



3.

Számítógépes morfológia

2018/2019. tanév, I. félév



Mini-nyelvészet

Morfológiatipológia és számítógépes közelítése

- ☐ A szóalakok a jelentésviszonyokat a világ nyelveiben többféle módon fejezik ki
- ☐ Izoláló nyelv
(pl. kínai, de már lassan az angol is!)
- ☐ Flektáló nyelv
(pl. német, szláv nyelvek – „túlterhelt” szóalakok)
- ☐ Agglutináló nyelv
(toldaléksorok)
- ☐ Inkorporáló (poliszintetikus) nyelv
pl. csukcs, aleut, inuktikut:
Parismunngaujumaniralauqsimanngittunga
Paris+mut+nngau+juma+niraa+lauq+si+ma+nngit+jun
(=Sose mondtam, hogy Párizsba akartam volna menni.)
- ☐ Konkatenatív és nem-konkatenatív morfológiák
(pl. arab, héber, máltai)

Morfológiai alapfogalmak

- ☐ morf
bagoly, ló, megy, ...
-hoz, -nak, -ért, ...
- ☐ allomorf
bagoly-/bagly-, ló-/lov-, megy-/men-/me-/mé-, ...
-hoz/-hez/-höz, -nak/-nek, (-ért), ...
- ☐ morféma
bagOly, lŌV, mEGY, ...
-hŌz, -nAk, -ért, ...
- ☐ szuppletív alakok
go – went, gut – besser – best, volt – van – lesz
- ☐ részlegesen szuppletív alakok
France – French – Franco-

Morfológiai alapfogalmak (2)

- ☐ inflexió = ragozás
house+s, ház+ak
- ☐ deriváció = képzés
dis+establish+ment, pázmány+os+od+ó
- ☐ igeragozás
(konjugáció)
- ☐ névszóragozás
(deklináció)
- ☐ paradigma
(maga a toldalékok egy adott tőtípushoz kapcsolódó rendszere)
- ☐ morfológiai osztályok
(a tövek viselkedési osztályai inflexióik szerint)
- ☐ lexikális osztályok
(nem feltétlenül formai kritériumokra épülő osztályok,
pl. a nemek a franciában)

Mit jelölünk morfológiai úton?

- ☐ szám
- ☐ személy
- ☐ eset
- ☐ nem
- ☐ hasonlítás: alapfok, közép fok, felső fok, túlzó fok
- ☐ idő (rég- és közel)múlt (sőt: elbeszélő, összetett, *-nd*), jelen, jövő (?)
- ☐ mód: kijelentő, felszólító, feltételes, kötő, ...
- ☐ cselekvő, műveltető, szenvedő, visszaható, ...
- ☐ igenevesítés (nem-finit alakok létrehozása): gerundium, participium, nomen actionis, infinitivus (főnévi, melléknévi, határozói, sőt igei igenevek)
- ☐ de pl. az aspektus a magyarban ritkán jelölt morfológiailag
 - nem jelöltek: a folyamatos cselekvő (ami megszakítható), pl. *keres, csinál*), a folyamatos nem-cselekvő (ami nem megszakítható), pl. *virágzik, tud*
 - lehetnek jelöltek: a befejezett ige (az igekötő „perfektál”), pl. *megindul, kitalál*, de: *villant*
- ☐ ...

Hogy jelöl a morfológia?

- ☐ affixum (=toldalék)
- ☐ szuffixum (=végződés)
- ☐ infixum
(valami hasonló: *ház+a+m*, *ház+a+i+m*)
- ☐ cirkumfixum
(valami hasonló: *ge+wander+t*, *leg+nagy+obb*)
- ☐ reduplikáció
tagalog: *sulat (ír)*, *susulat (írni fog)*
- ☐ klitikumok:
proklitikum (*a*, *dr.*)
enklitikum (*-e*)
- ☐ összetételek
Leben+s+versicherung+s+gesellschaft+s+angestellter

Atipikus/ritkább/bonyolultabb esetek a lexiko-morfológiában

- ☐ Inflexió: *oxen*, *teeth*, *formulae*, *cherubim*, *criteria*, *indices*, *mafiosi*
- ☐ Deriváció: *járda*, *lövölde*, *óvoda*; *bölcsőde*
- ☐ Összetétel: *blueberry*, *strawberry*, *raspberry*, *cranberry*; *esernyő*
- ☐ Furcsaságok (morfológiai idiómák?): *man-of-war*, *ládafia*
- ☐ Zárójelezési paradoxon a morfológia és a szintaxis határán:
((*electrical engineer*))ing)
((un(*grammatical*)))ity)
((*magyar nyelv*))ű)
- ☐ ((*barokk fuvol*))ista) vs. (*első* ((*fuvol*))ista))
((*haza(ad)*))ás) vs. (*szabad*((*rúg*))ás))
- ☐ A szemantikai viszonyok nehezen felismerhetők az összetételekben:
mosó/nő, *mosó/ruha*, *mosó/szappan*, *mosó/teknő*, *mosó/konyha*, ...



A morfológiai jelenségek formális leírása felé

A számítógépes morfológia célja

- ❑ A számítógépes morfológia olyan számítógépes technológiák és algoritmusok kialakításával foglalkozik, melyek segítségével különféle nyelvű toldalékolt szóalakok elemzése és generálása megoldható
- ❑ A számítógépes morfológia az írott alakokkal foglalkozik
- ❑ 1983: Koskenniemi vs. Winograd
- ❑ Szóalak-felismerés: a program visszaadja a toldalékolt szóalakból a szótári tövet (lemmatizálás), annak szófaját és a megjelenített nyelvtani információt
- ❑ Szóalak-generálás : a helyes szóalak előállítás a szótő és a morfológiailag releváns nyelvtani információ segítségével

Számítógépes morfológiai elemzés és generálás

❑ Példa morfológiai elemzésre:

1. Tövesítés (lemmatizálás): **dogs > dog**

2. Szófaj: **Haus > Haus/Noun**

3. Morfoszintaktikai jegyek: **went > go/Verb + Past**

4. Szóképzés (pl. török):

Finlandiyalılaştıramadıklarımızdanmışsınızcasına >

Finlandiya/Noun + Prop + A3sg + Pnon + Nom

['(behaving) as if you have been one of those whom we could not convert into a Finn(ish citizen)/someone from Finland']

5. Összetettség-elemzés:

számítógépesmorfológia-oktatás >

számítógép/Noun + N2Adj | morfológia/Noun | oktat/Verb + V2N

❑ Példa szóalak-generálásra:

kesztyű/Noun + PS-sg2-pl + Ade > kesztyűidhez

Miért kell morfológiai elemzés?

- A különböző szóalakok nagy száma miatt (típusok)
10 millió szavas angol szövegtörzs esetén $< 100\,000$,
10 millió szavas finn szövegtörzs esetében $> 800\,000$



- Toldalékoló nyelvek esetében tetszőleges szövegtörzshöz igaz az alábbi állítás: a szövegtörzsben aktuálisan előforduló szövegszó-típusok száma kisebb, mint azoknak a lehetséges szótípusoknak a száma, melyek nincsenek benn az aktuális törzsben

A számítógépes morfológia fontosabb technikai ismérvei

- ☐ A (minél nagyobb) szókészlet
- ☐ A (lehetőleg teljes) toldalékkészlet
- ☐ Az ismeretlen alakok kezelése
- ☐ Az elemzéshez választott módszer
- ☐ A lexikonok ábrázolásának módja

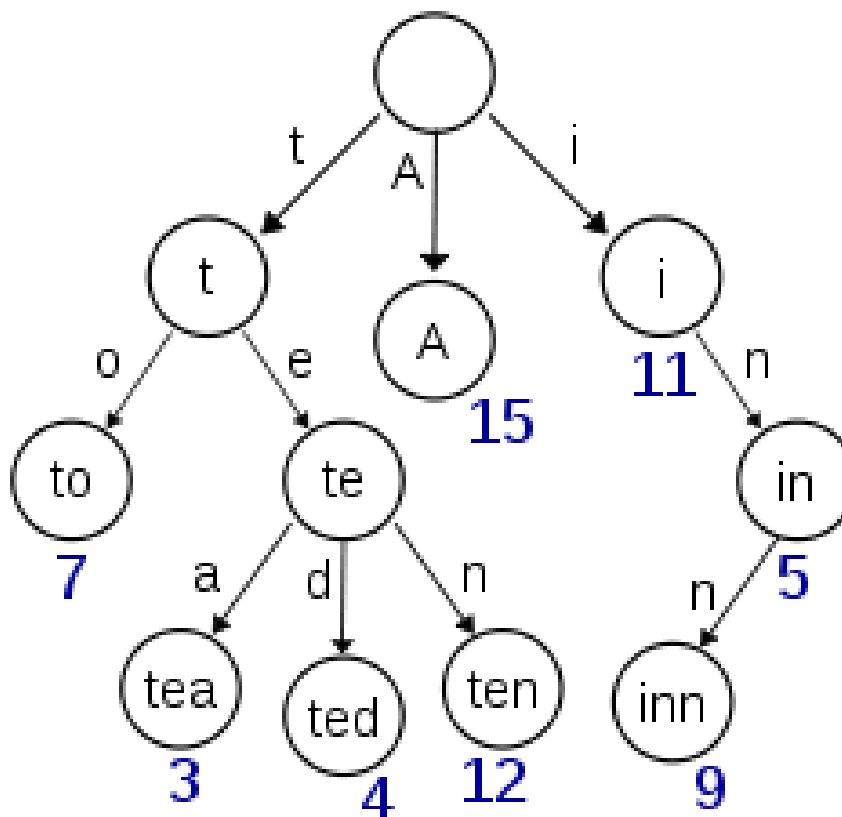


Néhány szó a szótárak tárolásáról

A szófa (=trie)

(A, i, in, inn, to, tea, ted, ten)

A szófa (Fredkin 1960) egy olyan, a szavak rákövetkező karaktereivel címkézett élsorozatokot tartalmazó fa, amelyben egy szót úgy találunk meg, hogy végigjárjuk karakterenként.



