli szimbólumokat eltávolítsuk a σ alkalmazásával kapott sztringekből az az, hogy a 6) és 7) csoport szabályait használjuk. Pontosabban az XB szimbólumokat csak akkor tudjuk eltávolítani, ha a következő feltételek teljesülnek:

- Y jelen van (ha először a 7. csoportba tartozó szabályt használnánk, akkor Y eltávolítása miatt további splicing műveletben a sztring nem vehetne részt, de nem is lenne T^*-beli),
- B az X után, baloldalon van (a szó vissza van forgatva), és
- az XB és Y által közrefogott aktuális sztring csak T-beli szimbólumokból áll (azaz nincsenek N-beli jelek).

X és B eltávolítása után, Y-t szintén eltávolíthatjuk. A fenti meggondolások alapján nyilván az így kapott T^*-beli sztringek $L(G)$-beliek (azaz $\sigma_1^*(A) \cap T^* \subseteq L(G)$) és fordítva, minden $L(G)$-beli sztring előállítható ilyen módon (azaz $L(G) \subseteq \sigma_1^*(A) \cap T^*$). Ezekből $\sigma_1^*(A) \cap T^* = L(G)$ adódik, amit bizonyítani kellett. □

Kiterjesztett H rendszer

El tudnánk-e érni más módon a Turing-unverzalitást (bármely RE-beli nyelv generálhatóságát) véges axiómahalmaz és véges szabályrendszer esetén?

Definíció

Kiterjesztett H-rendszer alatt a $S = (V, T, A, R)$ négyest értjük, ahol

- V egy ábécé,
- $T \subseteq V$ a terminális ábécé,
- $A \subseteq V^*$ a kezdeti sztringek (axiómák) halmaza, és
- $R \subseteq V^*\#V^*\$V^*\#V^*$ splicing szabályok halmaza, ahol $\$, \# \notin V$.

Definíció

Az $S = (V, T, A, R)$ **kiterjesztett H-rendszer által generált 1-splicing nyelv** $L_1^T(S) := \sigma_1^*(A) \cap T^*$.

Kiterjesztett H-rendszerek számítási ereje

Jelölés

FL1 és FL2 nyelvcsaládokra, $\mathrm{EH}_1(\mathrm{FL1}, \mathrm{FL2})$ jelöli azon $L_1^T(S)$ alakú nyelvek családját, amelyeket az $S = (V, T, A, R)$ kiterjesztett H rendszer generál, ahol $A \in$ FL1 és $R \in$ FL2.

Tételek: $\mathrm{EH}_1(\mathrm{FL1}, \mathrm{FL2})$-ről a következőket tudjuk:

FL1\FL2	FIN	REG	LIN	CF	CS	RE
FIN	REG	RE	RE	RE	RE	RE
REG	REG	RE	RE	RE	RE	RE
LIN	LIN, CF	RE	RE	RE	RE	RE
CF	CF	RE	RE	RE	RE	RE
CS	RE	RE	RE	RE	RE	RE
RE	RE	RE	RE	RE	RE	RE

(Ha egy cellában két nyelvcsalád szerepel: szigorúan köztük van.)
EH(FL1, FL2) jelentése hasonló, csak 2-splicingra.

Kémcsőrendszerek

1996-ban Csuhaj Varjú E., L. Kari és Gh. Păun vezették be a kémcsőrendszerek (test tube systems) számítási modelljét.

Definíció

n-edfokú ($n \geqslant 1$) **kémcsőrendszer** alatt egy $\Gamma = (V, (V_1, A_1, R_1), (V_2, A_2, R_2), \ldots, (V_n, A_n, R_n))$ konstrukciót értünk, ahol

- V egy ábécé,
- $V_i \subseteq V$ $(1 \leqslant i \leqslant n)$,
- $A_i \subseteq V^*$ $(1 \leqslant i \leqslant n)$,
- $R_i \subseteq V^* \# V^* \$ V^* \# V^*$ $(1 \leqslant i \leqslant n)$.

A rendszer (V_i, A_i, R_i) komponensének neve **kémcső**.

V_i-t az i-edik kémcső **szelektorának** hívjuk.

Jelölés: $B = V^* \setminus \bigcup_{i=1}^{n} V_i^*$.

Észrevétel: Minden $1 \leqslant i \leqslant n$-re $S_i = (V, A_i, R_i)$ egy H-rendszer.

Kémcsőrendszerek– konfiguráció

Definíció

Az $(L_1, \ldots, L_n)$, $L_i \subseteq V^*$, $1 \leqslant i \leqslant n$, rendezett n-est a rendszer **konfigurációjának**, L_i-t pedig az i-edik kémcső tartalmának hívjuk.

Jelölés:
Legyen $\Gamma = (V, (V_1, A_1, R_1), (V_2, A_2, R_2), \ldots, (V_n, A_n, R_n))$ egy kémcsőrendszer. Minden $1 \leqslant i \leqslant n$-re jelölje $\sigma_i(L)$ az $L \subseteq V^*$ nyelv $S_i = (V, A_i, R_i)$ H-rendszer szerinti 2-splicing nyelvét, azaz

$\sigma_i(L) := \{w \in V^* \mid (x, y) \vdash_r (w, z)$ vagy $(x, y) \vdash_r (z, w)$, ahol $x, y \in L, r \in R_i\}$.

$\sigma_i^*(L)$ jelöli az iteratív 2-splicing nyelvet.

Kémcsőrendszerek– tranzíció

Definíció

Az $(L_1, \ldots, L_n)$ és $(L_1', \ldots, L_n')$ konfigurációkra definiáljuk a **tranzíciót** a következőképpen:
$(L_1, \ldots, L_n) \Longrightarrow (L_1', \ldots, L_n')$ akkor és csak akkor, ha minden $1 \leqslant i \leqslant n$-re

$$L_i' = \bigcup_{j=1}^{n} (\sigma_j^*(L_j) \cap V_i^*) \cup (\sigma_i^*(L_i) \cap B)$$

Minden kémcső tartalmát a saját splicing szabályhalmaza szerint kombináljuk egymással, amíg lehetséges, (L_i-ről $\sigma_i^*(L_i)$-re változtatjuk a tartalmat, $1 \leqslant i \leqslant n$), az eredményt pedig szétosztjuk az n kémcső között a $V_1, \ldots, V_n$ szelektoroknak megfelelően (egy sztringet több kémcső is megkaphat!)

Azok a részek, amelyeket nem tudunk elosztani (nem tartoznak egyetlen V_i^*-hoz sem, $1 \leqslant i \leqslant n$) a kémcsőben maradnak.

Kémcsőrendszerek – számítás

Definíció

Egy (k hosszú, $k \geqslant 0$) **számításon** Γ-ban konfigurációk egy $C = \{(L_1^{(t)}, \ldots, L_n^{(t)}) \mid 0 \leqslant t \leqslant k\}$ sorozatát értjük, úgy hogy

- $(L_1^{(0)}, \ldots, L_n^{(0)}) = (A_1, \ldots, A_n)$ és
- $(L_1^{(t)}, \ldots, L_n^{(t)}) \Longrightarrow (L_1^{(t+1)}, \ldots, L_n^{(t+1)})$ minden $0 \leqslant t \leqslant k-1$-re.

Ha van olyan C számítás, amellyel $(L_1, \ldots, L_n)$-ből $(L_1', \ldots, L_n')$-be juthatunk, akkor enne jelölése $(L_1, \ldots, L_n) \Longrightarrow^* (L_1', \ldots, L_n')$.

Definíció

Γ által **generált nyelv**:
$L(\Gamma) = \{w \in V^* \mid$ létezik $(A_1, \ldots, A_n) \Longrightarrow^* (L_1^{(t)}, \ldots, L_n^{(t)}), t \geqslant 0$ számítás, amelyre $w \in L_1^{(t)}\}$.

Tehát az 1-es számú, „főkémcsőben” gyűlik a számítás eredménye.

Kémcsőrendszerek – példa

Példa: Tekintsük a $\Gamma = (\{a, b, c, d, e\}, (\{a, b, c\}, \{cabc, ebd, dae\}, \{b\#c\$e\#bd, da\#e\$c\#a\}), (\{a, b, d\}, \{ec, ce\}, \{b\#d\$e\#c, c\#e\$d\#a\}))$ kémcsőrendszert.

Állítás: $L(\Gamma) \cap ca^+b^+c = \{ca^n b^n c \mid n \geqslant 1\}$ (Így $L(\Gamma) \notin$ REG.)

Teljes indukcióval látható, hogy minden iterációs lépésben az 1-es kémcsőben mindig 2 lépésre van lehetőség (a sorrendjük kötetlen):

- $(\alpha a^i b^j | c, e | bd) \vdash (\alpha a^i b^{j+1} d, ec) \quad \alpha \in \{c, d\}$,
- $(da | e, c | a^i b^j \alpha) \vdash (da^{i+1} b^j \alpha, ce) \quad \alpha \in \{c, d\}$.

Ezek után nem lehet több splicing műveletet végezni, a kapott $da^{i+1}b^{j+1}d$ átkerül a 2-es kémcsőbe. Itt is 2 lehetőség van (a sorrend ismét kötetlen):

- $(\alpha a^i b^j | d, e | c) \vdash (\alpha a^i b^j c, ed) \quad \alpha \in \{c, d\}$,
- $(c | e, d | a^i b^j \alpha) \vdash (ca^i b^j \alpha, de) \quad \alpha \in \{c, d\}$.

Ezek után nem lehet több splicing műveletet végezni, a kapott $ca^i b^j c$ átkerül a 2-es kémcsőbe. Ebből az állítás következik.

Kémcsőrendszerek számítási ereje

$\mathrm{TT}_n(\mathrm{FL1}, \mathrm{FL2})$ jelöli azon $L(\Gamma)$ nyelvek családját, melyekre $\Gamma = (V, (V_1, A_1, R_1), (V_2, A_2, R_2), \ldots, (V_m, A_m, R_m))$ valamely $m \leqslant n$-re és $A_i \in \mathrm{FL1}$, $R_i \in \mathrm{FL2}$ minden $1 \leqslant i \leqslant m$ esetén.

