Loomuliku keele analüüs ja masintõlkimine

Loogiline programmeerimine ITI0211

Sügis 2020

J.Vain

Ajalooline taust

- Loomuliku keele analüüs oli üks esimesi Prologi rakendusi.
- Keele süntaktiline analüüs (*parsing*) seisneb lause grammatilise struktuuri selgitamises ja selle vastavuse kontrollimises grammatika reeglitele.
- Keele süntees on sõnastiku ja grammatika reeglite abil **süntaktiliselt** õigete lausete genereerimine.
- Grammatikate formaalse defineerimise süsteemid:
  - Produktsioonid (asendusreeglid): *A*, *B*,… ← *d*, *E*, *g*,*F*,…
  - Backus-Nauri kuju (BNF)
  - Definite Clause Grammar (DCG) (vt SWI-Prolog Help)
- jt

Näide: IBM Watsoni loomuliku keele protsessor Prologis

- *IBM Watson* on tehisintellekti-põhine ärirakenduste platvorm
  - https://www.ibm.com/watson/about/
  - https://www.youtube.com/watch?v=_Xcmh1LQB9I
- Kasutati televiktoriinis Jeopardy! Man vs. Machine

  Küsimus: `"POETS & POETRY: He was a bank clerk in the Yukon before he published “Songs of a Sourdough” in 1907, name the person.`

Lausendi interpreteerimise sammud Watsonis:

1. Grammatika parser eraldas välja põhilised lauseliikmed ja nende täpsustused (omadused)
2. Küsimuse analüüs tuvastas küsitava kategooria “`he`“
3. Leksika analüüs tuvastas konteksti "poets"

Näide: *IBM Watson* dialoogisüsteem autori tuvastuseks

Prolog faktid esitavad grammatika puu nummerdatud tippe:

```
lemma(1, "he").
partOfSpeech(1,pronoun).
lemma(2, "publish").
partOfSpeech(2,verb).
lemma(3, "Songs of a Sourdough").
partOfSpeech(3,noun).
subject(2,1).
object(2,3).
```

Reegel laulu autori leidmiseks (lihtsustatud kujul):

```
authorOf(Author,Composition) :-
    createVerb(Verb),
    subject(Verb,Author),
    author(Author),
    object(Verb,Composition),
    composition(Composition).

createVerb(Verb) :-
    partOfSpeech(Verb,verb),
    lemma(Verb,VerbLemma),member(VerbLemma, ["write", "publish",...]).
```

Grammatika formaalne esitus

- *Grammatika G on nelik G* = (*N*, Σ, *P*, *S*)
  N – mitteterminalid
  Σ - terminalid
  P - produktsioonireeglid
  S – algussümbol

Näide:
Olgu
N = {*S*, *B*},
Σ = {*a*, *b*, *c*},
P :

1) $S \rightarrow aBSc$
2) $S \rightarrow abc$
3) $Ba \rightarrow aB$
4) $Bb \rightarrow bb$

Grammatika G genereerib keele L(*G*):

- $\boldsymbol{S} \Rightarrow_2 \boldsymbol{abc}$
- $\boldsymbol{S} \Rightarrow_1 \boldsymbol{aBSc} \Rightarrow_2 aB\boldsymbol{abc}c \Rightarrow_3 a\boldsymbol{a}B\boldsymbol{b}cc \Rightarrow_4 aa\boldsymbol{bb}cc$
- $\boldsymbol{S} \Rightarrow_1 \boldsymbol{aBSc} \Rightarrow_1 aB\boldsymbol{aBSc}c \Rightarrow_2 aBaB\boldsymbol{abc}cc \Rightarrow_3 a\boldsymbol{a}B\boldsymbol{B}abccc \Rightarrow_3 aaB\boldsymbol{a}\boldsymbol{B}bccc \Rightarrow_3 aa\boldsymbol{a}\boldsymbol{B}Bbccc \Rightarrow_4 aaaB\boldsymbol{bb}ccc \Rightarrow_4 aaa\boldsymbol{bb}bccc$

Generatiivne grammatika

Generatiivse grammatika puhul genereeritakse laused mitteterminalsümbolite asendamise teel kasutades asendusreegleid.