A szófák általános tulajdonságai

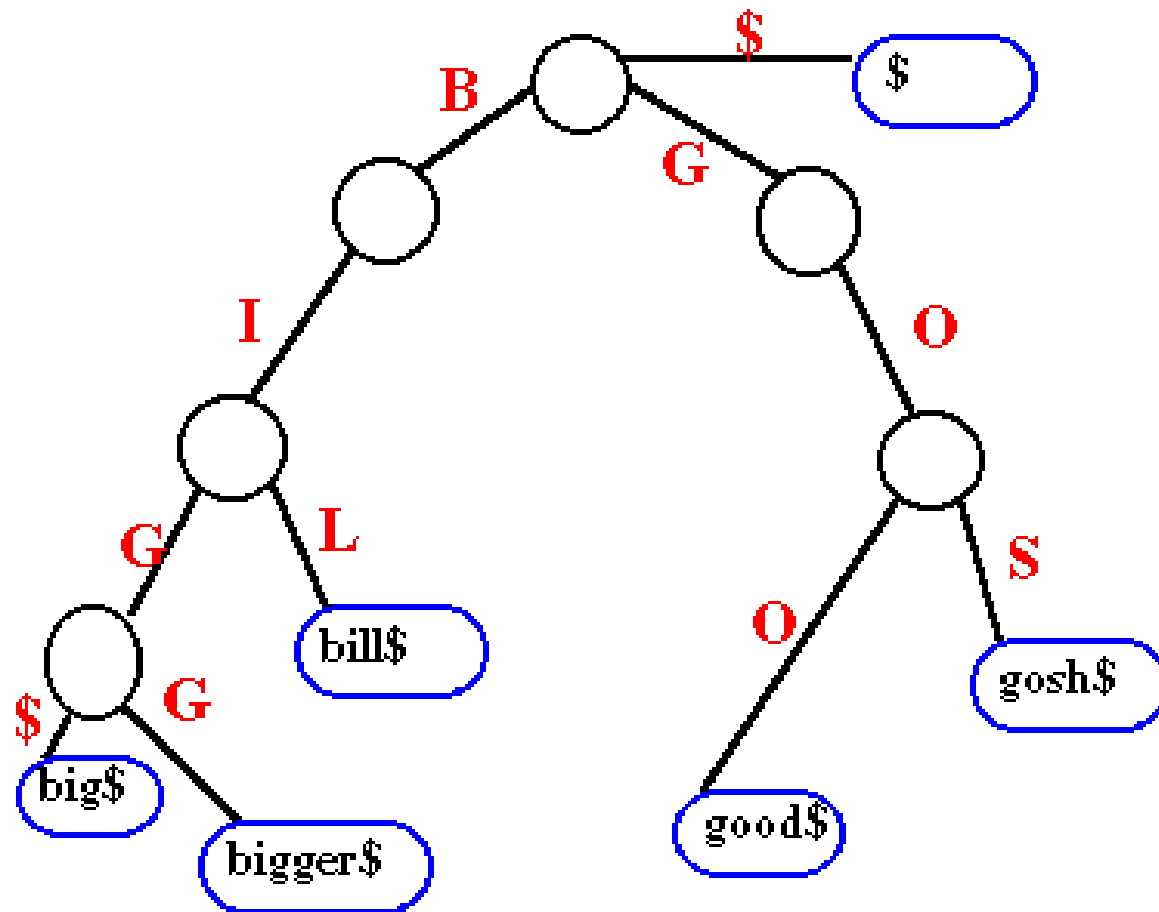
- ☐ Bináris keresőfák: $O(m \log(n))$ (n a fában tárolt elemek száma)
- ☐ Szófa: m hosszúságú kulcs megtalálása max. $O(m)$
- ☐ Nagy számú rövid füzér tárolása esetén a szófa kevesebb helyet igényel, mint a bináris keresőfa (ui. a kulcsokat nem tároljuk, a csomópontokat meg közösen használják az egyforma kezdőszeletű füzérek kulcsai)
- ☐ Hasítótáblák helyett is használható (bár néha a szófák lassabbak)
- ☐ Nem mindenre jó: vannak füzérként nehezen ábrázolható kulcsok (pl. a lebegőpontos számok)
- ☐ De: szótárak ábrázolására alkalmas
- ☐ ADFSA (körmentes determinisztikus véges automata) a szófánál is jobb, de csak ha nincs kiegészítő információ (csak puszta szólista)

(big, bigger, bill, good, gosh)



Módosított (kompakt) szófa

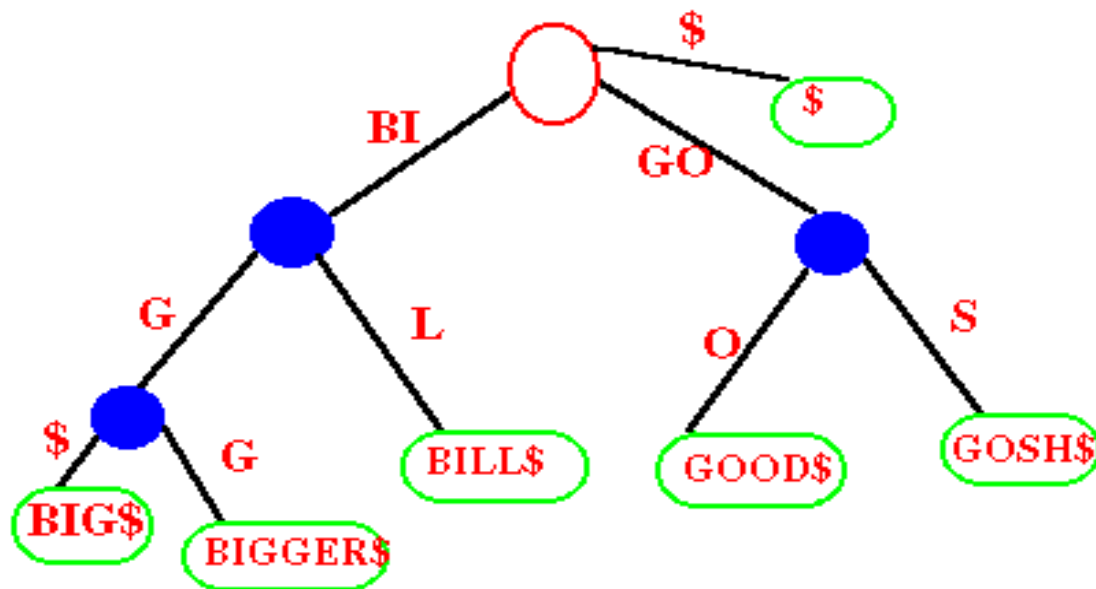
(big, bigg/er, bil/l, goo/d, gos/h)



Erősen módosított (PATRICIA) szófa

(big, bigg/er, bil/l, goo/d, gos/h)

- ❑ PATRICIA = Practical Algorithm to Retrieve Information Coded in Alphanumeric (Donald R. Morrison, 1968)



- ❑ Bármely élen több karakter is lehet, pl. az előtagok (igekötők, *re-*, *pre-*, *anti-* stb.) vagy a tipikus és ritka kezdő betűpárok
- ❑ Angol: a $26^2=676$ indító betűpárból csak 309 létezik (amiből 88 csak 15-nél kevesebb szó elején)

A Kay-féle szóábrázolás

(*alma, alom, anya, anyag, apa, apad, aránytalanság*)

- ☐ Kay (1977): tömörítés numerikus prefixekkel

alma – 0

alom – 2

anya – 1

anyag – 4

apa – 1

apad – 3

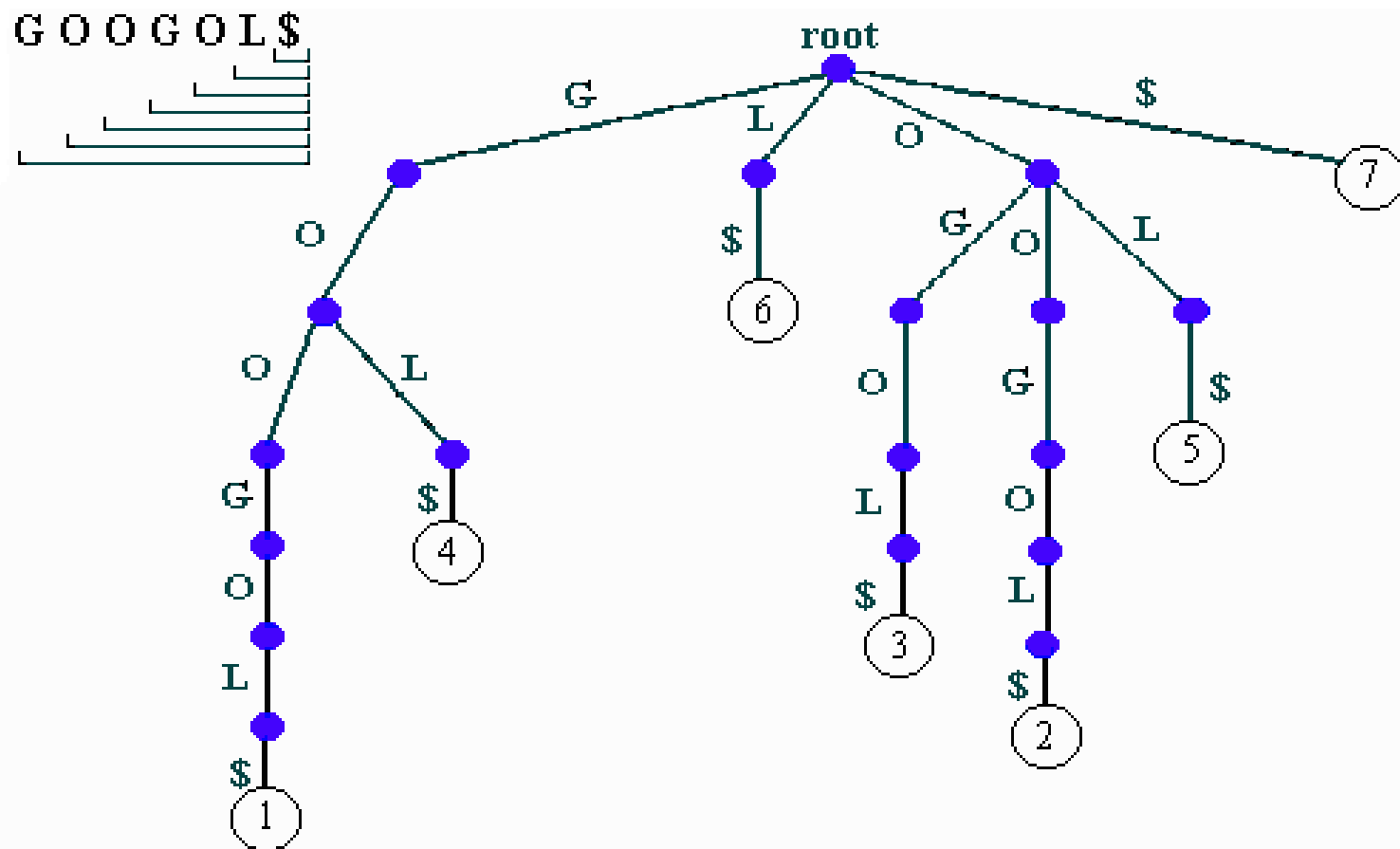
aránytalanság – 1

- ☐ Tehát a szótár:

alma, 2om, 1nya, 4g, 1pa, 3d, 1ránytalanság

- ☐ Akkor éri meg, ha hasonlítanak a szókezdetek
(nagy szótár esetén mindig!)