Γ-t ilyenkor (FL1, FL2) típusúnak mondjuk.

$\mathrm{TT}_*(\mathrm{FL1}, \mathrm{FL2}) = \bigcup_{n \geqslant 1} \mathrm{TT}_n(\mathrm{FL1}, \mathrm{FL2})$

Érdekesebb tételek (Csuhaj Varjú, Kari, Păun, 1996):

Tétel

- $\mathrm{TT}_2(\mathrm{FIN}, \mathrm{FIN}) \setminus \mathrm{REG} \neq \emptyset$, $\mathrm{TT}_4(\mathrm{FIN}, \mathrm{FIN}) \setminus \mathrm{CF} \neq \emptyset$, $\mathrm{TT}_6(\mathrm{FIN}, \mathrm{FIN}) \setminus \mathrm{CS} \neq \emptyset$, $\mathrm{TT}_6(\mathrm{FIN}, \mathrm{FIN}) \setminus \mathrm{R} \neq \emptyset$,
- $\mathrm{TT}_1(\mathrm{FIN}, \mathrm{FIN}) \subset \mathrm{REG} \subset \mathrm{TT}_2(\mathrm{FIN}, \mathrm{FIN}) \subseteq \mathrm{TT}_3(\mathrm{FIN}, \mathrm{FIN}) \subseteq \ldots \subseteq \mathrm{TT}_*(\mathrm{FIN}, \mathrm{FIN}) = \mathrm{TT}_*(\mathrm{FL1}, \mathrm{FL2}) = \mathrm{RE}$, ahol $\mathrm{FIN} \subseteq \mathrm{FL1}$, $\mathrm{FL2} \subseteq \mathrm{RE}$ tetszőlegesek,
- $\mathrm{H}(\mathrm{FL1}, \mathrm{FL2}) = \mathrm{TT}_1(\mathrm{FL1}, \mathrm{FL2})$ minden FL1-re és FL2-re,
- $\mathrm{EH}(\mathrm{FL1}, \mathrm{FL2}) \subseteq \mathrm{TT}_2(\mathrm{FL1}, \mathrm{FL2})$ minden FL1-re és FL2-re.

Számítási modellek

11. előadás

Fejlődő rendszerek

Aristid Lindenmayer (1925-1989), magyar származású holland biológus, matematikus a vörös algák (callithamnion roseum) fejlõdésének tanulmányozása során írta le a később róla elnevezett párhuzamos fejlődő rendszert.

Az algák sejtjeinek 9 állapota van. Csak ez határozza meg a következő állapotát. Bizonyos állapotokban lévő sejtek növelhetik az algát, sőt oldalágat is növeszthetnek.

Matematikailag így írható le, kezdetben egyetlen 1-es állapotú sejt van.

1→23	4→25	7→8	[→[
2→2	5→65	8→9[3]	]→]
3→24	6→7	9→9	

([és] oldalágat határol.)

Vörös algák

1→23,2→2,3→24,4→25, 5→65,6→7,7→8,8→9[3],9→9,[→[,]→]

Az 1 axiómából induló levezetés elemei:

1, 23, 224, 2225, 22265, 222765, 2228765, 2229[3]8765

2229[24]9[3]8765, 2229[225]9[24]9[3]8765

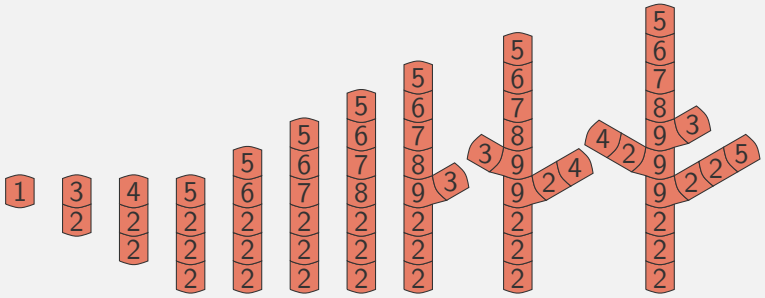
2229[2265]9[225]9[24]9[3]8765,

2229[22765]9[2265]9[225]9[24]9[3]8765,

2229[228765]9[22765]9[2265]9[225]9[24]9[3]8765

2229[229[3]8765]9[228765]9[22765]9[2265]9[225]9[24]9[3]8765

2229[229[24]9[3]8765]9[229[3]8765]9[228765]9[22765]9[2265]9[225]9[24]9[3]8765



0L-rendszer

Definíció

0L-rendszer (Lindenmayer-rendszer, 0L-grammatika, L-rendszer) alatt egy $G = (V, P, \omega)$ rendezett hármast értünk, ahol

- V egy véges ábécé,
- P környezetfüggetlen V feletti átírási szabályok véges halmaza, feltesszük, hogy minden $a \in V$-re létezik szabály P-ben,
- $\omega \in V^+$ pedig az axióma.

Definíció

$z_1, z_2 \in V^*$ szavak esetében $z_1 \Longrightarrow_G z_2$ írható ha
$z_1 = a_1 a_2 \cdots a_r$ $(a_i \in V, 1 \leqslant i \leqslant r)$, $z_2 = x_1 x_2 \cdots x_r$
$(x_i \in V^*, 1 \leqslant i \leqslant r)$ és minden $1 \leqslant i \leqslant r$ -re $a_i \to x_i \in P$.

(G elhagyható, ha világos, hogy melyik G-ről van szó.)

A $\Longrightarrow^*$ reláció a $\Longrightarrow$ reláció reflexív, tranzitív lezártja.

0L-rendszer által generált nyelv

Definíció

A $G = (V, P, \omega)$ **0L-rendszer által generált nyelv** $L(G) = \{u \in V^* \mid \omega \Longrightarrow^* u\}$.

Példa: Legyen $G = (V, P, \omega)$ egy 0L-rendszer, ahol $V = \{a\}, P = \{a \to a^2\}$, és $\omega = a^3$.

Ekkor $L(G) = \{a^{3 \cdot 2^i} \mid i \geqslant 0\}$.

Definíció

Amennyiben a $G = (V, P, \omega)$ 0L-rendszerben minden $a \in V$-re pontosan egy P-beli szabálya van G-nek, akkor **D0L-rendszerről** beszélünk (determinisztikus 0L-rendszer).

Ilyenkor P valójában egy $h : V \to V^*$ homomorfizmus, a $\langle h^t(\omega),\ t \geqslant 0\rangle$ sorozat $(h^0(\omega) := \omega, h^t(\omega) = h(h^{t-1}(\omega)), t \geqslant 0)$ a G D0L rendszer növekedési sorozata, $f(t) := |h^t(\omega)|, (f : \mathbb{N} \to \mathbb{N})$, a növekedési függvénye.

0L-rendszer – Példák

Példa: $G_{\text{FIB}} = (\{a, b\}, \{a \to b, b \to ab\}, a)$.

G_{FIB} D0L-rendszer, azaz valójában egy h homomorfizmus. Jelölje ω_n az n. átírás után kapott szót. $\omega_0 = a, \omega_1 = b, \omega_2 = ab = \omega_0\omega_1$. Teljes indukcióval belátjuk, hogy $n \geqslant 2$-re $\omega_n = \omega_{n-2}\omega_{n-1}$:

$\omega_{n+1} = h(\omega_n) = h(\omega_{n-2}\omega_{n-1}) = h(\omega_{n-2})h(\omega_{n-1}) = \omega_{n-1}\omega_n$.
Tehát az $f(n)$ növekedési sorozatra $f(0) = 1, f(1) = 1$,

$f(n) = f(n-1) + f(n-2)$ rekurzió teljesül, ami megegyezik a Fibonacci sorozatra vonatkozó rekurzióval, eggyel eltolva. Tehát $f(n) = F_{n+1}$, ahol F_n jelöli az n. Fibonacci számot.

Teljes indukcióval hasonlóan látható, hogy $|\omega_n|_b = F_n$ $(n \geqslant 0)$ és $|\omega_n|_a = F_{n-1} (n \geqslant 1)$.

Példa: Nem létezik olyan G 0L rendszer, melyre $L(G) = \{a, a^2\}$. Ugyanis, ha $\omega = a$, akkor $a \Rightarrow^* a^2$, és így $a^2 \Rightarrow^* a^4$, tehát $a^4 \in L(G)$. Míg ha $\omega = a^2$, akkor $a^2 \Rightarrow^* a$, de ekkor $a \Rightarrow^* \varepsilon$, mert az átírási szabályok környezetfüggetlenek (az egyik a-ból a-t a másikból ε-t vezettük le), tehát $\varepsilon \in L(G)$.

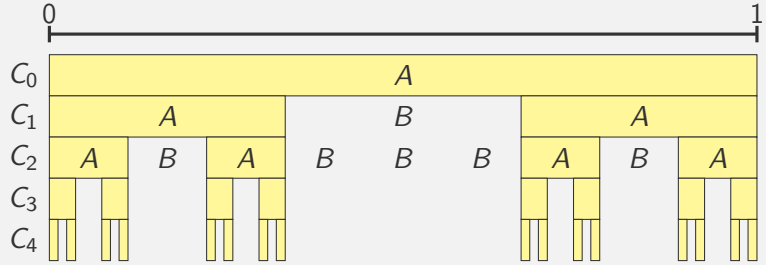
0L rendszer –Példák

Példa: $G_{\text{CANTOR}} = (\{A, B\}, \{A \to ABA, B \to BBB\}, A)$.