Näide (jätk): Olgu $N = \{S, B\}$, $\Sigma = \{a, b, c\}$,
P : 1) $S \rightarrow aBSc$ 2) $S \rightarrow abc$ 3) $Ba \rightarrow aB$ 4) $Bb \rightarrow bb$

Grammatika G genereerib keele L(G):

- $\boldsymbol{S} \Rightarrow_2 \boldsymbol{abc}$
- $\boldsymbol{S} \Rightarrow_1 \boldsymbol{aBSc} \Rightarrow_2 aB\boldsymbol{abc}c \Rightarrow_3 a\boldsymbol{aB}bcc \Rightarrow_4 aa\boldsymbol{bb}cc$
- $\boldsymbol{S} \Rightarrow_1 \boldsymbol{aBSc} \Rightarrow_1 aB\boldsymbol{aBSc}c \Rightarrow_2 aBaB\boldsymbol{abc}cc \Rightarrow_3 a\boldsymbol{aB}Babccc \Rightarrow_3 aaB\boldsymbol{aB}bccc \Rightarrow_3 aa\boldsymbol{aB}Bbccc \Rightarrow_4 aaaB\boldsymbol{bb}ccc \Rightarrow_4 aaa\boldsymbol{bb}bccc$

- Ühe mitteterminalsümboli asendamiseks võib olla grammatikas mitu alternatiivset reeglit (näites reeglid 1) ja 2) sümboli *S* asendamiseks)
- Keele terminalsõnade moodustumine grammatika puu harus sõltub seega reeglite rakendamise järjekorrast.
- Terminalsõnale asendusi rakendada ei saa.

Asendusreeglite esitusi

- Näide 1: Asendusreegel *vasaklineaarses* grammatikas:

mitteterminal sümbol $\rightarrow$ terminalsümbol, mitteterminalsümbol$_1$, ... , mitteterminalsümbol$_n$

- Näide 2: DCG-reeglid Prologis (DCG – Definite Clause Grammar) ei pruugi olla vasaklineaarsed

reegli_päis --> reegli_keha

- Näide (DCG Prologis):

```
lause --> nimisonafraas, tegusonafraas.
```

Võib mõist ka kui „is_a“ relatsiooni üldistust, kus asendatav mitteterminal tähistab ülemlassi ja alamklasside kompositsioonid on määratud “,“ - järjestusega

Näide: terminal- ja mitteterminalsümbolid

- DCG Reegel Prologis:

```
nimisona --> [poiss];[hobune];[tüdruk]
```

Mitteterminalsümbol

Terminalsümbolid

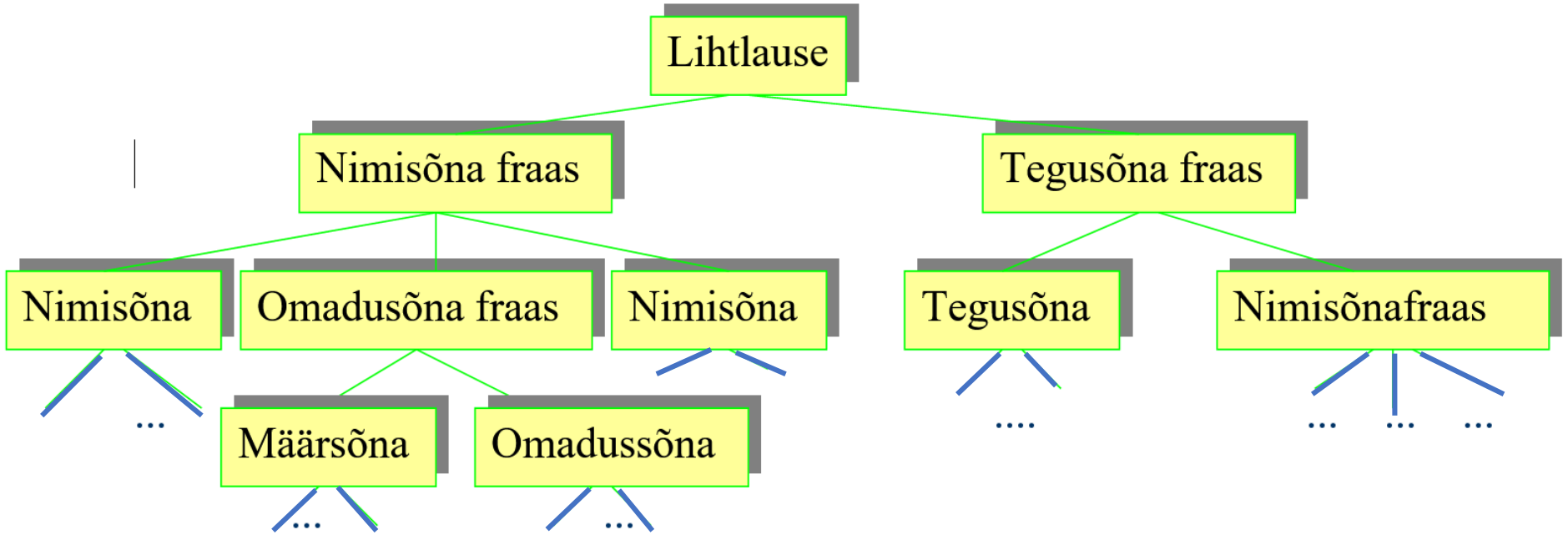
Siin on lihtne koosnevussuhe - terminalid on seotud läbi “;“

- Reegli tähendus:

  Sõnad "poiss", "tüdruk" ja "hobune" on nimisõnad.