Szuffixum-szófa



- ☐ Egy füzér minden lehetséges végszelete tárolva
- ☐ A szuffixum-szófának n levele van és a magassága is n

A szuffixum-szófa néhány tulajdonsága

Egy n hosszú S füzérre épített általánosított szófára (többek közt) az alábbiak állnak:

- ☐ $O(m)$ időben eldönthető, hogy egy m hosszú P füzér a részfüzére-e
- ☐ $O(m)$ időben megtalálható egy tetszőleges m hosszú P részfüzérének első előfordulása
- ☐ $O(m+z)$ időben megtalálható mind a z darab előfordulása egy m hosszú részfüzérének
- ☐ Az S_i és az S_j füzérek leghosszabb közös részfüzére megtalálható $O(n_i + n_j)$ idő alatt



Automaták és véges állapotú morfológiák

Nyelvek, nyelvtanok, automaták

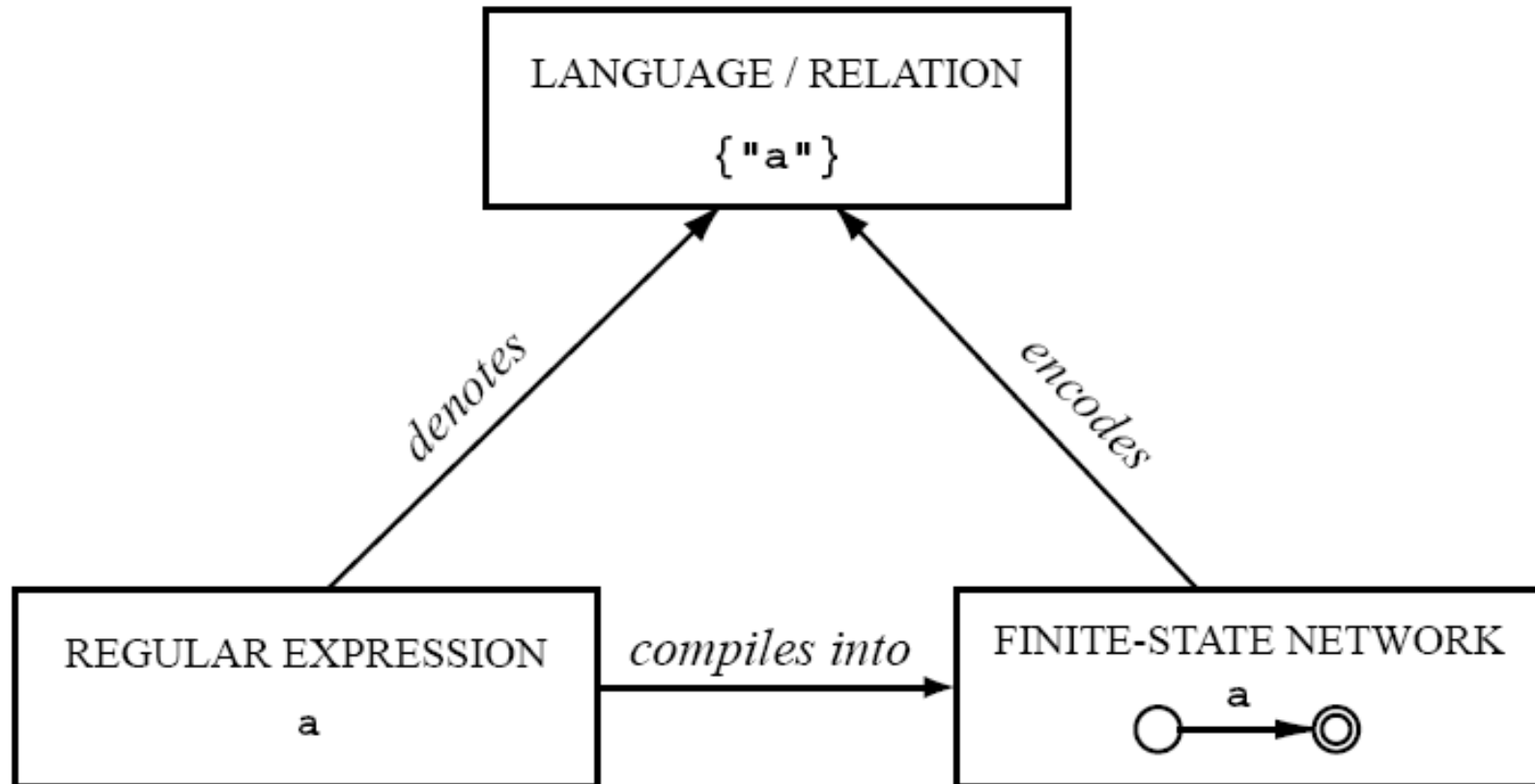
- ❑ **A formális nyelv** egy formális nyelvtan által leírt füzérek halmaza
- ❑ $G=(N, T, R, S)$ egy **formális nyelvtan**, ahol
 - N nemterminális szimbólumok véges halmaza
 - T terminális szimbólumok véges halmaza
 - R $(T \cup N)^* N (T \cup N)^* \rightarrow (T \cup N)^*$ alakú szabályok véges halmaza
 - $S \in N$ mondat-szimbólum
- ❑ Formális nyelvek és formális nyelvtanok **Chomsky-hierarchiája**
ahol $A, B \in N$; $a \in T$; $\alpha, \beta, \gamma \in (T \cup N)^*$:
 - 0-típusú nyelvtan $(\alpha \rightarrow \beta)$: rekurzív megszámlálható nyelvek
 - 1-típusú nyelvtan $(\alpha A \beta \rightarrow \alpha \gamma \beta)$: környezetfüggő nyelvek
 - 2-típusú nyelvtan $(A \rightarrow \gamma)$: környezetfüggetlen nyelvek
 - 3-típusú nyelvtan $(A \rightarrow a \text{ and } A \rightarrow aB)$: reguláris nyelvek
- ❑ Az **elfogadó automaták hierarchiája**:
 - 0: Turing-gép
 - 1: Lineárisan kötött automata
 - 2: Veremautomata – $O(n^3)$
 - 3: Véges állapotú automata – $O(n)$



Véges állapotú automaták

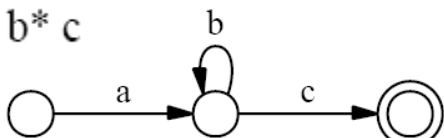
- ❑ **Véges állapotú automata:** $A = (S, \Sigma, s, F, T)$ ötös, ahol
 - S állapotok véges halmaza
 - Σ egy ábécének nevezett véges halmaz
 - $s \in S$ kiinduló állapot
 - $F \subseteq S$ a végállapotok halmaza
 - $T: S \times (\Sigma \cup \{\varepsilon\}) \rightarrow S$ átmenetfüggvény
- ❑ Az $X = x_0 x_1 \dots x_n$ Σ ábécéből alkotott füzért **A elfogadja**, ha létezik S -ben az átmenetek r_0, r_1, \dots, r_n sorrendje a következő feltételekkel:
 - i. $r_0 = s$
 - ii. $r_{i+1} = T(r_i, x_i) \quad (i = 0, \dots, n-1)$
 - iii. $r_n \in F$
- ❑ **Reguláris nyelv:** a véges állapotú automata által elfogadott füzérek halmaza
- ❑ **Reguláris kifejezés:** olyan formula, mely konkatenáció, unió és iteráció használatával meghatároz egy reguláris nyelvet

Reguláris kifejezés – reguláris nyelv – véges gép

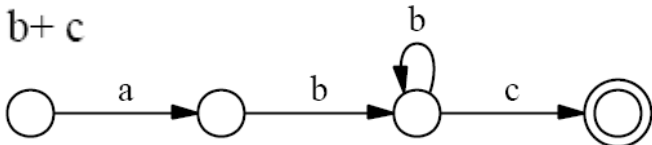


Reguláris kifejezések mint automaták

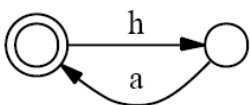
$a b^* c$



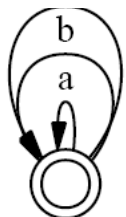
$a b^+ c$



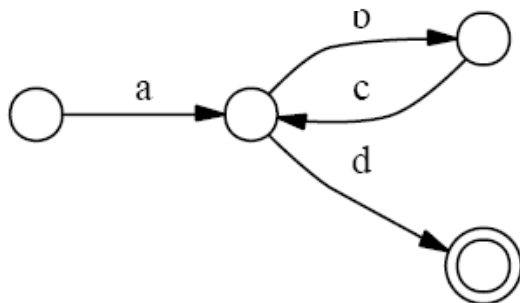
$[h a]^*$



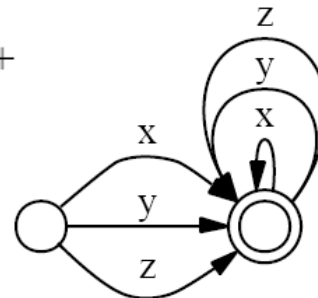
$[a | b | c]^*$



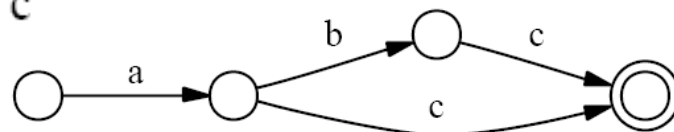
$a [b c]^* d$



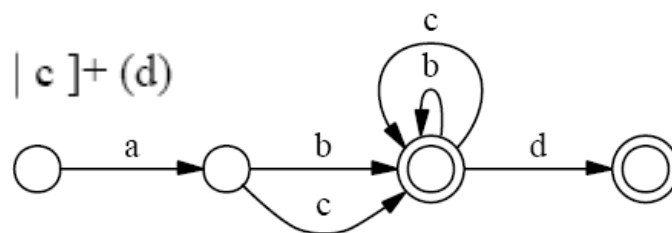
$[x | y | z]^+$



$a (b) c$



$a [b | c]^+ (d)$



Véges átalakítók (FST)