Legyen $\langle \omega_n, n \geqslant 0\rangle$ a G_{CANTOR} D0L-rendszer által generált szó-sorozat és $\omega_{n,i}$ $(1 \leqslant i \leqslant 3^n)$ az n. szó i. betűje. Ekkor

$$C_n := \bigcup_{i:\omega_{n,i}=A} \left[\frac{(i-1)}{3^n}, \frac{i}{3^n}\right]$$

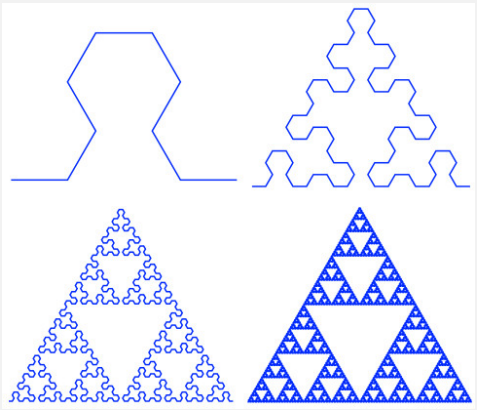
a jól ismert Cantor-halmaz közelítésének n-edik iterációja.



0L rendszer –Példák

$G_{\text{SIERPIŃSKI}} = (\{A, B, +, -\}, \{A \to B - A - B, B \to A + B + A, + \to +, - \to -\}, A)$.

A és B: teknőcgrafika rajzoljon ki egy egységnyi vonalat abban az irányban, amerre a teknős áll. + és - jelentése: forduljon balra illetve jobbra 60 fokkal.Az első páros iterációk ezeket adják:



Kiterjesztett 0L rendszer

Definíció

A $G = (V, P, \omega)$ 0L rendszert szaporodó (propagating) Lindenmayer-rendszernek (**P0L-rendszernek**) nevezzük, ha nincs ε-szabálya.

Definíció

Egy $G = (V, T, P, \omega)$ rendszert ($T \subseteq V$) kiterjesztett (extended) Lindenmayer-rendszernek (**E0L-rendszernek**) nevezünk, ha G V feletti 0L-rendszer. Egy G kiterjesztett Lindenmayer-rendszer által **generált nyelv** $L(G) = \{u \in T^* \mid \omega \Longrightarrow^* u\}$).

G kiterjesztett determinisztikus Lindenmayer-rendszer (**ED0L-rendszer**), ha V feletti D0L-rendszer.

T neve: terminális ábécé.

E0L rendszer – Példa

Példa: $G_{\text{SYNC}} = (\{A, B, C, A', B', C', F, a, b, c\}, \{a, b, c\}, P, ABC)$
P szabályai:

$$\begin{array}{lllll} A \to AA' & A \to a & A' \to A' & A' \to a & a \to F \\ B \to BB' & B \to b & B' \to B' & B' \to b & b \to F \\ C \to CC' & C \to c & C' \to C' & C' \to c & c \to F \end{array} \qquad F \to F$$

Az F nemterminálisnak szinkronizáló szerepe van. Ugyanis ha néhány átírási lépés után maradt még nemterminális a szóban, akkor kell még legalább egy átíró lépés. Ekkor viszont a már meglévő terminálisok megváltoztathatatlanul átíródnak F-re, így végül nem kapunk terminális szót.

Ha n átírás után a kapott szó nem tartalmaz terminálist akkor az csak $A(A')^n B(B')^n C(C')^n$ lehet. Tehát
$L(G_{\text{SYNC}}) = \{a^n b^n c^n \mid n \geqslant 1\}$.

Észrevétel: E0L-rendszerrel minden környezetfüggetlen nyelv generálható. Adjuk hozzá ugyanis az $a \to a$ E0L-szabályokat minden $a \in N \cup T$-re a nyelvet generáló grammatika szabályaihoz.

Táblázatos 0L-rendszer

Definíció

Egy $G = (V, P_1, \ldots, P_m, \omega)$ rendszert táblázatos Lindenmayer rendszernek (**T0L-rendszernek**) nevezünk, ha minden $1 \leqslant i \leqslant m$ esetén $G_i := (V, P_i, \omega)$ 0L rendszer.

Definíció

A $G = (V, P_1, \ldots, P_m, \omega)$ T0L rendszerben $z_1 \Longrightarrow_G z_2$ írható, ha létezik $1 \leqslant i \leqslant m$ melyre $z_1 \Longrightarrow_{G_i} z_2$.

(G elhagyható, ha világos, hogy melyik G-ről van szó.)

A $\Longrightarrow^*$ reláció a $\Longrightarrow$ reláció reflexív, tranzitív lezártja.

Definíció

A $G = (V, P_1, \ldots, P_m, \omega)$ **T0L-rendszer által generált nyelv** $L(G) = \{u \in V^* \mid \omega \Longrightarrow^* u\}$.

Kiterjesztett táblázatos 0L-rendszer

Definíció

Egy $G = (V, T, P_1, \ldots, P_m, \omega)$ rendszert kiterjesztett táblázatos Lindenmayer rendszernek (**ET0L-rendszernek**) nevezünk, ha minden $1 \leqslant i \leqslant m$ esetén $G_i := (V, T, P_i, \omega)$ E0L rendszer.

Definíció

A $G = (V, P_1, \ldots, P_m, \omega)$ T0L rendszerben $z_1 \Longrightarrow_G z_2$ írható, ha létezik $1 \leqslant i \leqslant m$ melyre $z_1 \Longrightarrow_{G_i} z_2$.

A $\Longrightarrow^*$ reláció a $\Longrightarrow$ reláció reflexív, tranzitív lezártja.

A $G = (V, P, \omega)$ **ET0L-rendszer által generált nyelv** $L(G) = \{u \in T^* \mid \omega \Longrightarrow^* u\}$.

Analóg módon definiáljuk a TD0L, ETD0L, EPT0L, stb. rendszereket.

Kiterjesztett táblázatos 0L-rendszer

Példa:

$G_{\text{SYNC2}} = (\{A, B, C, A', B', C', a, b, c\}, \{a, b, c\}, P_1, P_2, ABC)$

P_1 szabályai:		P_2 szabályai:	
$A \to AA'$	$A' \to A'$	$A \to a$	$A' \to a$
$B \to BB'$	$B' \to B'$	$B \to b$	$B' \to b$
$C \to CC'$	$C' \to C'$	$C \to c$	$C' \to c$

$L(G_{\text{SYNC2}}) = \{a^n b^n c^n \mid n \geqslant 1\}$.

Példa:

Legyen $G_{\text{DADOG}} = (\{S, A, B, a, b, c\}, \{a, b, c\}, P_1, P_2, SSS)$, ahol

P_1 szabályai:		P_2 szabályai:	
$S \to A$	$a \to a$	$S \to B$	$a \to a$
$A \to Sa$	$b \to b$	$B \to Sb$	$b \to b$
$B \to c$	$c \to c$	$A \to c$	$c \to c$

$L(G_{\text{DADOG}}) = \{cwcwcw \mid w \in \{a, b\}^*\}$.

ET0L-rendszerek

Definíció

Az E0L,D0L,ET0L, stb. rendszerek által generálható nyelvek nyelvosztályát rendre $\mathcal{L}(\text{E0L}), \mathcal{L}(\text{D0L}), \mathcal{L}(\text{ET0L})$, stb. jelöli.

A következőkben bizonyítás nélül ismertetjük az $\mathcal{L}(\text{ET0L})$ nyelvosztály néhány érdekes tulajdonságát.

D0L, DT0L rendszerek nem tudnak minden véges nyelvet generálni.

Tétel

Minden ET0L rendszert lehet 2 táblás ET0L rendszerrel szimulálni.

Tétel

Az $\mathcal{L}(\text{ET0L})$ és $\mathcal{L}(\text{EDT0L})$ zárt az unióra, konkatenációra, Kleene-lezártra, homomorfizmusra, reguláris nyelvvel való metszetre.

ET0L nyelvek egy tulajdonsága

Jelölés: Legyen $u = t_1 \dots t_n$, ahol $t_i \in X$ $(1 \leqslant i \leqslant n)$, továbbá $Y \subseteq X$, ekkor

$$|u|_Y := \left| \{ i \mid (1 \leqslant i \leqslant n) \wedge (t_i \in Y) \} \right|.$$

Az alábbi lemma (nem bizonyítjuk) egy szükséges (de nem elégséges !!!) feltételt ad egy nyelv ET0L rendszerrel való generálhatóságára.

Lemma

Legyen $L \subseteq T^*$ egy $\mathcal{L}(\text{ET0L})$-beli nyelv. Ekkor bármely $\varnothing \neq S \subseteq T$-re $\exists k > 0$ egész, hogy $\forall u \in L$-re

(i) vagy $|u|_S \leqslant 1$

(ii) vagy u-nak van olyan w részszava amelyre $|w| \leqslant k$ $|w|_S \geqslant 2$

(iii) vagy van L-nek végtelen sok olyan w szava, amelyre $|w|_S = |u|_S$.

0L-rendszerek számítási ereje

Példa: $\{(ab^n)^m \mid m \geqslant n \geqslant 1\} \notin \mathcal{L}(\text{ET0L})$.

Legyen $S = \{a\}$, $k > 0$ és $u = (ab^{k+1})^{k+1}$.

- Mivel $k + 1 \geqslant 2$ ezért (i) nem teljesül.
- Az a-k u-ban k-nál távolabb vannak egymástól, így (ii) sem.
- A $k + 1$ darab a-t tartalmazó L-beli szavak: $\{(ab)^{k+1}, (ab^2)^{k+1}, \dots, (ab^{k+1})^{k+1}\}$. Ezek száma véges, így (iii) sem.

Mivel $u \in L$, a Lemmából következik az állítás.

Megjegyzés: $\{(ab^n)^m \mid m \geqslant n \geqslant 1\} \in \text{CS} \setminus \text{CF}$

Tétel

$\text{CF} \subset \mathcal{L}(\text{E0L}) = \mathcal{L}(\text{EP0L}) \subset \mathcal{L}(\text{ET0L}) = \mathcal{L}(\text{EPT0L}) \subset \text{CS}$

Megjegyzés: Vannak 1L-rendszerek, 2L-rendszerek, stb.
1L-rendszer: egy bal- vagy jobboldalon jelenlévő szimbólumtól függ az átírás alkalmazhatósága (azaz az átírási szabályok környezetfüggőek).