- Kitsendused reeglite süntaksile Prologis:
  - terminalsümboliks võib olla mistahes Prologi term;
  - mitteterminalsümbol esitatakse DCG-s termiga, milleks ei tohi olla list

Näide (eesti keele lihtlause fraasistruktuuri grammatika)



Näide

- Olgu lause

  „Päkapiku liiga napp habe tingib tema sobimatuse jõuluvanaks.“

- Defineerime selle lause parsimiseks DCG grammatika.

Fraasistruktuuri grammatika esitus DCG-s

```
lihtlause --> nimisonafraas, tegusonafraas.
nimisonafraas --> nimisona, omadussonafraas, nimisona.
nimisonafraas --> nimisona, nimisonafraas ; [].
nimisona --> [pakapiku];[habe];[tema];[sobimatuse];[jouluvanaks].
% terminalsümbolid esinevad reeglis paremal pool ühiklistidena
omadussonafraas --> maarsona, omadussona.
maarsona --> [liiga].
omadussona --> [lyhike].
tegusonafraas --> tegusona, nimisonafraas.
tegusona --> [tingib] ; [pohjustab].
```

Juhul, kui antud lauseosa võib puududa

Näide (järg)

- DCG parsimispäring PROLOGis:

```
?- phrase(lihtlause,[pakapiku,liiga,napp,habe,tingib,tema,sobimatuse,jouluvanaks]).
```

  - *phrase* on süsteemi predikaat
  - esimene argument `lihtlause` on grammatika algsümbol
  - teine argument on algsümbolist DCG reeglitega tuletatava grammatika interpreteeritav/genereeritav fraas.

```
% vt. swi-prolog samples DCGXPAND
```

DCG-reegli teisendamine Horni lauseks

```
lihtlause(A, B) :-
        nimisonafraas(A, C),
        tegusonafraas(C, B).

nimisonafraas(A, B) :-
        nimisona(A, C),
        omadussonafraas(C, D),
        nimisona(D, B).
nimisonafraas(A, B) :-
        (   nimisona(A, C),
            nimisonafraas(C, B)
        ;   A=B         ).
```

Parsimine DCG-reeglitega on Prologi programmi täitimine

- Parsimisel genereerib Prolog *DCG* reeglite põhjal programmi, kus parameetrid annavad edasi interpreteerimist vajava fraasi osa.

```
lihtlause(A, B) :-
        nimisonafraas(A, C),
        tegusonafraas(C, B).

nimisonafraas(A, B) :-
        nimisona(A, C),
        omadussonafraas(C, D),
        nimisona(D, B).
nimisonafraas(A, B) :-
        (   nimisona(A, C),
            nimisonafraas(C, B)
        ;   A=B             ).
```

Nooled näitavad parameetrites edasiantavaid fraasi osi

DCG-reeglid Prologi programmina

```
tegusonafraas(A, B) :-
        tegusona(A, C),
        nimisonafraas(C, B).

nimisona([pakapiku| B], B) .
nimisona([habe| B], B) .
nimisona([tema| B], B) .
nimisona([sobimatuse| B], B) .
nimisona([jouluvanaks| B], B) .

omadussonafraas(A, B) :-
        maarsona(A, C),
        omadussona(C, B).

omadussona([napp|A], A).

maarsona([liiga|A], A).

tegusona([tingib|A], A).
```

Märkus:
- Reegli esituse muudab mugavaks Prologi listide pea ja saba eraldaja ”|”.
- Listi peas esitatakse reegli vasak pool ehk asendatav osa.

Näide inglisekeele fraasigrammatikast

```
sentence -->          nounphrase, verbphrase.
nounphrase -->        determiner, nounexpression.
nounphrase -->        nounexpression.
nounexpression -->    noun.
nounexpression -->    adjective, nounexpression.
verbphrase -->        verb, nounphrase.
determiner -->        [the] ; [a].
noun -->              [dog] ; [bone] ; [mouse] ; [cat].
verb -->              [ate] ; [chases].
adjective -->         [big] ; [brown] ; [lazy].
```