- ❑ **Véges átalakító:** $T = (S, \Sigma, \Gamma, s, F, \delta)$ hatos, ahol
 - S az állapotok véges halmaza
 - Σ a bemenő ábécének nevezett véges halmaz
 - Γ a kimenő ábécének nevezett véges halmaz
 - $s \in S$ a kezdő állapot
 - $F \subseteq S$ az elfogadó állapotok halmaza
 - $T: S \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow S$ átmenetfüggvény
- ❑ A T **átalakítja** az $\alpha \in \Sigma^*$ füzért $\beta \in \Gamma^*$ füzérbe (röviden: $\alpha[T]\beta$) ha létezik út a kezdőállapotból egy végállapotba α bemenősorozat és β kimenősorozat mellett
- ❑ Az **FSA** és a **FST** különbsége: az FSA kimenetén egy Boole-válasz jön létre, míg az FST egy füzért ad eredményül (a másik szalag tartalmát)

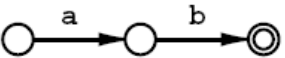
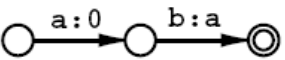
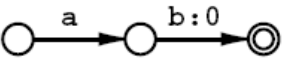
A reguláris nyelvek és a véges automaták helye

- ❑ $G=(N, T, R, S)$ egy **formális nyelvtan**, ahol
 - N nemterminális szimbólumok véges halmaza
 - T terminális szimbólumok véges halmaza
 - R $(T \cup N)^* N (T \cup N)^* \rightarrow (T \cup N)^*$ alakú szabályok véges halmaza
 - $S \in N$ mondat-szimbólum
- ❑ Formális nyelvek és formális nyelvtanok **Chomsky-hierarchiája**
ahol $A, B \in N$; $a \in T$; $\alpha, \beta, \gamma \in (T \cup N)^*$:
 - 0-típusú nyelvtan $(\alpha \rightarrow \beta)$: rekurzív megszámlálható nyelvek
 - 1-típusú nyelvtan $(\alpha A \beta \rightarrow \alpha \gamma \beta)$: környezetfüggő nyelvek
 - 2-típusú nyelvtan $(A \rightarrow \gamma)$: környezetfüggetlen nyelvek
 - 3-típusú nyelvtan $(A \rightarrow a \text{ and } A \rightarrow aB)$: reguláris nyelvek
- ❑ Az **elfogadó automaták hierarchiája**:
 - 0: Turing-gép
 - 1: Lineárisan kötött automata
 - 2: Veremautomata – $O(n^3)$
 - 3: Véges állapotú automata – $O(n)$

Műveletek véges átalakítókkal

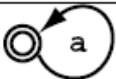
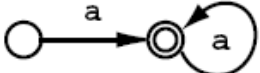
Konkatenáció:

 $w[T]y$ és $x[S]z$
 $acsa$
 $wx[TS]yz$

Expression	Language / Relation	Network
$a \circ b$	$\{“ab”\}$	
$a:0 \quad b:a$	$\{<“ab”, “a”>\}$	
$a \quad b:0$	$\{<“ab”, “a”>\}$	

Unió:

 $x[T]y$ vagy $x[S]y$
 $acsa$
 $x[TUS]y$

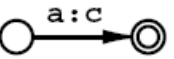
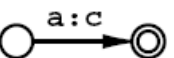
Expression	Language / Relation	Network
a^*	$\{“, “a”, “aa”, \dots\}$	
a^+	$\{“a”, “aa”, \dots\}$	

Iteráció:

 $w[T^*]y$ és $x[T]z$
 $acsa$
 $wx[T^*]yz$ és $\epsilon[T^*]\epsilon$

Metszet:

 $x[T]y$ és $x[S]y$
 $acsa \quad x[T \times S]y$

Expression	Language / Relation	Network
$a:b \quad .o. \quad b:c$	$\{<“a”, “c”>\}$	
$a:b \quad .o. \quad b \quad .o. \quad b:c$	$\{<“a”, “c”>\}$	

Kompozíció:

 $x[T]y$ és $y[S]z$
 $acsa$
 $x[T \circ S]z$



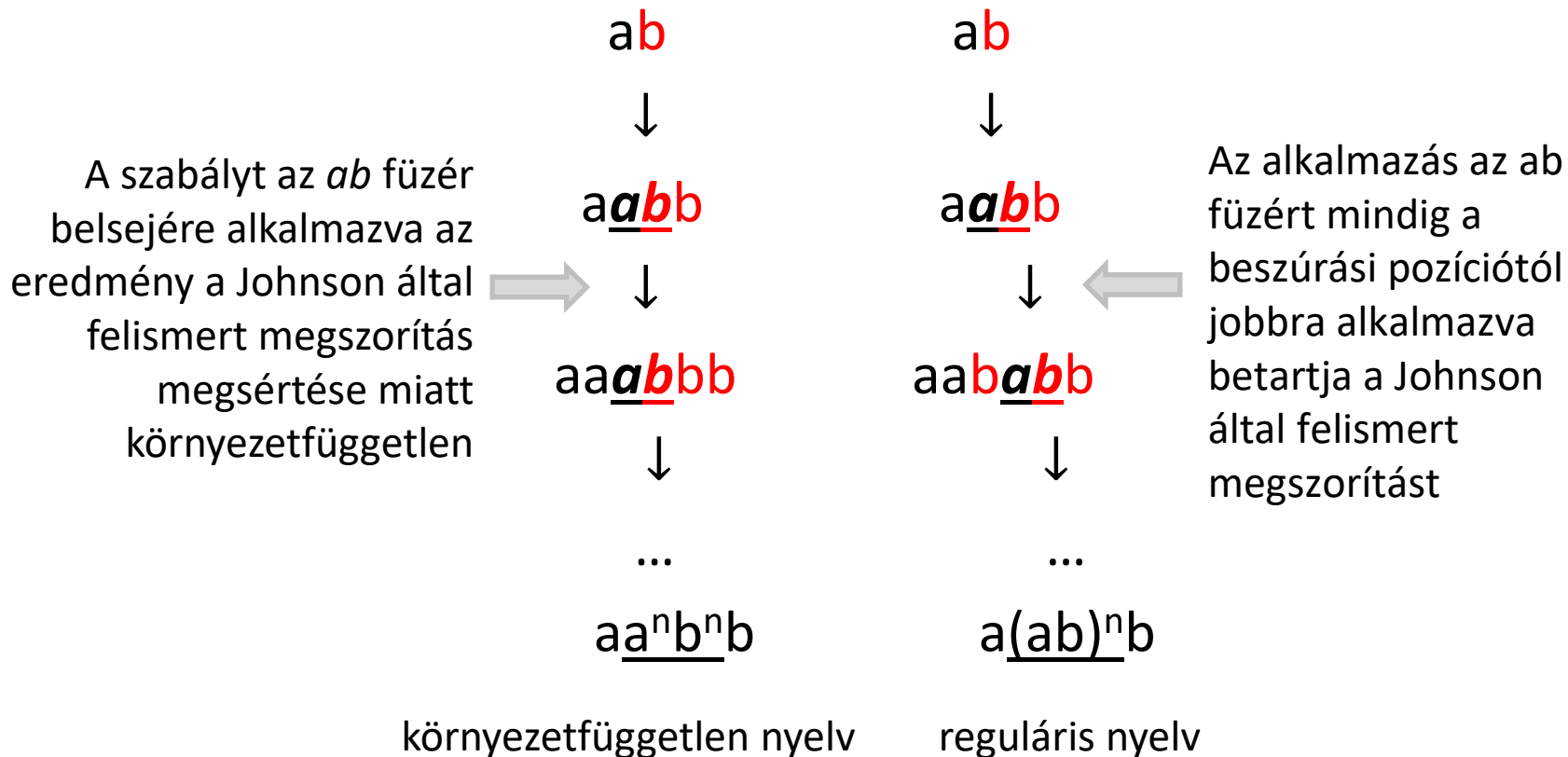
A kétszintes leírás felé...

Johnson felfedezése (1972)

- ❑ A generatív fonológiában, ha az $\alpha \rightarrow \beta / \gamma _ \delta$ szabállyal állítjuk elő a $\gamma\beta\delta$ füzért a $\gamma\alpha\delta$ füzérből, akkor a szabály bármely rákövetkező alkalmazása a β füzért érintetlenül hagyja, azaz csak γ és δ füzéreket érinti
- ❑ C. Douglas Johnson a *Formal Aspects of Phonological Description* (1972) című művében észrevette, hogy egyazon generatív szabály ismételt alkalmazása esetén nem szokás érinteni az előző alkalmazás kimenetét, hanem az újabb alkalmazás attól vagy balra vagy jobbra helyezkedik el
- ❑ Következmény: ha egyazon ciklusban nem alkalmazzuk az újraíró szabályt saját kimenetére, akkor a bemenet-kimenet párok leírhatók egy **reguláris relációval** (példa a következő dián!)
- ❑ Amint tudjuk korábbról, a reguláris reláció megfeleltethető egy reguláris nyelvnek, ami ekvivalens egy véges állapotú átalakítóval
- ❑ Johnson megmutatta, hogy a generatív fonológia szabályai sokkal kevésbé „erősek”, mint ahogy ezt a leírásukhoz használt (környezetfüggő) formalizmus sugallja: generatív képességük a fenti megszorítás általános elfogadottsága miatt az 1-es Chomsky-nyelvosztály helyett a 3-as nyelvosztályba sorolja őket

Johnson felismerése egy példán

$$\varepsilon \rightarrow ab / _ b$$



Kaplan & Kay (1981/1994)

- ☐ Kaplan és Kay (1981): egy hatékony elemző irányában kutatva leírták az újraíró szabályok átalakítókba való fordításának algoritmusát
- ☐ Azt vették észre, hogy a reguláris relációk zártak a soros kompozícióra nézve: ha veszünk két olyan szabályt, amit átalakítóval modellálunk, akkor ha az első átalakító kimenete a másik bemenete, a kompozíció művelete segítségével helyettük **egyetlen ekvivalens átalakítót** kaphatunk
- ☐ A kompozíció eredményeként kapott gép az első átalakító bemenetét a második kimenetére úgy képezi le, hogy **nem generál semmiféle köztes eredményt**
- ☐ Tetszőleges számú fonológiai újraíró szabályt sorban alkalmazva reguláris relációt kapunk, az alkalmazott **szabályok számától függetlenül**
- ☐ Megjegyzés: nincs olyan művelet az újraíró szabályok eredeti világában, ami ugyanerre volna képes

A szabálymegfordítás problémája

Egy (nem létező nyelvből származó) példa:

labiális realizáció p előtt: $N \rightarrow m$,

dentális realizáció egyébként: $N \rightarrow n$;

m után: $p \rightarrow m$

Az újraíró szabályok tehát:

$N \rightarrow m / __ p$; elsewhere, n .