Watson-Crick 0L-rendszerek

Alapötlet: Vizsgáljuk meg, hogy milyen nukleotidsorozatokat kaphatunk, ha a DNS evolúciójára fejlődő rendszerként tekintünk. DNS specifikus feltevés, hogy mindig rendelkezésre áll a generált sztring Watson-Crick komplemense. Így ha „rossz" sztringet kapunk áttérhetünk, a komplemens sztringe azzal folytatva a számítást.

Az áttérés lehet szabad vagy valamilyen kontroll hatására történő.

A Lindenmayer rendszerek minden betűre párhuzamosan átírják az aktuális sztringet a szabályaik szerint. A Watson-Crick komplemensre való áttérés is valójában egy Lindenmayer-típusú átírás, így természetes módon illeszkedik 0L rendszerekhez.

DNS-típusú ábécé, Watson-Crick morfizmus

Definíció

DNS-típusú ábécén egy $2n$ ($n \geqslant 1$) darab betűből álló, $\Sigma = \{a_1, \ldots, a_n, \bar{a}_1, \ldots, \bar{a}_n\}$ ábécét értünk. Az a_i és $\bar{a}_i$ ($1 \leqslant i \leqslant n$) betűket egymás komplemenseinek hívjuk, továbbá a nem felülhúzott betűket **purinnak**, a felülhúzottakat pedig **pirimidinnek** nevezzük.

Ez a terminológia az eredeti DNS ábécéből, $\{A, G, T, C\}$-ből ered, ahol az A és G betűk ténylegesen purinokat, a T és C betűk pedig pirimidineket jelölnek. T az A, C pedig a G komplemense.

Definíció

h_W-vel jelöljük azt a DNS-típusú Σ ábécé feletti betűnkénti endomorfizmust, amely minden betűhöz a komplemensét rendeli. h_W-t **Watson-Crick morfizmusnak** is nevezzük.

Példa

$\Sigma = \{a, g, t, c\}$, $h_W(a) = t, h_W(g) = c, h_W(t) = a, h_W(c) = g$.

Watson-Crick 0L-rendszerek

Definíció

Watson-Crick 0L rendszer (W0L rendszer) alatt egy olyan $W = (\Sigma, P, \omega, \varphi)$ rendezett 4-est értünk, ahol

- Σ DNS-típusú ábécé egy h_W Watson-Crick morfizmussal,
- $G_W = (\Sigma, P, \omega)$ egy 0L rendszer,
- $\varphi : \Sigma^* \to \{0, 1\}$ egy olyan leképezés, amelyre $\varphi(\omega) = \varphi(\epsilon) = 0$ és minden $u \in \Sigma^*$ szóra, amelyre $\varphi(u) = 1$ igaz, $\varphi(h_W(u)) = 0$ teljesül.

A komplementerre váltást előidéző feltétel (**trigger**) az, hogy $\varphi(u) = 1$, amit úgy értelmezünk, hogy a szó **„rossz"** lett. Amíg $\varphi(u) = 0$ teljesül, addig a szó **„jó"**. Feltesszük, hogy minden „rossz" szó WC-komplementere „jó", fordítva, nem feltétlen kell így legyen.

Ha az 0L rendszer D0L,E0L, stb, akkor Watson-Crick D0L, E0L stb. rendszerről beszélünk (röviden WD0L, WE0L stb. rendszerek).

Watson-Crick 0L-rendszerek – levezetés

Definíció

$z_1, z_2 \in V^*$ szavak esetében $z_1 \Longrightarrow_W z_2$ írható, ha

- $z_1 \Longrightarrow_{G_W} z_2$ és $\varphi(z_2) = 0$ vagy
- $z_1 \Longrightarrow_{G_W} h_W(z_2)$ és $\varphi(h_W(z_2)) = 1$.

Példa:

$\Sigma = \{a, g, t, c\}$, $h_W(a) = t, h_W(g) = c, h_W(t) = a, h_W(c) = g$.
$P = \{a \to \varepsilon, a \to ca, g \to cat, c \to ta, t \to \varepsilon\}$, $\omega = gac$.

A $\varphi(u) = 1 \Leftrightarrow catcat \subseteq u$ ($catcat$ részszava u-nak) trigger nem teljesíti W0L rendszer φ-re vonatkozó feltételét. Ilyenkor blokkolhat a rendszer, pl. a $catcatgtagta$ szóra nem lehetne tovább haladni. Ezért szükséges ez a feltétel.

Most viszont nem generálható blokkoló szó (g csak trigger után lehet a szóban, de ekkor nincs c). Maga az axióma blokkolhat.

Egy levezetés: $gac(\Rightarrow catcata) \overset{\text{trigger}}{\Longrightarrow} gtagtat(\Rightarrow catcatca) \overset{\text{trigger}}{\Longrightarrow}$
$gtagtagt(\Rightarrow (cat)\varepsilon\varepsilon(cat)\varepsilon(ca)(cat)\varepsilon) \overset{\text{trigger}}{\Longrightarrow} gtagtagtgta$.

Standard Watson-Crick 0L-rendszerek

Egy szó kielégíti a standard trigger komplemens átírásra vonatkozó feltételét (azaz a szó rossz), ha több pirimidint (felülhúzott betűt) tartalmaz, mint purint (nem felülhúzottat). Formálisan:

Definíció

$\Sigma = \{a_1, \ldots, a_n, \bar{a}_1, \ldots, \bar{a}_n\}$,
$\Sigma_{\mathrm{PUR}} = \{a_1, \ldots, a_n\}$, $\Sigma_{\mathrm{PYR}} = \{\bar{a}_1, \ldots, \bar{a}_n\}$.
Ekkor

$$\varphi_{\mathrm{STD}}(u) = \begin{cases} 0 & \text{ha } \sum_{a\in\Sigma_{\mathrm{PUR}}} |w|_a \geqslant \sum_{a\in\Sigma_{\mathrm{PYR}}} |w|_a \\ 1 & \text{ha } \sum_{a\in\Sigma_{\mathrm{PUR}}} |w|_a < \sum_{a\in\Sigma_{\mathrm{PYR}}} |w|_a \end{cases}$$

definiálja a standard komplementerre váltási (trigger) feltételt.

Standard Watson-Crick E0L-rendszer – Példa

Példa: $W = (V, T, P, \omega, \varphi_{\mathrm{STD}})$, ahol
$V = \{S, S_1, F, E, A, B, X, X_1, X_2, a, b, \bar{S}, \bar{S}_1, \bar{F}, \bar{E}, \bar{A}, \bar{B}, \bar{X}, \bar{X}_1, \bar{X}_2, \bar{a}, \bar{b}\}$,
$P = \{S \to SF\bar{E} \mid \bar{S}_1,\ \ F \to F,\ \ \bar{E} \to \bar{E},$
$S_1 \to \bar{X}_1 A,\ \ \bar{F} \to \bar{X}_1\bar{X}_1 X \mid \bar{X}_1 X,\ \ E \to A,$
$\bar{X}_1 \to \bar{X}_1,\ \ X \to XX_2,\ \ A \to A\bar{B},\ \ \bar{B} \to \bar{B},\ \ X_2 \to X_2,$
$X_1 \to \varepsilon,\ \ \bar{X} \to \varepsilon,\ \ \bar{X}_2 \to \varepsilon,\ \ \bar{A} \to a,\ \ B \to b\}$,
$T = \{a, b\}$, $\omega = S$.

$$S \overset{(m-1)}{\Longrightarrow} S(F\bar{E})^{m-1} (\Longrightarrow \bar{S}_1(F\bar{E})^{m-1}) \overset{\text{trigger}}{\Longrightarrow} S_1(\bar{F}E)^{m-1}$$
$$\Longrightarrow \bar{X}_1 A(\bar{X}_1 XA)^{n-1}(\bar{X}_1\bar{X}_1 XA)^{m-n} \quad (1 \leqslant n \leqslant m)$$

Lehet más a sorrendjük. Most $2m-1$ purin és $2m-n$ pirimidin van. Az $\bar{X}_1 \to \bar{X}_1, X \to XX_2, A \to A\bar{B}, \bar{B} \to \bar{B}, X_2 \to X_2$ szabályokkal $m-1$ új purinnal és m pirimidinnel nő a szó hossza. Így éppen n lépés után fordít át a standard trigger.

$$\overset{(n-1)}{\Longrightarrow} \alpha(\Longrightarrow \bar{X}_1 A\bar{B}^n(\bar{X}_1 XX_2^n A\bar{B}^n)^{n-1}(\bar{X}_1\bar{X}_1 XX_2^n A\bar{B}^n)^{m-n}) \overset{\text{trigger}}{\Longrightarrow}$$
$$X_1 A\bar{B}^n(X_1\bar{X}\bar{X}_2^n A\bar{B}^n)^{n-1}(X_1 X_1\bar{X}\bar{X}_2^n A\bar{B}^n)^{m-n} \Longrightarrow (ab^n)^m.$$

Tehát $L(W) = \{(ab^n)^m \mid m \geqslant n \geqslant 1\}$. (ET0L-lel nem generálható.)

Watson-Crick D0L rendszerek hálózatai

Watson-Crick D0L rendszerek hálózatai egy közös DNS típusú ábécé feletti véges sok páronként összeköttetésben lévő D0L rendszerből állnak. A rendszer determinisztikus és számításának egy üteme a következőkből áll:

1. a hálózat csúcsaiban elhelyezkedő determinisztikus Lindenmayer rendszerek az aktuális szóhalmazon szinkronizált módon átírást végeznek. A csúcsokban determinisztikus átírást alkalmazunk, azaz valójában minden csúcshoz egy homomorfizmus van hozzárendelve,
2. majd az így nyert szavakat a Watson-Crick komplementaritás elvét alapul vevő valamilyen kommunikációs protokollt használva egymásnak közvetítik.