Alternatiivsed asendused samale mitteterminalile

DCG sisekujus kasutatakse erisuslisti (*difference list*) listide paar *X*-*Y*, kus listis *Y* on listi *X* sabaosa, mis ei unifitseerunud antud reeglis määratud termiga.

```
?- phrase(sentence,[dog,chases,cat]).

1-1 CALL phrase(sentence,[dog,chases,cat])
    2-1 CALL nounphrase([dog,chases,cat],_0)
        3-1 CALL determiner([dog,chases,cat],_0)
        3-1 FAIL determiner([dog,chases,cat],_0)
    2-1 REDO nounphrase([dog,chases,cat],_0)
        3-1 CALL nounexpression([dog,chases,cat], _0)
            4-1 CALL noun([dog,chases,cat],_0)
            4-1 EXIT noun([dog,chases,cat],[chases,cat])
        3-1 EXIT nounexpression([dog,chases,cat],[chases,cat])
    2-1 EXIT nounphrase([dog,chases,cat], [chases,cat])
    2-2 CALL verbphrase([chases,cat],[])
        3-1 CALL verb([chases,cat],_4)
        3-1 EXIT verb([chases,cat],[cat])
        3-2 CALL nounphrase([cat],[])
            4-1 CALL determiner([cat],[])
            4-1 FAIL determiner([cat],[])
        3-2 REDO nounphrase([cat],[])
            4-1 CALL nounexpression([cat],[])
                5-1 CALL noun([cat],[])
                5-1 EXIT noun([cat],[])
            4-1 EXIT nounexpression([cat],[])
        3-2 EXIT nounphrase([cat],[])
    2-2 EXIT verbphrase([chases,cat],[])
1-1  EXIT sentence([dog,chases,cat])
```

Keele lausete genereerimine

Kui päringu `phrase/2` ainuke väärtustatud parameeter on grammatika juursümbol, siis tagastab päring grammatika reeglite suhtes kõik korrektsed laused.

Näide:

```
?- phrase(↓lihtlause, ↑L).

L = [pakapiku, liiga, napp, pakapiku, tingib, pakapiku, liiga, napp, pakapiku] ;
L = [pakapiku, liiga, napp, pakapiku, tingib, pakapiku, liiga, napp, habe] ;

L = [pakapiku, liiga, napp, pakapiku, tingib, pakapiku, liiga, napp, tema] ;
L = [pakapiku, liiga, napp, pakapiku, tingib, pakapiku, liiga, napp, sobimatuse]
;
L = [pakapiku, liiga, napp, habe, tingib, tema, sobimatuse, jouluvanaks].
```

Masintõlkimine

- Lihtne sarnase grammatikaga, erinevate sõnastikega keelte vahel
- Tõlkeprogrammi üldstruktuur:

```
translate(Lähte_keele_lause):-
     find_unknowns(Lähte_keele_lause).     % kas on tundmatuid sõnu
translate(Lähte_keele_lause):-
     sentence(Lähte_keele_lause,[], Sihtkeelelause,[]),
     send_print(Sihtkeelelause).
translate(_):-
     write('Sorry, do\t recognize this type of sentence'),nl.
```

Tõlkeprogrammi üldstruktuur (järg)

Phrase_unit(↓ [Recognizable_unit|Rest], ↑Rest, ↓ Earlier_Translation, ↑ Expanded_Translation):-

dictionary(↓ Recognizable_unit, ↑ Unit_Transaltion),

append(↓ Earlier_Translation, ↓ Unit_Translation, ↑ Expanded_Translation).

Masintõlkimine

- Tõlge erineva grammatikaga (mitte ainult sõnastikega) keelte vahel nõuab lisaks konteksti tundlikku ja **semantilist analüüsi**.
- Näide: lihtsate sünonüümide tuvastus:

```
verb(vaata, [look|X],X).
verb(vaata, [look,around|X],X).
verb(lõpeta,[end|X],X).
verb(lõpeta,[quit|X],X).
verb(lõpeta,[close|X],X).
```

- Üldiselt eeldab tõlge konteksti tuvastust ja tõlget fraasi-tasandi reeglitega
- fraasi-tasandi reeglid võivad omada sihtkeele erinevate vastete jaoks mitmeid alternatiive ja erinevaid grammatilisi struktuure.

Fraasi tuvastuse sidumine reaktsiooniga

See on sisuliselt loomuliku keelec tõlge kontrolleri käskude ja tegevus-stsenaariumite keelde:

- Verbaalsele väljendile võib vastavusse seada reaktsiooni (parametriseeritud programmi), mis genereeritakse ja käivitatakse fraasi tuvastanud predikaadis.
- Täitva programmi saab genereerida genereeriva grammatikaga, mille konkreetsed reeglid väärtustatakse parsimisel saadud muutujate väärtustega.
- Näide:

```
? - cmd[ vii, tass, lauale, Y).

cmd(Command_i, Rest):-
    phrase(cmd_i_grammar,Command_i,Rest),
    reaction(action_grammar,Result),!.
    …
    verb --> [vii];[too];…. % kui käsus sisaldub „vii“, siis genereeritakse
                            tegevusgrammatika reegel, mis konkretiseerib “vii“
    verb(A,C):- vii(A,C), assert(action --> vii,…).
```

Lisalugemist

- http://www.learnprolognow.org/slides/official/LPNchapter7.pdf
- http://www.learnprolognow.org/slides/official/LPNchapter8.pdf