Einführung in die Grammatikentwicklung: Übung 7 (28. Juni 2020)

Ziele:

1. Hinzufügen semantischer Information zu Lexikoneinträgen und Regeln
2. Kennenlernen von relationalen Beschränkungen (*relational constraints*)

Übungen:

1. TRALE starten und Grammatik 7 laden: Klicken Sie auf das grüne T, unter dem Task 7 steht.
2. In diesem Schritt werden wir die Grammatik um semantische Information erweitern. Die grundlegende Operation für die Komposition semantischer Information ist die Listenverkettung. Semantische Relationen werden von Wörtern eingeführt und parallel zur syntaktischen Kombination von Wörtern (oder Wortgruppen) zusammengefügt, wenn größere Phrasen gebildet werden. Wir werden den Typ *relation* benutzen um Grundeinheiten der Semanik zu repräsentieren, die mit Wörtern verknüpft sind. Dieser Typ hat die Untertypen *arg1-relation*, *arg1-2-relation* und *arg1-2-3-relation* für Prädikate mit entsprechender Stelligkeit:

$$\begin{bmatrix} \textit{relation} \\ \text{PRED} \;\; \textit{string} \\ \text{ARG0} \;\; \textit{index} \end{bmatrix} \qquad \begin{bmatrix} \textit{arg1-relation} \\ \text{PRED} \;\; \textit{string} \\ \text{ARG0} \;\; \textit{index} \\ \text{ARG1} \;\; \textit{index} \end{bmatrix} \qquad \ldots \qquad \begin{bmatrix} \textit{arg1-2-3-relation} \\ \text{PRED} \;\; \textit{string} \\ \text{ARG0} \;\; \textit{index} \\ \text{ARG1} \;\; \textit{index} \\ \text{ARG2} \;\; \textit{index} \\ \text{ARG3} \;\; \textit{index} \end{bmatrix}$$

   (a) Fügen Sie die atomaren Typen *index*, *object* und *event* ein. Diese Typen entsprechen den Variablen für die Zuweisung semantischer Rollen.

   (b) Fügen Sie die oben beschriebenen Typen *relation*, *arg1-relation*, *arg1-2-relation* und *arg1-2-3-relation* als Untertypen von *feat-struc* ein.

   Der Typ *string* wird in Trale als `(a_ _)` geschrieben. Der zweite _ steht dabei für die Stelle, an der ein beliebiger String eingesetzt werden kann. Wenn Strings in Merkmalbeschreibungen verwendet werden, muss an den entsprechenden Stellen auch `(a_ _)` bzw. ein entsprechend spezifizierter Wert stehen: `pred:(a_ hund_rel)`

   Wir verwenden den String-Typ nur, weil wir zu faul sind, für jedes Wort einzeln in die Typhierarchie einzufügen.

   Der Bedeutungsbeitrag von Wörtern und Phrasen wird als Wert des beim Typ *syn-struc* neu einzuführenden Merkmals SEM repräsentiert. Der Wert von SEM ist eine Merkmalstruktur, die wie folgt aufgebaut ist:

$$\begin{bmatrix} \textit{semantics} \\ \text{INDEX} \;\; \textit{index} \\ \text{KEY} \;\; \textit{relation} \\ \text{RELS} \;\; \textit{list} \end{bmatrix}$$

   Das INDEX-Merkmal entspricht der externen Variable, die von anderen Ausdrücken gebunden werden kann. Der KEY-Wert zeigt auf die ausgewählte Relation, die für semantische Selektion benutzt wird (typischerweise kommt diese Information vom semantischen Kopf, siehe auch (Copestake, Flickinger, Pollard & Sag 2005)). Der RELS-Wert enthält eine Liste von Relationen (siehe unten). Für die Lexikoneinträge für *Hund-* und *jag-* nehmen wir die folgende Semantik an (als Wert des SEM-Merkmals):

$$\begin{bmatrix} \textit{semantics} \\ \text{INDEX } \boxed{1}\ \textit{object} \\ \text{KEY } \boxed{2} \begin{bmatrix} \textit{relation} \\ \text{PRED } \textit{“hund\_rel”} \\ \text{ARG0 } \boxed{1} \end{bmatrix} \\ \text{RELS } \langle\ \boxed{2}\ \rangle \end{bmatrix} \qquad \begin{bmatrix} \textit{semantics} \\ \text{INDEX } \boxed{4}\ \textit{event} \\ \text{KEY } \boxed{5} \begin{bmatrix} \textit{arg1-2-relation} \\ \text{PRED } \textit{“jagen\_rel”} \\ \text{ARG0 } \boxed{4} \\ \text{ARG1 } \textit{index} \\ \text{ARG2 } \textit{index} \end{bmatrix} \\ \text{RELS } \langle\ \boxed{5}\ \rangle \end{bmatrix}$$