A csúcsokból szavak sosem törlődnek, így a számítás eredménye egy kitüntetett csúcsban keletkező szavak összessége (azaz a $\leqslant n$ lépésű számítások limesze, $n \to \infty$).

Watson-Crick D0L rendszerek hálózatai

Definíció

$\Gamma = (\Sigma, P_1, \omega_1, \ldots, P_r, \omega_r, \varphi)$ **N$_r$WD0L rendszer** ($r \geqslant 1$ Watson-Crick D0L rendszer hálózata) ha a következők teljesülnek:

- $\Sigma = \{a_1, \ldots, a_n, \bar{a}_1, \ldots, \bar{a}_n\}$ DNS-típusú ábécé
- $G_i = (\Sigma, P_i, \omega_i)$ D0L rendszer ($1 \leqslant i \leqslant r$),
- $\varphi : \Sigma^* \to \{0, 1\}$ egy olyan leképezés (a rendszer triggere), amelyre $\varphi(\omega_i) = \varphi(\epsilon) = 0$ ($1 \leqslant i \leqslant r$) és minden $u \in \Sigma^*$ szóra, amelyre $\varphi(u) = 1$ igaz, $\varphi(h_W(u)) = 0$ teljesül.

G_i a rendszer i-edik komponense (csúcsa) ($1 \leqslant i \leqslant r$), jelölje g_i a G_i által meghatározott homomorfizmust.

Az 1-es számú komponens kitüntetett, **mesterkomponensnek** nevezzük.

Definíció

Egy N$_r$WD0L rendszert **standardnak** nevezünk, ha $\varphi = \varphi_{\mathrm{STD}}$.

Watson-Crick D0L rendszerek állapotai

Definíció

Egy $\Gamma = (\Sigma, P_1, \omega_1, \ldots, P_r, \omega_r, \varphi)$ N$_r$WD0L rendszer **állapota** egy olyan $(L_1, \ldots, L_r)$ rendezett r-es, ahol $\forall 1 \leqslant i \leqslant r$ esetén $L_i \subseteq \Sigma^*$ és $\forall w \in L_i$-re $\varphi(w) = 0$.
$(\{\omega_1\}, \ldots, \{\omega_r\})$ a Γ hálózat **kezdeti állapota**.

NWD0L rendszerek kommunikációs protokolljai

Definíció

$s_1 = (L_1, \ldots, L_r)$-ből **közvetlenül levezethető** $s_2 = (L'_1, \ldots, L'_r)$ az **(a) protokoll szerint** (jelölése $s_1 \Longrightarrow_{(a)} s_2$), ha
$L'_i = C'_i \cup \bigcup_{j=1}^{r} h_W(B'_j)$, ahol
$C'_i = \{g_i(v) \mid v \in L_i, \varphi(g_i(v)) = 0\}$ és
$B'_j = \{g_j(u) \mid u \in L_j, \varphi(g_j(u)) = 1\}$.

Definíció

$s_1 = (L_1, \ldots, L_r)$-ből **közvetlenül levezethető** $s_2 = (L'_1, \ldots, L'_r)$ a **(b) protokoll szerint** (jelölése $s_1 \Longrightarrow_{(b)} s_2$), ha
$L'_i = h_W(B'_i) \cup \bigcup_{j=1}^{r} C'_j$, ahol
$B'_i = \{g_i(u) \mid v \in L_i, \varphi(g_i(u)) = 1\}$ és
$C'_j = \{g_j(v) \mid u \in L_j, \varphi(g_j(v)) = 0\}$.

NWD0L rendszerek kommunikációs protokolljai

Tehát mindkét protokoll esetében, miután WD0L módon alkalmaztuk a derivációs lépést, a csúcspont megtartja a helyes és a kijavított szavakat (a helytelen szavak komplemensét), és

- az (a) protokoll esetében az összes kijavított szó másolatát küldi el a többi csúcsnak,
- a (b) protokoll esetében pedig az összes helyes szóét.

A két protokollt két különböző kommunikációs stratégia motiválja:

- az (a) protokoll szerint a csúcspontok a detektált hibák kijavításáról informálják egymást,
- a (b) protokoll szerint a kapott helyes szavakról.

NWD0L rendszerek kommunikációs protokolljai

1. Példa: $r = 3$

$\Sigma_{\text{PUR}} = \{a, g\}, \Sigma_{\text{PYR}} = \{t, c\}$, $a \overset{h_W}{\longleftrightarrow} t$, $g \overset{h_W}{\longleftrightarrow} c$, φ_{STD} trigger.

(a) protokoll szerint:

$L'_1 = C'_1 \cup h_W(B'_1) \cup h_w(B'_2) \cup h_W(B'_3)$.
$L'_2 = h_W(B'_1) \cup C'_2 \cup h_w(B'_2) \cup h_W(B'_3)$.
$L'_3 = h_W(B'_1) \cup h_w(B'_2) \cup C'_3 \cup h_W(B'_3)$.

$a \to ac$	$a \to c$	$a \to a$
$g \to g$	$g \to g$	$g \to ga$
$t \to t$	$t \to tt$	$t \to t$
$c \to c$	$c \to c$	$c \to c$

$L_1 = \{acg, taa\}$	$L_2 = \{a, gaa\}$	$L_3 = \{atg, aa\}$
$C'_1 = \{accg\}$	$C'_2 = \{\}$	$C'_3 = \{atga, aa\}$
$B'_1 = \{tacac\}$	$B'_2 = \{c, gcc\}$	$B'_3 = \{\}$
$h_W(B'_1) = \{atgtg\}$	$h_W(B'_2) = \{a, cgg\}$	$h_W(B'_3) = \{\}$
$L'_1 = \{accg, atgtg, a, cgg\}$	$L'_2 = \{atgtg, a, cgg\}$	$L'_3 = \{atga, aa, atgtg, a, cgg\}$

NWD0L rendszerek kommunikációs protokolljai

2. Példa: $r = 3$

$\Sigma_{\mathrm{PUR}} = \{a, g\}, \Sigma_{\mathrm{PYR}} = \{t, c\}$, $a \overset{h_W}{\longleftrightarrow} t$, $g \overset{h_W}{\longleftrightarrow} c$, φ_{STD} trigger.

(b) protokoll szerint:

$L'_1 = C'_1 \cup h_W(B'_1) \cup C'_2 \cup C'_3$.
$L'_2 = C'_1 \cup C'_2 \cup h_w(B'_2) \cup C'_3$.
$L'_3 = C'_1 \cup C'_2 \cup C'_3 \cup h_W(B'_3)$.

$a \to ac$ $g \to g$ $t \to t$ $c \to c$	$a \to c$ $g \to g$ $t \to tt$ $c \to c$	$a \to a$ $g \to ga$ $t \to t$ $c \to c$
$L_1 = \{acg, taa\}$	$L_2 = \{a, gaa\}$	$L_3 = \{atg, aa\}$
$C'_1 = \{accg\}$	$C'_2 = \{\}$	$C'_3 = \{atga, aa\}$
$B'_1 = \{tacac\}$	$B'_2 = \{c, gcc\}$	$B'_3 = \{\}$
$h_W(B'_1) = \{atgtg\}$	$h_W(B'_2) = \{a, cgg\}$	$h_W(B'_3) = \{\}$
$L'_1 = \{accg, atgtg, atga, aa\}$	$L'_2 = \{accg, atga, a, cgg, aa\}$	$L'_3 = \{accg, atga, aa\}$

Az NWD0L rendszer által generált nyelv

Adott (x) protokoll szerint $(x \in \{a, b\})$ **többlépéses levezetést** a szokásos módon, $\Longrightarrow_{(x)}$ reflexív, tranzitív lezártjaként definiáljuk és $\Longrightarrow^*_{(x)}$-gal jelöljük.

Definíció

A $\Gamma = (\Sigma, P_1, \omega_1, \ldots, P_r, \omega_r, \varphi)$ NWD0L rendszer által (x) protokollal $(x \in \{a, b\})$ **generált nyelv**
$L_{(x)}(\Gamma) = \bigcup\{L_1 \mid (\{\omega_1\}, \ldots, \{\omega_r\}) \Longrightarrow^*_{(x)} (L_1, \ldots, L_r)\}$.

Azaz a generált nyelv a működés során valamikor a mesterkomponensbe bekerülő szavak halmaza. A generált nyelvhez tartozó szavak halmaza iterációról iterációra folyamatosan bővül.

W0L és NWD0L rendszerek számítási ereje

Néhány érdekesebb eredmény:

- A WD0L rendszereket V. Michalche és A. Salomaa vezették be (1997), az NWD0L-rendszereket Csuhaj Varjú E. és A. Salomaa (2003).
- Minden Turing-kiszámítható függvény kiszámítható WD0L rendszerrel (P. Sosík)
- A standard Watson-Crick E0L rendszerek illetve a standard Watson-Crick ETD0L rendszerek számítási ereje megegyezik a Turing gépekével. (Csima J., Csuhaj Varjú E., A. Salomaa)
- A standard NWD0L rendszerek számítási ereje megegyezik a Turing gépekével (Csuhaj Varjú E.) Sőt, a Turing teljességhez elegendő a generált szavak nem üres prefixeit kommunikálni. (Csuhaj Varjú E.)
- NWD0L rendszerek segítségével a rendszer masszív párhuzamosságát kihasználva lineáris időben oldhatók meg NP-teljes problémák (SAT, Hamilton kör). (Csuhaj Varjú E., A. Salomaa)

Számítási modellek

12. előadás

Beszúró-törlő rendszerek

Definíció

Beszúró-törlő rendszernek (InsDel rendszernek) nevezzük a $\gamma = \langle V, T, A, I, D \rangle$ rendezett 5-öst, ahol

- V egy ábécé,
- $T \subseteq V$ a terminális ábécé,
- $A \subseteq V^*$ véges nyelv (az axiómák halmaza),
- I a **beszúró szabályok** véges halmaza, elemei (u, α, v) rendezett 3-asok, ahol $u, \alpha, v \in V^*$,
- D a **törlő szabályok** véges halmaza, elemei (u, α, v) rendezett 3-asok, ahol $u, \alpha, v \in V^*$.