$p \rightarrow m / m __$

Lexikálisból felszíni alak (generálás):

$kaNpat \Rightarrow kammatt$

Felszíniből lexikális alak (elemzés):

$kammatt \Rightarrow \{kaNpat, kampat, kammatt\}$

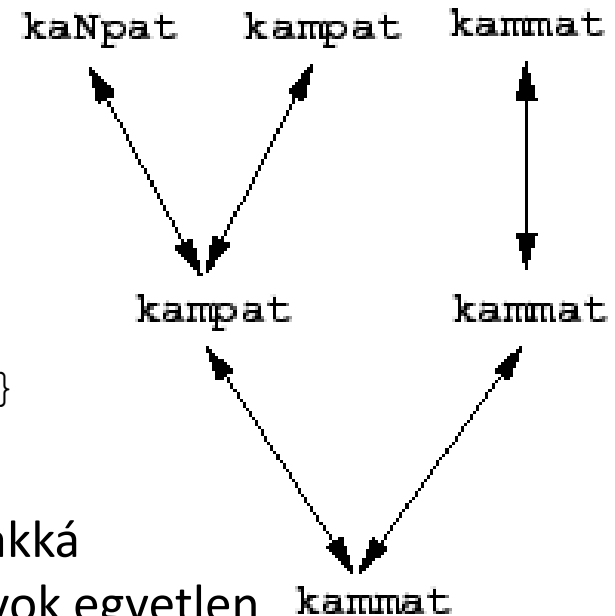
Ugyanazon generáló szabályok elemzésre való

használatakor a kötelező szabályok opcionálisakká

válhatnak: a $kaNpat$ lexikális füzérből a szabályok egyetlen $kammatt$

felszíni alakot generálnak, de a felszíni alakból az inverz

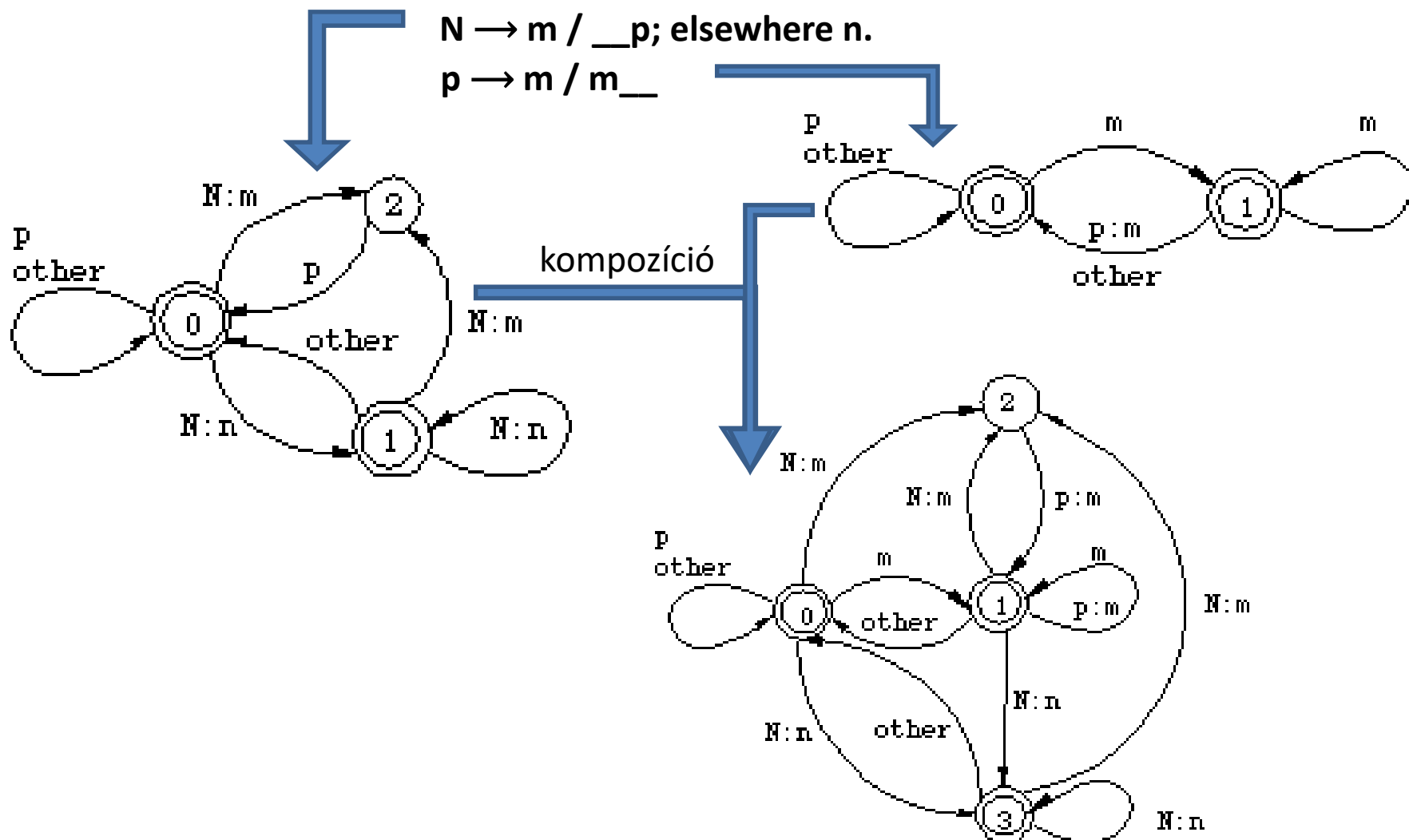
leképezés „egy a sokhoz” típusú is lehet



Figures from <http://www.ling.helsinki.fi/~koskenni/essli-2001-karttunen/>

A 'kaNpat' automata

$$N \rightarrow m / _p; \text{elsewhere } n.$$

$$p \rightarrow m / m_$$


A korábbi beszűrőszabály véges automatája

$\varepsilon \rightarrow ab / _ b$

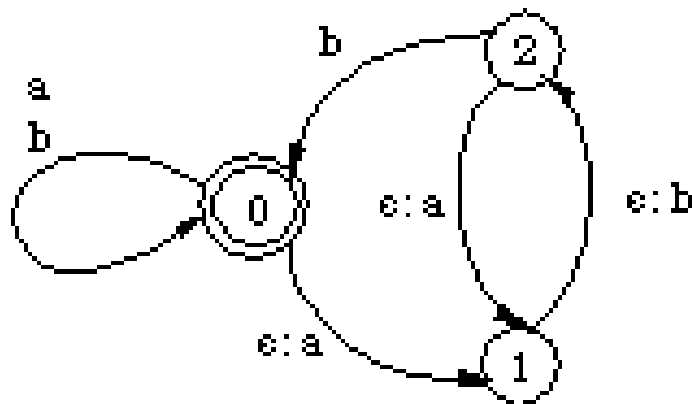
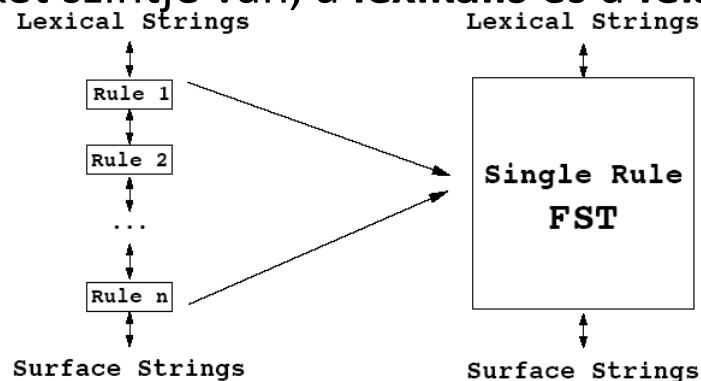


Figure from <http://web.stanford.edu/~laurik/publications/fsc-91/fsc91.html>

Összegzés: véges átalakítók soros kapcsolása

- ❑ A fonológiai levezetések köztes állapotai mindig kiküszöbölhetők az egyes szabályokból kapott átalakítók kompozíciójával: az eredményül kapott átalakítónak csak két szintje van, a **lexikális** és a **felszíni**



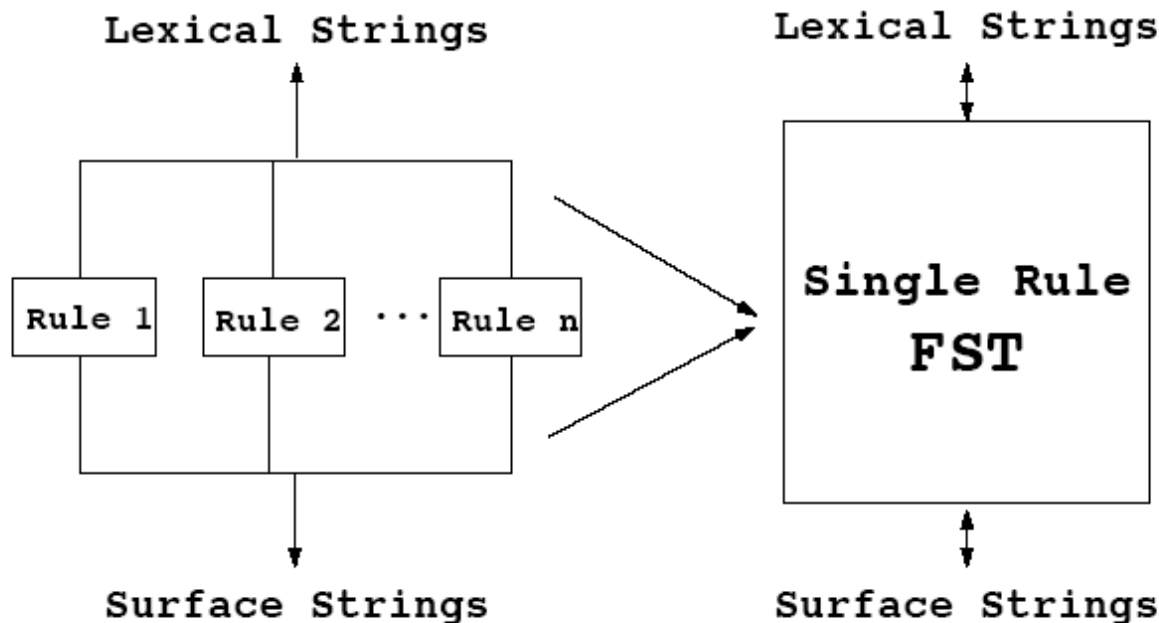
Figures from <http://www.ling.helsinki.fi/~koskenni/esslli-2001-karttunen/>

