Relationale Grammatik
Prädikation und Komplementation

Nadine Perera

14. Dezember 2009

Agenda

- Implementierung von R7-R11
- Diskussionspunkte
- Vorgehen beim Regel-Implementieren
- Erklärung der Änderungen von Grammatik, Lexikon und Modell
- Beispiele
- Implementierung von Numerus und Kasus
- Kasus-Beispiele

Regeln laut Böttner-Buch

R7	$VP \rightarrow TV + PN$	$[VP] = [TV]:[PN]$
R8	$VP \rightarrow Cop + PP$	$[VP] = [PP]$
R9	$PP \rightarrow P + PN$	$[PP] = [P]:[PN]$
R10	$VP \rightarrow Cop + NP$	$[VP] = [NP]$
R11	$NP \rightarrow RN + von + PN$	$[NP] = [RN]:[PN]$

- Beachte: NP statt wie bei uns N'
  $\Rightarrow$ was passiert mit Artikeln?
- Unbestimmter Artikel im Existenzquantor abgebildet
- Bestimmter Artikel?
  $\Rightarrow$ Semantik für bestimmten Artikel nicht vorgesehen bei Böttner
  $\Rightarrow$ Genus-Unterscheidung sinnvoll?

Diskussionspunkte 1 - Det und Universen

- Vorschlag: Kategorie Det ohne Genus-Koordination und ohne Semantik implementieren, d.h. "Nimm das erste Element" statt "Es darf nur ein Element geben, überprüfe dies."
- Neue Regeln brauchen ein neues Universum mit Ortsnamen, z.B. "Maria ist in Madrid", "Madrid ist in Spanien", "Madrid ist die Hauptstadt von Spanien" etc.
  $\Rightarrow$ Überlegung: ein Universum für Personen und Orte oder jeweils eines für Personen und eines für Orte?
- Wie würden wir diese beiden Universen vereinigen (Funktor)?
- Für ein gemeinsames Universum entschieden.

Diskussionspunkte 2 - Numerus und Kasus

- Wann brauchen wir Numerus und Kasus von Nomen, z.B. Nsa (für Nomen Singular Akkusativ) oder Npd (für Nomen Plural dativ)?
- Unsere Nomen sind vom Typ CN ("Frau", "Hund"), RN ("Bruder", "Hauptstadt"), PN ("Anna", "Spanien")
- Numerus und Kasus bei PN sinnvoll? Maria, Marias, Maria, Maria?
- Außerdem: S $\rightarrow$ PN VP, d.h. Sätze haben nur PN als Subjekt, PNs sind bisher Singular
  $\Rightarrow$ Kasus und Numerus nur bei Objekt interessant

Vorgehen beim Regel-Implementieren

- Verändern:
  1. pg.glr (die Grammatik)
  2. pg.lex (das Lexikon)
  3. pg1.sml (das Modell)
- Nach jeder Änderung: Paket neu machen mit
  - `CM.make "pg1.cm";`
  - `open NL1;`
- Neue Sätze testen, z.B. mit
  - `ev "Maria ist in Madrid";`
  - `evals "Maria ist in Madrid";`

Grammatik: Durchgeführte Änderungen 1/2

```
<     | PN of set | CN of set | RN of rel
---
>     | PN of set | CN of set | RN of rel | P of rel

<     | ein | eines | EQ | NQ | UQ    (* Quantoren: ein, ...
---
>     | von | Det | EQ | NQ | UQ    (* Quantoren: ein, ...

<     | VP of set | IVP of set | TVP of set
---
>     | VP of set | IVP of set | TVP of set | PP of set
```

Grammatik: Durchgeführte Änderungen 2/2

```
<       | CopV ein N'     (N')                    (* R5 *)
---
>       | CopV EQ N'      (N')                    (* R5 *)
>       | CopV Det N'     (N')                    (* R10 *)
>       | CopV PP         (PP)                    (* R8 *)

> PP : P PN              (PreIm(P,PN))           (* R9 *)

< N' : CN               (CN)                    (* R6 *)
<       | RN eines N'     (PreIm(RN,N'))          (* Bruder ei...
---
> N' : CN               (CN)                    (* R6a *)
>       | PN              (PN)                    (* R6b *)
>       | RN von N'       (PreIm(RN,N'))          (* R11 *)
>       | RN EQ N'        (PreIm(RN,N'))          (* Bruder ei...
```