$(u, \alpha, v) \in I$ beszúró szabály az $uv \to u\alpha v$ szabálynak felel meg. (α-t beszúrhatjuk az (u, v) környezetbe.)

$(u, \alpha, v) \in D$ törlő szabály az $u\alpha v \to uv$ szabálynak felel meg. (α-t törölhetjük az (u, v) környezetből.)

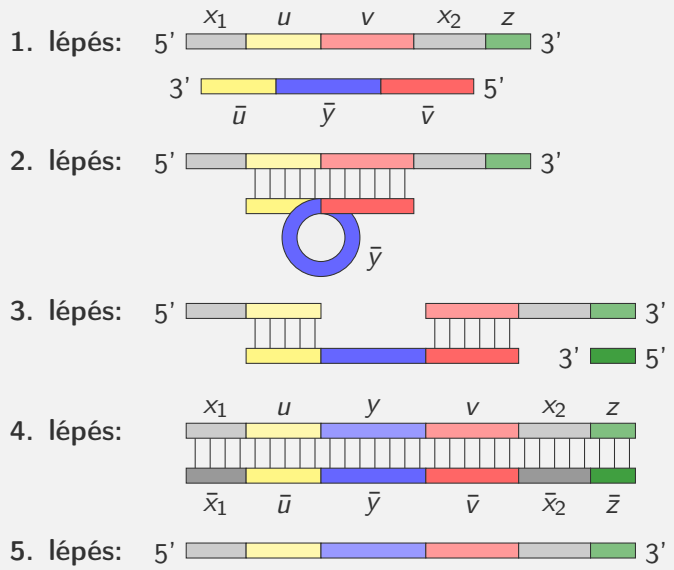
Motiváció – a DNS szerkesztése

Az evolúció során is beszúródhat vagy törlődhet, általában egyszerre csak egyetlen szimbólum a DNS szekvenciából.

y beszúrása u és v közé hibás illesztéssel:

1. Egy kémcsőben $5' - x_1uvx_2z - 3'$-hez adjuk $3' - \bar{u}\bar{y}\bar{v} - 5'$-t.
2. Hő hatására $\bar{u}$ u-hoz, $\bar{v}$ v-hez tapad meghajlítva a $\bar{y}$-t.
3. Az első szál elvágása restrikciós enzimmel u és v között
4. A hiányzó komplemensekkel való dupla szállá történő kiegészítés a $\bar{z}$ primer és egy polimeráz hozzáadására.
5. A két szál szétválasztása olvasztás hatására.

Motiváció – a DNS szerkesztése



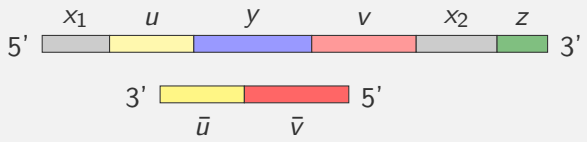
Motiváció – a DNS szerkesztése

y **törlése** u **és** v **közül hibás illesztéssel:**

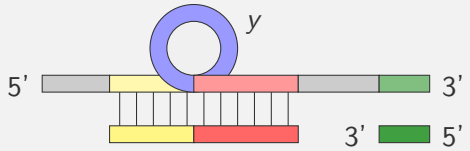
1. Egy kémcsőben $5' - x_1uyvx_2z - 3'$-hez adjuk $3' - \bar{u}\bar{v} - 5'$-t.
2. Hő hatására $\bar{u}$ u-hoz, $\bar{v}$ v-hez tapad meghajlítva y-t.
3. A $\bar{z}$ primer hozzáadására egy restrikciós emzim az első szálat elvágja u y illetve y és v között, azaz eltávolítja y-t, majd polimerizációval a hiányzó komplemensekkel való dupla szállá egészülnek ki.
4. A két szál szétválasztása olvasztás hatására.

Motiváció – a DNS szerkesztése

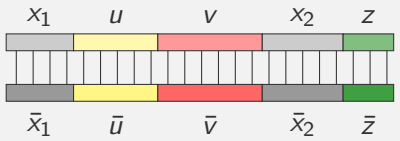
1. lépés:



2. lépés:



3. lépés:



4. lépés:

Beszúró-törlő rendszer által generált nyelv

Definíció

$x \Longrightarrow_{\text{ins}} y$ akkor és csak akkor, ha $x = x_1uvx_2, y = x_1u\alpha vx_2$ valamely $(u, \alpha, v) \in I$ -re és $x_1, x_2 \in V^*$-ra.

$x \Longrightarrow_{\text{del}} y$ akkor és csak akkor, ha $x = x_1u\alpha vx_2, y = x_1uvx_2$ valamely $(u, \alpha, v) \in D$ -re és $x_1, x_2 \in V^*$-ra.

Ekkor a **közvetlen (egylépéses) levezetést** a következőképpen definiálhatjuk: $(x \Longrightarrow y) \overset{\text{def}}{\Leftrightarrow} (x \Longrightarrow_{\text{ins}} y) \vee (x \Longrightarrow_{\text{del}} y)$

Definíció

Közvetett (többlépéses) levezetés: a $\Longrightarrow$ reflexív, tranzitív lezártja. Jelölése: $\Longrightarrow^*$.

Definíció

A $\gamma = \langle V, T, A, I, D\rangle$ beszúró-törlő rendszer által **generált nyelv**: $L(\gamma) = \{w \in T^* \mid x \Longrightarrow^* w \text{ valamely } x \in A\text{-ra}\}$

Beszúró-törlő rendszer súlya

Egy beszúró-törlő rendszer komplexitásának mértéke lehet a beszúrható/törölhető sztringek maximális hossza, illetve a beszúró/törlő kontextusok maximális hossza.

Definíció

A $\gamma = \langle V, T, A, I, D\rangle$ beszúró-törlő rendszer **súlya** $(n, m; p, q)$, ahol

- $n = \max\{\, |\alpha| \mid (u, \alpha, v) \in I \,\}$,
- $m = \max\{\, |u| \mid (u, \alpha, v) \in I \text{ vagy } (v, \alpha, u) \in I \,\}$,
- $p = \max\{\, |\alpha| \mid (u, \alpha, v) \in D \,\}$,
- $q = \max\{\, |u| \mid (u, \alpha, v) \in D \text{ vagy } (v, \alpha, u) \in D \,\}$,

γ **összsúlya** $n + m + p + q$.

1. Példa $\gamma_1 = \langle \{a, b\}, \{a, b\}, \{ab\}, \{(a, ab, b)\}, \varnothing\rangle$

$L(\gamma_1) = \{a^n b^n \mid n \geqslant 1\}$.

Ez egy (2,1,0,0) súlyú (azaz 3 összsúlyú) rendszer.

Beszúró-törlő rendszer – példa

2. Példa $\gamma_2 =$
$\langle \{S, S', a, b\}, \{a, b\}, \{S\}, \{(\varepsilon, S'aSb, \varepsilon), (\varepsilon, S'ab, \varepsilon)\}, \{(\varepsilon, SS', \varepsilon)\} \rangle$

Állítás: Minden terminális szót eredményező mondatforma néhány szomszédos SS' pár elhagyásával a^nSb^n vagy a^nb^n alakú.

Az állítás bizonyítása: A levezetés hosszára vonatkozó teljes indukcióval. Tegyük fel, hogy egy szó n levezetési lépés után ilyen alakú.

Egy szomszédos SS' pár elhagyásával továbbra is ilyen alakú marad.

A beszúró szabályok S'-vel kezdődnek. Terminális után (vagy a szó legelejére) nem történhet beszúrás, mivel akkor a további levezetés során a most beszúrt S' előtt közvetlenül mindig egy terminális állna (vagy semmi se), így nem lehetne törölni a szóból.

Beszúró-törlő rendszer – példa

S' után közvetlenül azért nem szúrhatunk be, mert bármely két S között van legalább egy terminális (minden S beszúrásakor mögé és elé is terminális kerül), és így bár az előtte lévő S' később esetleg törölhető egy S-sel együtt, de minden további S-et (a később beszúrandókat is beleértve) legalább egy terminális választaná el a most beszúrt S'-től.

Így csak közvetlenül S után történhetnek beszúrások. Ekkor viszont az indukciós feltevés alapján a szavak alakja továbbra is az állítás szerinti lesz. Ezzel az állítást beláttuk.

Tehát $L(\gamma_2) = \{a^nb^n \mid n \geqslant 1\}$.

Ez egy (4,0,2,0) súlyú (azaz 6 összsúlyú) rendszer.

Beszúró-törlő nyelvcsaládok

Definíció

$\mathrm{INS}_n^m\mathrm{DEL}_p^q := \{ L \mid L$ generálható $(n', m'; p', q')$ súlyú beszúró-törlő rendszerrel, ahol $n' \leqslant n, m' \leqslant m, p' \leqslant p, q' \leqslant q \}$

Ha az n, m, p, q paraméterek közül valamelyik nem korlátozott, akkor a megfelelő paraméter helyére $*$-t írunk.

Tehát az összes beszúró-törlő rendszer családját $\mathrm{INS}_*^*\mathrm{DEL}_*^*$ jelöli.

CF InsDel rendszerek számítási ereje

Tétel (Margenstern-Paun-Rogozhind-Verlan, 2005)

$\mathrm{INS}_*^0\mathrm{DEL}_*^0 =$ RE

Bizonyítás:
Minden RE-beli nyelv generálható 0-típusú grammatikával. Legyen $G = \langle N, T, P, S \rangle$ egy tetszőleges 0-típusú grammatika.