- ❑ Az egyetlen generatív átalakító használata **sokkal hatékonyabb** felismerésre is, mintha az eredeti szabályoknak megfelelő átalakítók egyenként invertált sorrendben működnének
- ❑ Kaplan és Kay megoldotta ugyan az egyes szabályok sorozata átalakítóba való fordításának problémáját, azonban a nagy szabályrendszerek egyetlen átalakítóba való kompozíciója az akkori technikai korlátok miatt a **gyakorlatban megvalósíthatatlan** volt

Véges átalakítók párhuzamos kapcsolása

Kimmo Koskenniemi PhD-értekezése: *Two-level Morphology* (1983)

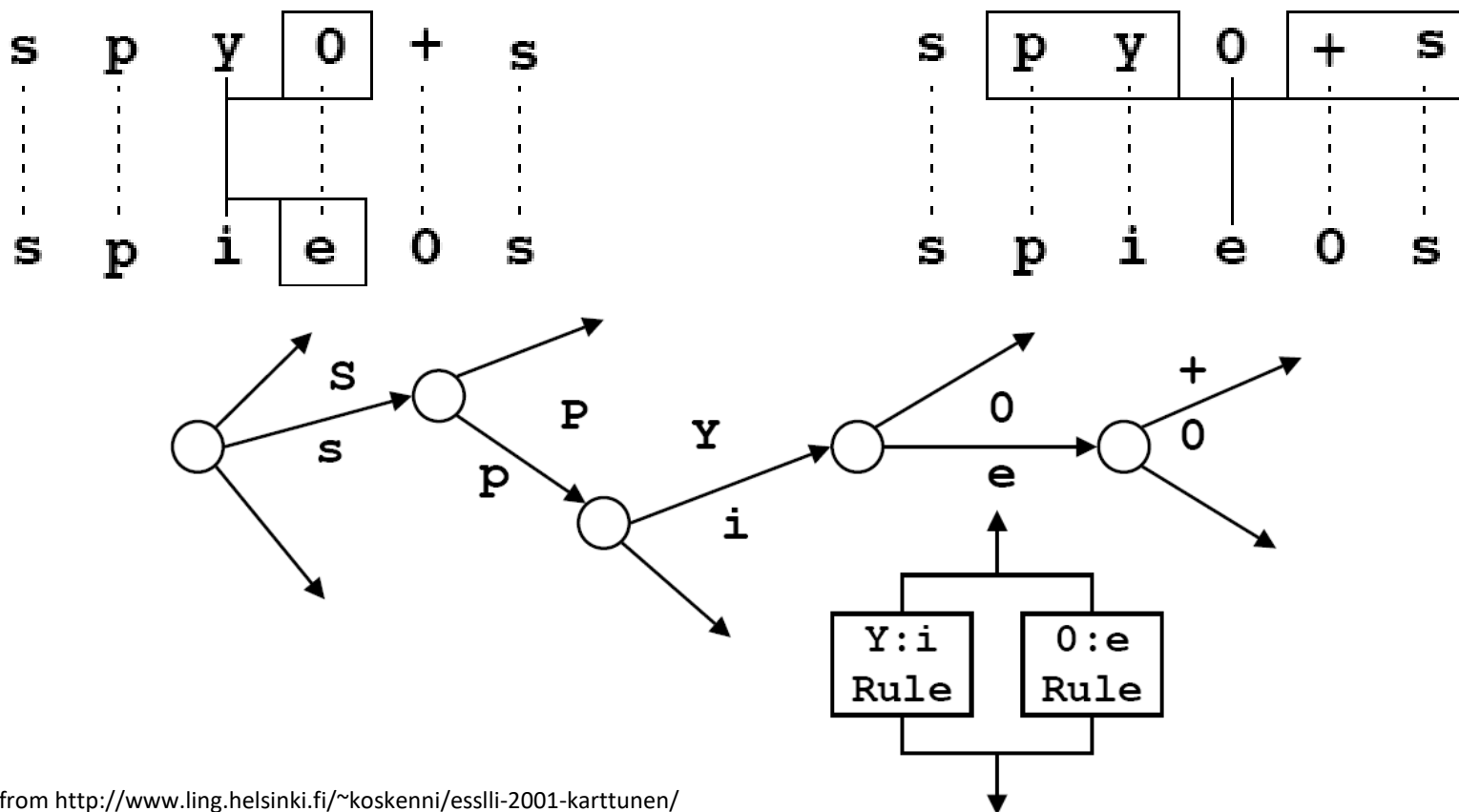
Tudva, hogy a lexikális és a felszíni alakok közötti megfeleltetés leírható reguláris relációval, Koskenniemi egy lényegi változtatást, a szabályhalmazból származó átalakítók **párhuzamos** kapcsolását javasolta



Figures from <http://www.ling.helsinki.fi/~koskenni/esslli-2001-karttunen/>

Egy „igazi” (angol nyelvi) példa

$$y:i \Leftrightarrow _ 0:e$$



Figures from <http://www.ling.helsinki.fi/~koskenni/esslli-2001-karttunen/>

A kétszintes szabályok alakja

- ➔ Egy kétszintes szabály az alábbi három részből áll:
 - **megfeleltetett párok:** amiről valójában a szabály „szól”
 - **környezet** (bal környezet (*lc*) és/vagy jobb környezet (*rc*))
 - **szabályoperátor**
- ➔ **Megfeleltetett párok:** egy lexikális és egy felszíni karakterből álló pár (pl. *t:c* ahol a lexikális *t* megfelel a felszíni *c*-nek)
- ➔ **Környezet:** az adott jelenséget körülvevő fonológiai helyzetet specifikálja (az aláhúzás jelenti a a megfeleltetett pár helyzetét az adott környezetben)
- ➔ **Szabályoperátor:** a megfeleltetett pár és a környezete között fennálló reláció; nagyjából a formális logika feltételeket és következményeket leíró operátorainak felelnek meg:
 - ⇒ a megfeleltetés *csak akkor* áll fenn, ha ez a környezet
 - ⇐ ha ez a környezet, *akkor* a megfeleltetés fennáll
 - ⇔ a megfeleltetés *akkor és csak akkor* áll fenn, ha ez a környezet
 - ⇎ a megfeleltetés *soha* nem fordul elő ebben a környezetben (*/⇐*)

Generatív vs. kétszintes szabályok

- ❑ A $t \rightarrow c / __ i$ generatív szabály azt jelenti, hogy
 - a t átalakul c -be, ha i előtt áll, és miután újraírtuk t -t c -vé, t **többé nem létezik**
 - A generatív szabályok sorozata **tetszőleges számú köztes szint** segítségével konvertál beleső reprezentációkat felszíni formákra
 - A generatív szabályok **egyirányúak**: csak belső reprezentációkból tudnak felszíni alakokat létrehozni, fordított irányban nem alkalmazhatók

- ❑ Az ezzel analóg kétszintes szabály: $t : c \Rightarrow __ i$
 - A lexikális t megfelel a felszíni c -nek i előtt; de nem változik át c -vé, hanem **megmarad** t -ként a szabály alkalmazása után is
 - A kétszintes szabályok egyfajta megfeleltetést fejeznek ki, nem újraírást, párhuzamosan alkalmazandók, és nem sorosan, és az újraírással szemben nem hoznak létre **semmilyen köztes reprezentációs szintet**
 - A kétszintes szabályok **kétirányúak** és deklaratívak: bizonyos megfeleltetéseket fogalmazznak meg a lexikális és a felszíni formák között

"Kizárólag, de nem mindig"

$$L:S \Rightarrow E$$

- ☐ Mit jelent?
 - L csak E-ben realizálódik S-ként
 - L S-ként való realizáció nincs megengedve \neg E-ben
 - Ha L:S, akkor ennek E-ben kell lennie
 - De: L: \neg S megengedett E-ben
- ☐ Logikailag: az \Rightarrow operátor azt jelenti, hogy a megfeleltetésből következik a környezet, de a környezetből nem feltétlenül következik a megfeleltetés
- ☐ Példa: **t:c** \Rightarrow **i** (a lexikális **t** megfelel csak **i** előtt felel meg a felszíni **c**-nek, de ebben a környezetben nem feltétlenül mindig; azaz a lexikális **t** más realizációi is előfordulhatnak ebben a környezetben, beleértve a **t:t** párt is)
- ☐ Negatív megfogalmazásban: ez a szabály letiltja a **t:c** pár minden olyan előfordulását, ami nem **i** előtt van
- ☐ A \Rightarrow szabály nagyjából a generatív fonológia opcionális szabályának felel meg, és tipikusan a szabad variációk leírására használják

"Mindig, de nem kizárólag"

$$L:S \Leftarrow E$$

- ☐ Mit jelent?
 - L E-ben mindig S-ként realizálódik
 - $L \rightarrow S$ -ként való realizációja nincs megengedve E-ben
 - Ha L E-ben van, akkor L:S-nek kell lennie
 - De: L:S előfordulhat máshol
- ☐ Logikailag: az \Leftarrow operátor azt jelenti, hogy a környezetből következik a megfeleltetés, de a megfeleltetésből nem feltétlenül következik a környezet
- ☐ Példa: $t:c \Leftarrow __ i$ (a lexikális t i előtt mindig kötelezően megfelel a felszíni c -nek, de nem szükségszerűen csak ebben a környezetben; azaz a $t:c$ előfordulhat más környezetben is)
- ☐ Negatív megfogalmazásban: ha a $t : \neg c$ pár azt jelenti, hogy a lexikális t minden felszíni alaknak megfelelhet, ami nem c , akkor a fenti szabály megtiltja a $t : \neg c$ előfordulását az adott környezetben
- ☐ A \Leftarrow szabály nagyjából a generatív fonológia kötelező szabályának felel meg, és tipikusan akkor használják, amikor a megfeleltetés kötelező egy adott környezetben, de előfordulhat más környezetben is

