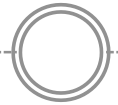
A nyelvtechnológia alapjai



5. Gyakorlat

Kétszintes morfológia

Generatív nyelvtan

- Chomsky (1957): generatív nyelvtan
  - A formális nyelveket jól tudjuk kezelni: modellezzük a természetes nyelvet valamilyen formális nyelvvel.
- próbáljuk meg reguláris nyelvvel, mert az a legkényelmesebb
- bizonyos szintaktikai jelenségekre nem elég – vita azóta is
  - $\rightarrow$környezetfüggő kell
- morfológia leírására viszont alkalmas!
- Példa:
  - kutya + t = kutyát
  - perec + vAl = pereccel

Generatív szabályok

- sorba rendezett „újraíró szabályok” = Chomsky-nyelvtan produkciós szabálya
  - újraírás: S$\rightarrow$abc alkalmazásakor S eltűnik
- szabályok sorrendje nyilván fontos
- a szabályalkalmazás után az elemzés újabb szintje jön létre

Kétszintes morfológia

- Kimmo Koskenniemi (1983): kétszintes morfológia
  - Lényege: párhuzamos generatív szabályokat egy szabállyá alakító algoritmus.
- reguláris (=véges állapotú) eszköz a morfológia leírására

Kétszintes szabályok

- nem sorrendben futnak, hanem párhuzamosan az automaták metszetét számolja ki
- szabályok sorrendje nem számít
  - egymástól függetlenül lehet leírni az egyes jelenségeket
- csak egy szabály $\rightarrow$ összesen két szint:
  - mögöttes – felszíni (lexical – surface)
  - hivatkozhatunk a mögöttes és a felszíni szimbólumokra
- nem újraírás: megfeleltetés
- a szabályok kétirányúak
  - generálás: mögöttes $\boxed{\leftrightarrow}$ felszíni: elemzés

PC-KIMMO

- Szabályformátum
  - példa: latin kiejtés
  - lexikon: leírt alak (pl. ratio), felszín: kiejtés (racio)
  - Forma:
    - t:c <=> _ @:i
  - szimbólum: mögöttes:felszíni
  - x:x rövidítése: x
  - a szimbólum helye a környezetben: _
  - tetszőleges elem: @

PC-KIMMO

- Szabályformátum
  - üres elem: 0 → beszúrás: 0:x, törlés: x:0
  - Példa: y:0 <= g _ g y
- operátorok: =>, <=, <=>, /<=

a:b <=> l _ r	=> és <= egyszerre
a:b => l _ r	ha a:b megjelenik, akkor mindenképp az adott környezetben kell megjelennie
a:b <= l _ r	ha a mögöttes a az adott környezetben jelenik meg, akkor mindenképp b-ként kell realizálódnia
a:b /<= l _ r	ha a mögöttes a az adott környezetben jelenik meg, akkor sosem realizálódhat b-ként

t:c <=> _ @:i

	t	t	@	@
	c	@	i	@
1:	2	3	1	1
2.	0	0	1	0
3:	2	3	0	1

- az automata éleit szimbólumpárokkal címkézzük
- kezdőállapot: az első
- elfogadó állapot: a kettőspontos

PC-KIMMO

- Fájlok:
  - szabályok: .rul
  - lexikon: .lex
  - „take” fájlok: .tak – szkript
  - generálás fájlok: .gen
  - felismerés fájlok: .rec
- Parancsok:
  - l(oad) r(ules) <fájl>
  - l(oad) l(exicon) <fájl> – ebben a sorrendben!
  - t(ake) <fájl>
  - g(enerate) – üzemmód
  - r(ecognize) – üzemmód
  - co(mpare) g(enerate)/r(ecognize)

Xerox Finite-State Tool

- Xerox Finite-State Tool
  - Xerox Palo Alto Research Center(PARC)
  - Xerox Research Centre Europe(XRCE)
- Tools
  - xfst – interface to Finite-State Calculus algorithms
  - lexc – lexicon compiler
  - twolc – two-level rule compiler

XFST

- Stack – Last-in, first-out (LIFO)
  - push, pop
  - a legfelső automatát használja
- Automaták – reguláris kifejezések
  - define *változó reguláris_kifejezés* ;
  - read regex *reguláris_kifejezés* ;
- Felszíni szint - elemzés
  - apply up
- Mögöttes szint - generálás
  - apply down

XFST

- Skript fájl
  - Betöltés: source script.xfst
  - define var [...];
  - regex [var];
- Környezetfüggő szabályok
  - [A -> B || L _ R]
    - A-ból B lesz, ha A előtt L és A után R áll

XFST

- Fontosabb reguláris kifejezése és operátorok
  - 0 – üres sztring (epsilon)
  - ? – bármilyen sztring (any)
  - .#. – sztring eleje vagy vége
  - () – opcionális
  - + – iteráció: 1 vagy többször
  - * – iteráció: 0 vagy többször
  - | – únio
  - .o. – kompozíció

Köszönöm a figyelmet

ध्यान देने के लिए आपका धन्यवाद

आप अपना ध्यान के लिए धन्यवाद

थैंक यू फॉर योर अटेंशन

Thank you for your attention

Thank you for your attention

Thank you for the attention

Merci de votre attention

Je vous remercie pour votre attention

Merci pour l'attention

感谢您的关注

谢谢你的关注

谢谢您的注意