Minden P-beli szabály $R : u \to v$ alakú, ahol R a szabály egyedi címkéje. Jelölje M a címkék halmazát. Feltehető, hogy M és $N \cup T$ diszjunkt halmazok.

Definiáljuk az alábbi $\gamma = \langle N \cup T \cup M, T, \{S\}, I, D \rangle$ beszúró-törlő rendszert, ahol
$I = \{(\varepsilon, vR, \varepsilon) \mid R : u \to v \in P, R \in M, u, v \in (N \cup T)^*\}$ és
$D = \{(\varepsilon, Ru, \varepsilon) \mid R : u \to v \in P, R \in M, u, v \in (N \cup T)^*\}$.

A $(\varepsilon, vR, \varepsilon)$ és $(\varepsilon, Ru, \varepsilon)$ szabálypárt **M-kapcsolatban állónak** nevezzük.

CF InsDel rendszerek számítási ereje

$L(G) \subseteq L(\gamma)$.

Az az $R : u \to v$ szabályt használó G-beli $x_1ux_2 \Longrightarrow x_1vx_2$ derivációs lépés γ-ban a $x_1ux_2 \Longrightarrow_{\text{ins}} x_1vRux_2$ beszúró és a $x_1vRux_2 \Longrightarrow_{\text{del}} x_1vx_2$ törlő lépésekkel szimulálható.

$L(G) \supseteq L(\gamma)$

Állítás: Ha γ-ban $S \Longrightarrow^* \omega \in T^*$, akkor ω-nak van olyan γ-beli levezetése is, ahol minden $n \geqslant 1$-re a $2n-1$-edik és $2n$-edik lépésben alkalmazott szabály M-kapcsolatban áll.

Az állítás bizonyítása:

Legyen $\delta : S \Longrightarrow \omega_1 \Longrightarrow \omega_2 \Longrightarrow \cdots \Longrightarrow \omega_{2k} = \omega$ γ-beli levezetés.

Minden δ-ban megjelenő $R \in M$ címke törlődik is később. Ez alapján párosíthatunk minden alkalmazott beszúró szabályt egy vele M kapcsolatban lévő későbbi törlő szabállyal. Így éppen k darab egymással M-kapcsolatban álló szabálypárt kapunk.

CF InsDel rendszerek számítási ereje

Azt mondjuk, hogy ilyen pár **illeszkedik egymáshoz**, ha δ-ban közvetlenül egymást követi az alkalmazásuk.

Tegyük fel, hogy a párjaink között $0 < m \leqslant k$ darab egymáshoz nem illeszkedő pár van.

Legyen $(\varepsilon, vR, \varepsilon) \in I$ és $(\varepsilon, Ru, \varepsilon) \in D$ egy ilyen pár. Ekkor

$\delta : S \Longrightarrow^* z_1z_2 \Longrightarrow_{\text{ins}} z_1vRz_2 \Longrightarrow^+ y_1Ruy_2 \Longrightarrow_{\text{del}} y_1y_2 \Longrightarrow^* \omega$

valamely $z_1, z_2, y_1, y_2 \in (N \cup T \cup M)^*$-ra.

Ez azt jelenti, hogy δ az alábbi részekből áll:

(1) $S \Longrightarrow^* z_1z_2$,
(2) $(\varepsilon, vR, \varepsilon) \in I$ alkalmazása
(3) $z_1v \Longrightarrow^* y_1$, // szabályalkalmazások R-en nem nyúlhatnak át
(4) $z_2 \Longrightarrow^* uy_2$, // az illeszkedő párok egyazon oldalon vannak
(5) $(\varepsilon, Ru, \varepsilon) \in D$ alkalmazása
(6) $y_1y_2 \Longrightarrow^* \omega$.

CF InsDel rendszerek számítási ereje

Ezeket (1), (4), (2), (5), (3), (6) sorrendben átrendezve

(1) $S \Longrightarrow^* z_1z_2$,
(4) $z_2 \Longrightarrow^* uy_2$,
(2) $(\varepsilon, vR, \varepsilon) \in I$ alkalmazása
(5) $(\varepsilon, Ru, \varepsilon) \in D$ alkalmazása
(3) $z_1v \Longrightarrow^* y_1$,
(6) $y_1y_2 \Longrightarrow^* \omega$.

ω alábbi levezetését kapjuk:

$\delta' : S \Rightarrow^* z_1z_2 \Rightarrow^* z_1uy_2 \Rightarrow_{\text{ins}} z_1vRuy_2 \Rightarrow_{\text{del}} z_1vy_2 \Rightarrow^* y_1y_2 \Rightarrow^* \omega$.

Ezáltal illeszkedő párok nem válhattak szét, ezért így már csak legfeljebb $m-1$ nem illeszkedő szabálypárunk maradt.

Tehát a nem illeszkedő szabálypárok száma 0-ra csökkenthető, ezzel az állítást bizonyítottuk. Két egymást követő derivációs lépés γ-ban, amely M-kapcsolatban álló szabályokat használ megfelel egy G-beli levezetési lépésnek, így ω G-ben is generálható. □

CF InsDel rendszer – példa

Példa: Legyen $G = \langle \{S, X, Y\}, \{a, b, c\}, P, S \rangle$ az $L = \{a^ib^ic^i \mid i \geqslant 1\}$ nyelvet generáló grammatika a következő szabályrendszerrel

$R_1 : S \to aSX$,
$R_2 : S \to aY$,
$R_3 : YX \to bYc$,
$R_4 : cX \to Xc$,
$R_5 : Y \to bc$.

A megfelelő $\gamma = \langle V, \{a, b, c\}, \{S\}, I, D \rangle$ beszúró-törlő rendszer, ahol $V = \{S, X, Y, a, b, c, R_1, R_2, R_3, R_4, R_5\}$:

I-beli szabályok	D-beli szabályok
$(\varepsilon, aSXR_1, \varepsilon)$	$(\varepsilon, R_1S, \varepsilon)$
$(\varepsilon, aYR_2, \varepsilon)$	$(\varepsilon, R_2S, \varepsilon)$
$(\varepsilon, bYcR_3, \varepsilon)$	$(\varepsilon, R_3YX, \varepsilon)$
$(\varepsilon, XcR_4, \varepsilon)$	$(\varepsilon, R_4cX, \varepsilon)$
$(\varepsilon, bcR_5, \varepsilon)$	$(\varepsilon, R_5Y, \varepsilon)$

CF InsDel rendszerek számítási ereje

Következmény (Margenstern-Paun-Rogozhind-Verlan, 2005)

$\mathrm{INS}_3^0\mathrm{DEL}_3^0 = \mathrm{RE}$

Bizonyítás:
Ismeretes, hogy minden $G = \langle N, T, P, S\rangle$ nulladik típusú grammatika **0-adik típusú Kuroda normálformára** hozható. A normálforma alakja:
$A \to a, A \to BC, A \to \varepsilon, AB \to CD$, ahol $A, B, C, D \in N$ és $a \in T$.

Minden szabály mindkét oldalának hossza legfeljebb 2, így a Tétel bizonyításában szereplő konstrukció szerint a beszúró-törlő szabályok középső α komponensére $|\alpha| \leqslant 3$ adódik. □

CF InsDel rendszerek számítási ereje

Egy élesebb tétel a következő:

Tétel (Margenstern-Paun-Rogozhind-Verlan, 2005)

$\mathrm{INS}_3^0\mathrm{DEL}_2^0 = \mathrm{RE}$

Bizonyítás: Legyen $G = \langle N, T, P, S\rangle$ egy 0-típusú Kuroda-normálformában adott grammatika, ahol P szabályai M elemeivel injektív módon vannak címkézve és $M \cap (N \cup T) = \varnothing$.

A konstruált beszúró-törlő rendszer a következő:
$\gamma = \langle N \cup \{A' \mid A \in N\} \cup T \cup M \cup \{R', R'' \mid R \in M\}, T, \{S\}, I, D\rangle$.

Minden $R : u \to v \in P$ környezetfüggetlen szabályra hozzáadunk γ-hoz egy $(\varepsilon, vR, \varepsilon) \in I$ beszúró és egy $(\varepsilon, Ru, \varepsilon) \in D$ törlő szabályt.

Az $R : AB \to CD \in P$ nem környezetfüggetlen szabályra adjuk hozzá γ-hoz a $(\varepsilon, CDR', \varepsilon), (\varepsilon, R''B'A', \varepsilon) \in I$ és a $(\varepsilon, A'A, \varepsilon), (\varepsilon, B'B, \varepsilon), (\varepsilon, R'R'', \varepsilon) \in D$ szabályokat.

CF InsDel rendszerek számítási ereje

Ilyenkor azt mondjuk, hogy ezek a szabályok **M-kapcsolatban** állnak.

$L(G) \subseteq L(\gamma)$:

G környezetfüggetlen szabályai γ-ban hasonlóképpen szimulálhatók, mint ahogy azt az előző Tételben csináltuk.

Az $R : AB \to CD$ szabályt a következőképpen lehet szimulálni:

$x_1ABx_2 \Longrightarrow_{\mathrm{ins}} x_1CDR'ABx_2 \Longrightarrow_{\mathrm{ins}} x_1CDR'R''B'A'ABx_2 \Longrightarrow_{\mathrm{del}}^3 x_1CDx_2$.

$L(G) \supseteq L(\gamma)$:

Az előző tételben látottakhoz hasonlóan bármely γ-beli derivációs, amely nem egymást követő M-kapcsolatban álló szabályokból áll, átrendezhető úgy, hogy olyan ekvivalens derivációt kapjunk, amelyben az egymást követő lépések összeillenek.