"Mindig és kizárólag"

$$L:S \Leftrightarrow E$$

- ☐ Az \Leftarrow és az \Rightarrow operátor kombinációja
- ☐ Mit jelent?
 - L S-ként akkor és csak akkor realizálható, ha E a környezet
 - Mind $L:S \Rightarrow E$, mind $L:S \Leftarrow E$ fennáll
 - L:S kötelező E-ben, de máshol sehol
- ☐ Példa: **t:c** \Leftrightarrow ___ **i** (a lexikális **t** akkor és csak akkor felel meg a lexikális **c**-nek, ha **i** előtt áll)
- ☐ A \Leftrightarrow szabályt akkor használják, ha egy megfeleltetés **kötelező** egy adott környezetben (v.ö. \Leftarrow operátor) és **semmilyen más környezetben nem** fordul elő (v.ö. \Rightarrow operátor)
- ☐ Ekvivalens a bikondicionális logikai operátorral és azt jelenti, hogy egy megfeleltetés **akkor és csak akkor** megengedett, ha az az adott környezetben van

"Soha"

L:S \nLeftarrow E

- ☐ Az \nLeftarrow operátort tipikusan egy általános szabály alóli kivételek kezelésére használják"
- ☐ Az \nLeftarrow operátort gyakran így írják: / \Leftarrow
- ☐ Mit jelent?
 - L soha nem realizálódik S-ként E-ben
 - L S-ként való realizációja nem megengedett E-ben
 - Ha L E-ben van, akkor L:-S-nek kell fennállnia
- ☐ Példa: **t:c** \nLeftarrow **___ i:ê** (a lexikális **t** nem felelhet meg a felszíni **c**-nek **i:ê** előtt)
- ☐ A szabály által megfogalmazott megfeleltetés az adott környezetben le van tiltva
- ☐ Megengedi tehát a **tatê** és **catê** felszíni alakokat, de tiltja a **tacê** és **cacê** alakot
- ☐ A \nLeftarrow operátor hasonló az \Leftarrow operátorhoz abban, hogy nem tiltja meg az adott megfeleltetést más környezetekben

Konvenciók, speciális szimbólumok

- ❑ **Alapértelmezett** (pl.. **t:t**, **i:i** – röviden: **t**, **i**) és **speciális megfeleltetések** (pl. **t:c**)
- ❑ **Dzsókerszimbólum** (általában: **@**) az adott ábécé tetszőleges karaktere helyett állhat, pl. **t:c** \Rightarrow **__ i:@**
- ❑ **Nullszimbólum** (általában: **0** vagy **ε**): beszúrásnál és törlésnél a kétszintes rendszer egyik szalagján üres szimbólumnak kell állnia, mert a két szalag csak egyenlő számú szimbólum esetén használható megfeleltetésre, pl.

LR: **0 t a t + i**

SR: **' t a c 0 i**

- ❑ **Határszimbólum** (általában: **#**) jelzi a szó elejét vagy végét (kizárólag egy másik határszimbólummal párban: **##**)
- ❑ **Részhalmaz:** egyszavas nevekkel jelzett karakterhalmazok, pl. **C** a mássalhangzók halmaza, **V** a magánhangzóké, **T** a felpattanó hangzóké, vagy **NAS** a nazálisoké:

SUBSET C

b c d f g p t k b d g m n n g s l r w y

SUBSET V

i e a o u

SUBSET T

p t k b d g

SUBSET NAS

m n n g

Kétszintes lexikonok

- ☐ **Kétszintes lexikonok:**
a **tőlexikonok**,
az **alternációs minták lexikonjai** és
a **toldaléklexikonok** (inflexiószak és derivációszak)
- ☐ **Lexikon:** egy név és az alábbi formájú lexikális elemek listája
- ☐ **Lexikális elemek:** lexéma, folytatási osztály, kimeneti információ
- ☐ A **P folytatási osztály** definíciója (ami valahol másol van megadva):
($P = PS \ K0 \ \#$)
azt állítja, hogy az eredeti lexikális elemet vagy a **PS** allexikon
(birtokos toldalékok), vagy a **K0** allexikon (klitikumok) valamelyik
eleme követheti, vagy egy határszimbólum

Az angol morfológia kétszintes leírása

(Karttunen & Wittenburg 1985)

Alternációk

ALTERNATIONS

```
( /N = N )  
( /IN = C1 )  
( /MN = MN )  
( /V = P3 PS PP PR I AG AB )  
( /IV1 = PR I AG AB )  
( /IV2 = P3 PR I AG AB )  
( /A = PA CA CS Ly )  
( /IP3 = IP3 )  
( /IPS = IPS )  
( /IPF = IPP )  
( C1 = C1 )  
( C2 = C2 )  
( I = I )  
( P3 = P3 )  
( PS = PS )  
( PP = PP )  
( PR = PR )  
( AG = AG )  
( PA = PA )  
( CA = CA )  
( CS = CS )  
( Ly = Ly )  
( Root = Root )  
( AB = AB )  
( # = )
```

END

Az angol morfológia kétszintes leírása

(Karttunen & Wittenburg 1985)

Lexiconok 1.

LEXICON N	0	C1	"N SG";	+s	C2	"N PL"
LEXICON MN	0	C2	"MASS N"			
LEXICON C1	0	#	"";	's	#	" GEN"
LEXICON C2	0	#	"";	'	#	" GEN"
LEXICON P3	+s	#	"V PRES SG 3RD"			
LEXICON IP3	0	#	"V PRES SG 3RD"			
LEXICON PS	+ed	#	"V PAST"			
LEXICON IPS	0	#	"V PAST"			
LEXICON PP	+ed	#	"V PAST PRT"			
LEXICON IPP	0	#	"V PAST PRT"			
LEXICON IP	0	#	"V PAST"			
LEXICON PR	+ing	#	"V PROG"			
LEXICON I	0	#	"V"			
LEXICON IP1	0	#	"V PRES SING 1ST"			
LEXICON AG	+er	/N	"AG "			
LEXICON PA	0	#	"A"			
LEXICON CA	+er	#	"A COMP"			
LEXICON CS	+est	#	"A SUP"			
LEXICON Ly	ly	#	"ADV"			
LEXICON AB	+able	#	"VERB ABL"			

Az angol morfológia kétszintes leírása

(Karttunen & Wittenburg 1985)

Lexiconok 2.

LEXICON Root

am	#	"V PRES SING 1ST";
am	#	"AUX";
are	#	"AUX";
are	#	"V PRES PL";
are	#	"V PRES SING 2ND";
at	#	"PREP";
at`tack	/N	"";
at`tack	/V	"";
a`gree	/V	"";
be	#	"AUX";
be	/IV1	"";
beer	/N	"";
believe	/V	"";
big	/A	"";

... ..

under`stand	/IV2	"";
under`stood	/IPP	"";
under`stood	/IPS	"";
undid	/IPS	"";
undo	/IV1	"";
undoes	/IP3	"";
undone	/IPP	"";
untie	/V	"";
went	/IPS	"";
went	/IPS	"";

END

2018. október 3.

A 'kanPat' példa kétszintes szabályokkal

$$N \rightarrow m / _p; \text{elsewhere } n.$$

$$p \rightarrow m / m_$$

Alphabet

a b c d e f g h i j k l m N:m N:n n o p q r s t u v x y w z ;

Rules

"N realized as m"

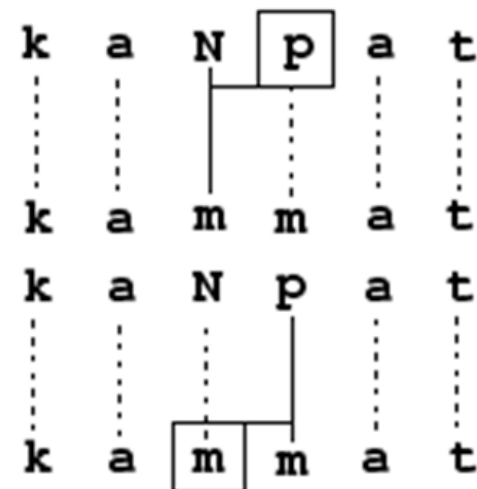
! Before a lexical "p." Always, and only there.

$N:m \Leftrightarrow _p;$

"p realized as m"

! After a surface "m". Always and only there.

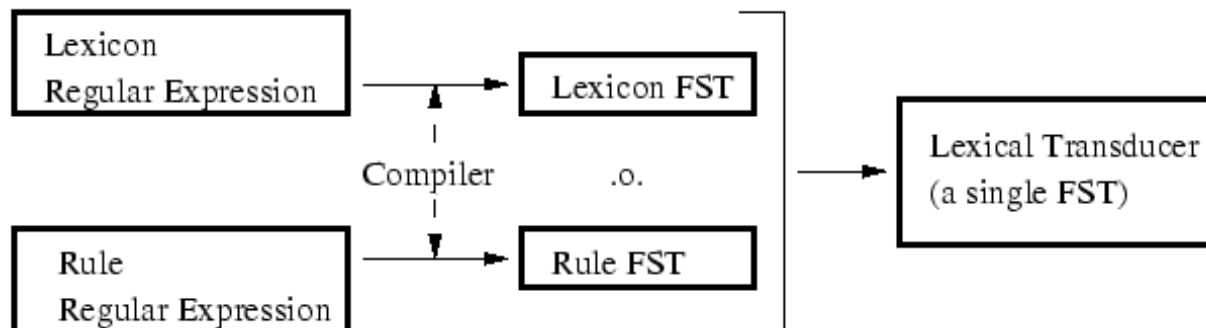
$p:m \Leftrightarrow :m _;$



Figures from <http://www.ling.helsinki.fi/~koskenni/esslli-2001-karttunen/>

A Xerox lexikonkompozíciója

- ❑ Ha elvégezzük a kompozíció műveletét a kétszintes lexikon és a kétszintes szabályhalmaz átalakítóin, a lexikon által nem megengedett füzérek automatikusan kiesnek
- ❑ Eleinte a nagyméretű lexikonok és a nagyméretű szabályrendszer kompozíciója eredményétől tartottak a kutatók, hogy az használhatatlanul nagy méretű lesz
- ❑ Lauri Karttunen és más Xerox-kutatók kimutatták, hogy a lexikon és a kétszintes rendszer FST-inek kompozíciója soha **nem lesz szignifikánsan nagyobb** a kiinduló lexikon FST-jénél, és **sokkal kisebb** lesz, mint a szabályok véges átalakítóinak metszeteként kapott FST
- ❑ A kapott egyetlen lexikális FST a kiinduló lexikon minden lexikális alakját és ezeknek a szabályok által realizált összes felszíni reprezentációját tartalmazza:



Lexikon + lexikális reprezentáció + felszíni alak

Lexikális szint

	f	o	x	+N	+PL			
--	----------	----------	----------	-----------	------------	--	--	--



Köztes szint

	f	o	x	^	s			
--	----------	----------	----------	----------	----------	--	--	--



Felszíni szint

	f	o	x	e	s			
--	----------	----------	----------	----------	----------	--	--	--

Hogy konvertálunk kétszintes szabályokat FST-be?