Lexikon: Durchgeführte Änderungen

```
< ("ein" | "eine")  => ([T.ein(!line,!line)],yytext);
< "eines"    => ([T.eines(!line,!line)],yytext);
< "einen"    => ([T.EQ(!line,!line)],yytext);
---
> "von"    => ([T.von(!line,!line)],yytext);
> ("der"|"die"|"das"|"den")=>([T.Det(!line,!line)],yytext);
> ("Ein" | "Eine" |"einen" | "ein" | "eine" | "eines")
>  => ([T.EQ(!line,!line)],yytext);

> "Rom"   => (PN "Rom");
> "Madrid"   => (PN "Madrid");
> "Italien"   => (PN "Italien");
> "Spanien"   => (PN "Spanien");
> "Hauptstadt"   => (RN "Hauptstadt");

> ("in") => (P "in");
```

Modell: Durchgeführte Änderungen

```
<     datatype element = a|b|c|d|e|f|g|h|m
<     val elements = [a,b,c,d,e,f,g,h,m]
---
>     datatype element = a|b|c|d|e|f|g|h|m|n|s|w|p
>     val elements = [a,b,c,d,e,f,g,h,m,n,w,s]

>       | ConS "Stadt"      = [w,n]
>       | ConS "Rom"       = [w]
>       | ConS "Madrid"    = [n]
>       | ConS "Spanien"   = [s]
>       | ConS "Italien"   = [p]

>     fun ConR "in"         = [(w,p),(n,s)]
>       | ConR "Hauptstadt"= [(n,s),(w,p)]
```

Beispiele 1/3

```
- ev ‘‘Madrid ist die Hauptstadt von Spanien’’;
Absyn: 0 < (Madrid . (Hauptstadt : Spanien))
val it =
  [nullary
     (E (ProdS (ConS ‘‘Madrid’’,PreIm (ConR ‘‘Hauptstadt’’,
      ConS ‘‘Spanien’’))))]
  : Absyn.absyn list

- ev ‘‘Madrid ist den Hauptstadt von Spanien’’;
Absyn: 0 < (Madrid . (Hauptstadt : Spanien))
val it =
  [nullary
     (E (ProdS (ConS ‘‘Madrid’’,PreIm (ConR ‘‘Hauptstadt’’,
      ConS ‘‘Spanien’’))))]
  : Absyn.absyn list
```

Beispiele 2/3

```
  - ev ‘‘Madrid ist eine Hauptstadt von Spanien’’;
Absyn: 0 < (Madrid . (Hauptstadt : Spanien))
val it =
  [nullary
     (E (ProdS (ConS ‘‘Madrid’’,PreIm (ConR ‘‘Hauptstadt’’,
      ConS ‘‘Spanien’’)))))]
  : Absyn.absyn list

- ev "Dieter ist der Bruder eines Studenten";
Absyn: 0 < (Dieter . (Bruder : Student))
val it =
  [nullary (E (ProdS (ConS "Dieter",PreIm (ConR "Bruder",
   ConS "Student")))))]
  : Absyn.absyn list
```

Beispiele 3/3

```
-ev "Dieter ist ein Bruder eines Studenten";
Absyn: 0 < (Dieter . (Bruder : Student))
val it =
  [nullary (E (ProdS (ConS "Dieter",PreIm (ConR "Bruder",
   ConS "Student"))))]
  : Absyn.absyn list

- ev "Dieter ist Bruder eines Studenten";
Error, line 1: No parse tree found.
```

Numerus/Kasus - Überlegungen

```
TVP : TV PN              (PreIm(TV,PN))            (* R7 *)
    | TV Det N'          (Exp(TV,N'))              (* ?? *)
    | TV UQ N'           (Exp(TV,N'))              (* R19 *)
    | TV EQ N'           (PreIm(TV,N'))            (* R20 *)
    | TV NQ N'           (CompS(PreIm(TV,N')))     (* R21 *)
```

Neue Regel ?? ermöglicht: "Dieter kennt den Bruder von Anna"
aber auch: "Dieter kennt den Anna"

Kasus interessant für RN und CN:

- "Dieter geht mit dem Hund spazieren"
- "Dieter kennt den Bruder von Anna"
- Kasus in Quantoren hineinziehen?

Numerus/Kasus - Implementierung

- Zunächst Numerus implementiert, Ordner PA-GLR-R11-Num
  - für Subjekt und Verb, d.h. VP $\rightsquigarrow$ VPs | VPp, IVP $\rightsquigarrow$ IVPs | IVPp, TVP $\rightsquigarrow$ TVPs | TVPp, IV $\rightsquigarrow$ IVs | IVp, CopV $\rightsquigarrow$ CopVs | CopVp, TV $\rightsquigarrow$ TVs | TVp
  - Bisher wird nur VPs genutzt wegen S $\rightarrow$ PN VP
- Dann Kasus implementiert, Ordner PA-GLR-R11-NumKas
  - N' $\rightsquigarrow$ N'n | N'g | N'd | N'a, RN $\rightsquigarrow$ RNn | RNg | RNd | RNa, CN $\rightsquigarrow$ CNn | CNg | CNd | CNa, Det $\rightsquigarrow$ Detn | Detg | Detd | Deta, EQ $\rightsquigarrow$ EQn | EQg | EQd | EQa, EQ $\rightsquigarrow$ NQn | NQg | NQd | NQa, UQ $\rightsquigarrow$ UQn | UQg | UQd | UQa
- Lexikon angepasst
- Semantik bisher unverändert!