Ezt most nem részletezzük. □

CF InsDel rendszerek számítási ereje

Tétel (Margenstern-Paun-Rogozhind-Verlan, 2005)

$\mathrm{INS}_2^0\mathrm{DEL}_3^0 = \mathrm{RE}$

Bizonyítás: Minden $G = \langle N, T, P, S\rangle$ nulladik típusú grammatika az alábbi normálformára is hozható. A normálforma alakja:

$A \to a, A \to BC, A \to \varepsilon, AB \to \varepsilon$, ahol $A, B, C, D \in N$ és $a \in T$.

Világos, hiszen egy $AB \to CD$ alakú szabály $(A, B, C, D \in N)$ az alábbi 3 szabállyal szimulálható:
$A \to CD_R, D_R \to DX_B, X_BB \to \varepsilon$, ahol D_R, X_B új egyedi nemterminálisok.
Konstruálunk egy $\gamma = \langle N \cup \{A' \mid A \in N\} \cup T \cup M, T, \{S\}, I, D\rangle$ (2,0;3,0) súlyú beszúró-törlő rendszert a következőképpen:

CF InsDel rendszerek számítási ereje

- Minden $u \to \varepsilon \in P$ szabályhoz legyen $(\varepsilon, u, \varepsilon) \in D$
- Minden $R : A \to a \in P$ esetén legyen $(\varepsilon, aR, \varepsilon) \in I$ és $(\varepsilon, RA, \varepsilon) \in D$
- Minden $A \to BC$ szabály esetén legyen $(\varepsilon, BB', \varepsilon) \in I$, $(\varepsilon, CC', \varepsilon) \in I$ és $(\varepsilon, C'B'A, \varepsilon) \in D$

Az $R : A \to BC$ szabályt a következőképpen lehet szimulálni (a többi szabályra a szimuláció nyilvánvaló):

$x_1 A x_2 \Longrightarrow_{\text{ins}} x_1 BB' A x_2 \Longrightarrow_{\text{ins}} x_1 BCC'B'Ax_2 \Longrightarrow_{\text{del}} x_1 BCx_2$

Ez bizonyítja az $L(G) \subseteq L(\gamma)$ tartalmazást.

A fordított irányú tartalmazás hasonlóan igazolható, mint az előző bizonyításokban, bebizonyítható, hogy a nem illeszkedő szabálypárok száma 0-ra csökkenthető. □

CF InsDel rendszerek számítási ereje

Lemma (Verlan, 2005)

Minden (2,0;2,0) súlyú $\gamma = \langle V, T, A, I, D \rangle$ beszúró-törlő rendszerhez megadható egy vele ekvivalens (2,0;0,0) súlyú $\gamma' = \langle T', T', A', I', \varnothing \rangle$ beszúró-törlő rendszer.

A lemmát nem bizonyítjuk. A bizonyítása többlépcsős, először megmutatható, hogy $A = \{\varepsilon\}$ esetén a nemterminálisok kiküszöbölhetők, majd ez belátható tetszőleges axiómarendszerre is, végül a törlő szabályokat is ki lehet váltani.

Tétel (Verlan, 2005)

1. $\text{INS}_2^0\text{DEL}_2^0 \subseteq \text{CF}$
2. $\text{REG} \nsubseteq \text{INS}_2^0\text{DEL}_2^0$
3. $\text{INS}_2^0\text{DEL}_2^0 \nsubseteq \text{REG}$

CF InsDel rendszerek számítási ereje

Bizonyítás:

1. A lemma szerint minden γ egy (2,0;2,0) súlyú InsDel rendszerhez $\exists\, \gamma' = \langle T, T, A, I, \varnothing \rangle$ γ-val ekvivalens (2,0;0,0) súlyú InsDel rendszer. Legyen a $G = \langle \{S, Z\}, T, P_A \cup P_I \cup \{Z \to \varepsilon\}, S \rangle$ CF grammatika a következő:

$P_A = \{\, S \to Za_1Za_2Z \cdots Za_nZ \mid a_1a_2 \cdots a_n \in A \,\}$

$P_I = \{\, Z \to ZaZbZ \mid ab \in I \,\} \cup \{\, Z \to ZaZ \mid a \in I \,\}$.

Ekkor könnyen látható, hogy $L(G) = L(\gamma')$.

2. $L = \{a^n b^m \mid n, m \geqslant 0\}$ nem generálható (2,0;2,0) súlyú InsDel rendszerrel. Valóban, tegyük fel, hogy $\gamma' = \langle T, T, A, I, \varnothing \rangle$ InsDel rendszer generálja (a lemma szerint ilyen is van). $I \neq \varnothing$, különben nem tudnánk egy végtelen nyelvet generálni. Legyen $(\varepsilon, \alpha, \varepsilon) \in I$ és $w \in L$ tetszőleges olyan szó, mely mindkét betűt tartalmazza, ekkor $\alpha^i w \alpha^i \in L (i \geqslant 0)$, de ez nem lehet.

CF InsDel rendszerek számítási ereje

3. Tekintsük a $G_n = \langle \{S\}, T_n, P_n, S \rangle$ grammatika által generált $D_n := L(G_n)$ nyelvet, ahol $T_n = \{a_1, a_1', \ldots, a_n, a_n'\}$ és $P_n = \{S \to SS, S \to \varepsilon\} \cup \{S \to a_i S a_i' \mid 1 \leqslant i \leqslant n\}$.

D_n a helyes zárójelezések nyelve n zárójelpártípussal. Ezt a nyelvet szokás Dyck-nyelvnek is nevezni.

Ismert, de könnyen be látható például a Myhill-Nerode tétellel vagy a reguláris nyelvek pumpálási lemmájával (kis Bar-Hillel lemma), hogy D_n nem reguláris.

Viszont a $\gamma_n = \langle T_n, T_n, \{\varepsilon\}, \{(\varepsilon, a_i a_i', \varepsilon) \mid 1 \leqslant i \leqslant n\}, \varnothing \rangle$ beszúró-törlő rendszer D_n-t generálja. □

Következmény $\text{INS}_2^0\text{DEL}_2^0 \subset \text{CF}$

CF InsDel rendszerek számítási ereje

No	súly	$(n, m; p, q)$	ereje	hivatkozás
1	6	(3, 0; 3, 0)	RE	Margenstern et al, 2005
2	5	(1, 2; 1, 1)	RE	Kari et al, 1997
3	5	(1, 2; 2, 0)	RE	Kari et al, 1997
4	5	(2, 1; 2, 0)	RE	Kari et al, 1997
5	5	(1, 1; 1, 2)	RE	Takahara-Yokomori, 2003
6	5	(2, 1; 1, 1)	RE	Takahara-Yokomori, 2003
7	5	(2, 0; 3, 0)	RE	Margenstern et al, 2005
8	5	(3, 0; 2, 0)	RE	Margenstern et al, 2005
9	4	(1, 1; 2, 0)	RE	Paun et al, 1998
10	4	(1, 1; 1, 1)	RE	Takahara-Yokomori, 2003
11	4	(2, 0; 2, 0)	$\subset$CF	Verlan, 2005
12	$m+1$	(m, 0; 1, 0)	$\subset$CF	Verlan, 2005
13	$p+1$	(1, 0; p, 0)	$\subset$REG	Verlan, 2005

Mivel 4-súlyúra többféle eredmény is van érdemes a súly fogalmát finomítani.

Beszúró-törlő rendszerek súlya

Definíció

Legyen $\gamma = \langle V, T, A, I, D \rangle$ beszúró-törlő rendszer

$n := \max\{|\alpha| \,|\, (u, \alpha, v) \in I\}$,
$m := \max\{|u| \,|\, (u, \alpha, v) \in I\}$,
$m' := \max\{|v| \,|\, (u, \alpha, v) \in I\}$,
$p := \max\{|\alpha| \,|\, (u, \alpha, v) \in D\}$,
$q := \max\{|u| \,|\, (u, \alpha, v) \in D\}$,
$q' := \max\{|v| \,|\, (u, \alpha, v) \in D\}$.

A beszúró-törlő rendszer **módosított súlya** az $(n, m, m'; p, q, q')$ vektorral adható meg, **módosított összsúlya** $n + m + m' + p + q + q'$.

Beszúró-törlő rendszerek súlya

Az új súlyozással az eredmények:

No	súly	$(n, m; p, q)$	ereje	új súly	$(n, m, m'; p, q, q')$
1	6	(3, 0; 3, 0)	RE	6	(3, 0, 0; 3, 0, 0)
2	5	(1, 2; 1, 1)	RE	8	(1, 2, 2; 1, 1, 1)
3	5	(1, 2; 2, 0)	RE	7	(1, 2, 2; 2, 0, 0)
4	5	(2, 1; 2, 0)	RE	6	(2, 1, 1; 2, 0, 0)
5	5	(1, 1; 1, 2)	RE	8	(1, 1, 1; 1, 2, 2)
6	5	(2, 1; 1, 1)	RE	7	(2, 1, 1; 1, 1, 1)
7	5	(2, 0; 3, 0)	RE	5	(2, 0, 0; 3, 0, 0)
8	5	(3, 0; 2, 0)	RE	5	(3, 0, 0; 2, 0, 0)
9	4	(1, 1; 2, 0)	RE	5	(1, 1, 1; 2, 0, 0)
10	4	(1, 1; 1, 1)	RE	6	(1, 1, 1; 1, 1, 1)
11	4	(2, 0; 2, 0)	$\subset$CF	4	(2, 0, 0; 2, 0, 0)

InsDel rendszerek számítási ereje

További eredmények:

No	súly	$(n, m, m'; p, q, q')$	ereje	hivatkozás
14	5	(2, 0, 0; 1, 1, 1)	RE	Krassovitskiy et al., 2008
15	6	(1, 1, 0; 1, 1, 2)	RE	Krassovitskiy et al., 2008
16	6	(1, 1, 0; 2, 0, 2)	RE	Matveevici et al., 2007
17	5	(2, 0, 0; 2, 0, 1)	RE	Krassovitskiy et al., 2008
18	5	(1, 1, 0; 1, 1, 1)	$\not\supset$REG	Krassovitskiy et al., 2008