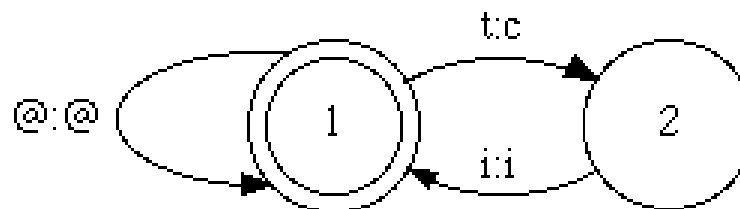
- ☐ A felhasználó környezetfüggő szabályokat használhat
- ☐ Minden nyelvi jelenséget egy önálló kétszintes szabály segítségével fogalmazunk meg (a többit maga a kétszintes rendszer kezeli)
- ☐ A környezetfüggőnek tűnő kétszintes szabályrendszert egyetlen FST-be lehet fordítani
- ☐ Ez a fordítás a TWOL-rendszer létrehozásától (1983) évekig csak kézzel történt
- ☐ A kézi szabályfordítás az átalakítók részletes ismeretét és az újszerű szabályok szemantikájának mély megértését követelte, amit nem sok kutató tudott az elvárt szinten elsajátítani, hiszen a sokszor egymással komplex interakcióba lépő szabályok működésének megértése sok-sok órás koncentrált munkát követelt mind a létrehozáskor, mind a teszteléskor
- ☐ Az első automatikus szabályfordítót Koskenniemi and Karttunen (1987) hozta létre, Ron Kaplan (Xerox) véges-állapotú kalkulusa első implementációjának segítségével, aminek alapját Kaplan és Kay (1994) véges állapotú nyelvészeti leírása adta

Példa: $a \Rightarrow$ konverziója FST-be

A kétszintes szabály:

$t:c \Rightarrow __ i$

Az ekvivalens FSA:



Az ekvivalens FSA táblázatos alakban:

	t	i	@
c	i	@	
1:	2	1	1
2:	0	1	0

Példák kétszintes szabályok táblázatos alakjára

$$t:c \leftarrow ___ i$$

	t	t	i	@
	c	@	i	@

1:	1	2	1	1
2:	1	2	0	1

$$t:c \Leftrightarrow ___ i$$

	t	t	i	@
	c	@	i	@

1:	3	2	1	1
2:	3	2	0	1
3:	0	0	1	0

$$t:c \nleftarrow ___ i:\hat{e}$$

	t	i	@
	c	\hat{e}	@

1:	2	1	1
2:	2	0	1

Kétszintes definíciók és szabályok

```
ALPHABET a b c d e f g h i j k l m n o p q r s t u v w x y z  
+
```

```
; + is morpheme boundary
```

```
NULL 0
```

```
ANY @
```

```
BOUNDARY #
```

```
SUBSET C b c d f g h j k l m n p q r s t v w x y z
```

```
SUBSET V a e i o u
```

```
; more subsets
```

```
RULE „Defaults” 1 29
```

```
    a b c d e f g h i j k l m n o p q r s t u v w x y z + @
```

```
    a b c d e f g h i j k l m n o p q r s t u v w x y z 0 @
```

```
1: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
RULE "Voicing s:z <=> V____V" 4 4
```

```
    V s s @
```

```
    V z @ @
```

```
1: 2 0 1 1
```

```
2: 2 4 3 1
```

```
3: 0 0 1 1
```

```
4: 2 0 0 0
```

```
; more rules
```

Az angol morfológia kétszintes leírása

(Karttunen & Wittenburg 1985)

Automata 1.

ALPHABET

a b c d e f g h i j k l m n o p q r s t u v w x y z '

+

NULL. 0

ANY =

SUBSET V a e i o u

SUBSET C b c d f g h j k l m n p q r s t v w x z

SUBSET S s x z

END

"Surface Characters " 1 28

a b c d e f g h i j k l m n o p q r s t u v w x y z ' "

a b c d e f g h i j k l m n o p q r s t u v w x y z ' =

1: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

"Epenthesis" 68

c h s S y + + =

ch s S i e 0 =

1: 2 1 4 3 3 0 1 1

2: 2 3 3 3 3 0 1 1

3: 2 1 3 3 3 5 6 1

4: 2 3 3 3 3 5 6 1

5. 0 0 1 0 0 0 0 0

6: 1 1 0 1 1 1 1 1

"Gemination"

16 26

```
V b d f g l m n p r s t+++++` =
```

V b d f e l m n p r s t b d f g l m n p r s t 0 0 =

1: 4 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1

[illegible]

```
3: 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0
```

4: 16 5 6 7 8 9 10 11 12 13 14 15 0 0 0 0 0 0 0 0 0 0 0 0 0 1 116

5: 16 16 16 16 16 16 16 16 16 16 16 16 16 2 0 0 0 0 0 0 0 0 0 0 3 1 1

```
6: 16 16 16 16 16 16 16 16 16 16 16 16 16 0 2 0 0 0 0 0 0 0 0 0 3 1 1
```

```
7: 16 16 16 16 16 16 16 16 16 16 16 16 16 0 0 2 0 0 0 0 0 0 0 0 3 1 1
```

```
8: 16 16 16 16 16 16 16 16 16 16 16 16 16 0 0 0 2 0 0 0 0 0 0 0 3 1 1
```

```
9: 16 16 16 16 16 16 16 16 16 16 16 16 16 0 0 0 0 2 0 0 0 0 0 0 3 1 1
```

10: 16 16 16 16 16 16 16 16 16 16 16 16 16 0 0 0 0 0 2 0 0 0 0 0 3 1 1

```
11: 16 16 16 16 16 16 16 16 16 16 16 16 16 0 0 0 0 0 0 2 0 0 0 0 3 1 1
```

```
12: 16 16 16 16 16 16 16 16 16 16 16 16 16 0 0 0 0 0 0 0 2 0 0 0 3 1 1
```

```
13: 16 16 16 16 16 16 16 16 16 16 16 16 16 0 0 0 0 0 0 0 0 2 0 0 3 1 1
```

```
14: 16 16 16 16 16 16 16 16 16 16 16 16 16 0 0 0 0 0 0 0 0 2 0 3 1 1
```

```
15: 16 16 16 16 16 16 16 16 16 16 16 16 16 0 0 0 0 0 0 0 0 0 0 2 3 1 1
```

```
16: 16 16 16 16 16 16 16 16 16 16 16 16 0 0 0 0 0 0 0 0 0 0 0 0 16 16
```



Az angol morfológia kétszintes leírása

(Karttunen & Wittenburg 1985)

Automata 2.

"Y-Spelling" 6 7

C y y + i a =

C y i = i a =

1: 2 1 0 1 1 1 1
2: 2 5 3 1 1 1 1
3: 0 0 0 4 0 0 0
4: 1 1 0 1 0 0 1
5: 1 1 0 6 1 1 1
6: 0 0 0 0 1 1 0

"Elision" 15 8

V i e e + g c =

V i e 0 0 g c =

1: 2 2 3 4 1 11 11 1
2: 1 1 5 6 1 11 11 1
3: 1 1 5 6 9 11 11 1
4: 0 0 0 0 8 0 0 0
5: 1 1 1 4 7 11 11 1
6: 0 0 0 0 10 0 0 0
7: 1 1 0 0 0 1 1 1
8: 1 1 1 0 0 0 0 0
9: 0 0 0 0 0 1 1 1
10: 0 0 1 0 0 0 0 0
11: 1 1 14 12 1 1 1 1
12: 0 0 0 0 13 0 0 0
13: 0 1 1 0 0 0 0 0
14: 1 1 1 0 15 11 11 1
15: 1 0 0 0 1 11 11 1

"I-Spelling" 7 6

i e + i e =

y 0 0 i e =

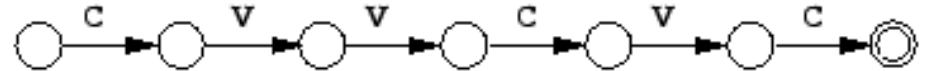
1: 2 1 1 5 1 1
2: 0 3 0 0 0 0
3: 0 0 4 0 0 0
4: 0 0 0 1 0 0
5: 1 1 1 1 6 1
6: 0 1 7 0 0 1
7: 0 0 0 0 0 1

END

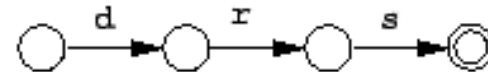
Nemkonkatenatív morfológiák FST-ben

- ❑ **Az összefésülő algoritmus:** egy olyan mintakitöltő művelet, mely két reguláris nyelvet kombinál: a mintát (template) és a kitöltőt (filler) egyetlen reguláris alakká
- ❑ A minta kezdőállapotából kiindulva az algoritmus megpróbálja megtalálni az összes megfelelő illesztést a mintaélek és a kitöltőélek között
- ❑ Az illesztés akkor sikeres, ha a kitöltőél címkéje benne van abban az osztályban, amit a mintaél címkéje határoz meg ($d \in C, r \in C, s \in C, u \in V, i \in V$)

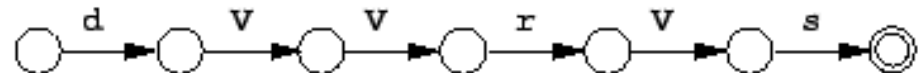
Minta:



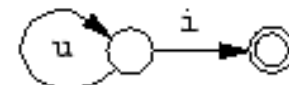
Mássalhangzókitöltő:



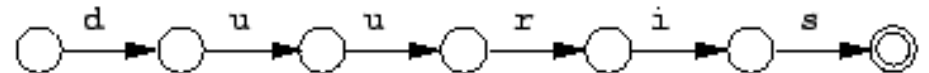
Köztes eredmény:



Magánhangzókitöltő:



Végeredmény:



A kétszintes leírás néhány nehézsége

- ☐ A felszín és a lexikális alak kötelezően azonos hosszúsága
- ☐ A szuppletív (lexikalizálódott) alakok és a nem produktív toldalékolás kezelése
- ☐ Ha a szótár „mindent kibír”, miért kell a „nehéz” alakokat is levezetésekkel kezelni?
- ☐ Pl. *jöv+ök, jösz+sz, jön+0, jöt+tök, jö+het, ...*
- ☐ *..., jó, ..., gyere, gyerünk, gyertek*
- ☐ Mit ér a reguláris rendszer, ha vannak reguláris nyelvvel nem leírható morfológiai jelenségek?
- ☐ Pl. *nagy, nagy+obb, leg+nagy+obb*