Lexikonänderungen

Neue Funktionen sind zweistellig:

```
fun PN(s:string) = ([T.PN(A.ConS s,!line,!line)],s)
fun CNn s t = ([T.CNn(A.ConS s,!line,!line)],t)
fun CNg s t = ([T.CNg(A.ConS s,!line,!line)],t)
fun CNd s t = ([T.CNd(A.ConS s,!line,!line)],t)
fun CNa s t = ([T.CNa(A.ConS s,!line,!line)],t)
fun RNn s t = ([T.RNn(A.ConR s,!line,!line)],t)
fun RNg s t = ([T.RNg(A.ConR s,!line,!line)],t)
fun RNd s t = ([T.RNd(A.ConR s,!line,!line)],t)
fun RNa s t = ([T.RNa(A.ConR s,!line,!line)],t)
fun IVs s t = ([T.IVs(A.ConS s,!line,!line)],t)
fun IVp s t  = ([T.IVp(A.ConS s,!line,!line)],t)
fun TVs s t = ([T.TVs(A.ConR s,!line,!line)],t)
fun TVp s t = ([T.TVp(A.ConR s,!line,!line)],t)
fun CA(s:string) = ([T.CA(A.ConS s,!line,!line)],s)
fun P(s:string) = ([T.P(A.ConR s,!line,!line)],s)
```

Lexikonänderungen

Zweiter Text (“yytext”) wird für den Parsebaum benutzt, der erste Text (“Mann”) für die Semantik, damit die Mengen weiterhin einheitlich heißen.

```
"Mann"      => (CNn "Mann" yytext);
"Mannes"    => (CNg "Mann" yytext);
"Mann"      => (CNd "Mann" yytext);
"Mann"      => (CNa "Mann" yytext);

"arbeiten" => (IVp "arbeiten" yytext);
"arbeitet" => (IVs "arbeiten" yytext);
```

Kasus-Beispiele

```
- ev "Madrid ist den Hauptstadt von Spanien";
Error, line 1: No parse tree found.

- ev "Madrid ist die Hauptstadt von Spanien";
Absyn: 0 < (Madrid . (Hauptstadt : Spanien))
val it =  [nullary
     (E (ProdS (ConS "Madrid",PreIm
     (ConR "Hauptstadt",ConS "Spanien"))))]
  : Absyn.absyn list

- ev "Madrid ist der Hauptstadt von Spanien";
Absyn: 0 < (Madrid . (Hauptstadt : Spanien))
val it = [nullary
     (E (ProdS (ConS "Madrid",PreIm
     (ConR "Hauptstadt",ConS "Spanien"))))]
  : Absyn.absyn list
```

Anmerkungen

- Kein Genus implementiert, daher “Madrid ist der/die/das Hauptstadt von Spanien” zugelassen.
- Kasus sorgt für den richtigen Fall der Hauptstadt
- Numerus-Koordination zwischen Subjekt “Madrid” und Verb “ist” ist vorhanden, “Madrid sind die Hauptstadt von Spanien” würde auch dann nicht geparst werden, wenn die Satzstruktur VPp schon zulassen würde.

Bäume direkt anzeigen mit Graphviz (MacOS)

in der Datei

`SMLNJ-DIR/ml-glr/lib/chart.sml` folgenden Block

```
if progExists dot andalso progExists gv
then
 (OS.Process.system (dot^" -Tps "^filename^" -o "^psname),
OS.Process.system (gv^" "^psname^" &"))
else
  (TextIO.output(TextIO.stdErr,
    ("Program " ^ dot ^ " or " ^ gv ^ " is not installed;"
          ^ "displaying of trees is suppressed. "));
   (OS.Process.failure, OS.Process.failure))
```

ersetzen mit

```
(OS.Process.system ("echo"^" "),
OS.Process.system ("open"^" "^filename^" &"))
```

Zusammenfassung

- Regeln R7-R11 implementiert und erklärt
- Implementierung von Numerus und Kasus

- Vielen Dank für die Aufmerksamkeit!
- Fragen?