(a) Führen Sie den Typ *semantics* ein, fügen Sie das Merkmal SEM bei der Typdefinition von *syn-struc* ein und beschränken Sie dessen Wert auf *semantics*.

(b) Definieren Sie ein Makro `relational_obj` für Objekte, die einen semantischen Beitrag leisten. Stellen Sie sicher, dass zum Ausdruck kommt, dass (i) lexikalische Elemente eine RELS-Liste mit genau einem Element haben, (ii) die KEY-Relation mit dem ersten (und einzigen) Element von RELS übereinstimmt und (iii) der INDEX das ARG0 des KEY ist.

Das Makro soll ein Argument haben (Pred), das dem Relationsname entspricht. Der Wert für *Hund* wäre `a_ hund_rel`.

(c) Alle Makros für lexikalische Elemente mit semantischem Beitrag (z. B. Verben, Nomina, bestimmte Präpositionen) müssen so geändert werden, dass sie das Makro `relational_obj` aufrufen.

Erweitern Sie die Makros *det*, *noun* und *verb* so, dass der semantische INDEX für Determinirer und Nomen vom Typ *object* und für Verben vom Typ *event* ist.

(d) Erweitern Sie die Makroaufrufe in den Lexikoneinträgen in lexicon.pl um ein Argument, das dem Relationsname entspricht.

(Komplementpräpositionen werden vorerst ignoriert.) Laden Sie die Grammatik neu und überprüfen Sie mit dem Menüpunkt ‘Trale|Show|Word’ den Lexikoneintrag für *Hunde* und *jagst*. Stellen Sie sicher, dass die Einträge so aussehen, wie es oben angegeben wurde.

Die Bedeutung einer Phrase wird als Verkettung der Listen, die die Bedeutung der Töchter enthalten, berechnet: Der RELS-Wert einer Phrase ist die Konkatenation der RELS-Werte der Töchter. Wir benutzen eine Liste für die Repräsentation der Bedeutung, die Reihenfolge der Listenelemente ist allerdings irrelevant: Wir benutzen eine Liste, um eine Multimenge (Engl. *multi-set* oder auch *bag*) zu repräsentieren.

(a) In allen Phrasen werden INDEX und KEY vom semantischen Kopf beigesteuert. In unserer Grammatik entspricht in allen Konstruktionen der semantische Kopf dem syntaktischen Kopf. Fügen Sie eine entsprechende Implikation für den Typ *headed-phrase* in der Datei syntax.pl ein.

(b) Für die Verkettung der RELS-Werte der Töchter verwenden wir eine relationale Beschränkung. Formal sieht das wie folgt aus:

```
binary_headed_phrase *>
  (sem:rels:append(HRels,NHRels),
   head_dtr:sem:rels:HRels,
   non_head_dtr:sem:rels:NHRels).
```

Der RELS-Wert der Kopftochter (HRels) und der RELS-Wert der Nicht-Kopftochter (NHRels) ist ein Argument der *append*-Relation. Diese liefert als Ergebnis die Verkettung der beiden Listen. Die Relation *append* ist in der Datei constraints.pl definiert und hat die folgende Form:

```
fun append(+,+,-).
append(X,Y,Z) if
   when( (X=(e_list;ne_list);
          Z=(e_list;ne_list))
       , undelayed_append(X,Y,Z)
       ).

undelayed_append([],L,L) if true.
undelayed_append([H|T1],L,[H|T2]) if append(T1,L,T2).
```

Das entspricht im Prinzip der Definition von *append*, die Sie wahrscheinlich von Prolog kennen:

```
append([],L,L).
append([H|T1],L,[H|T2]) :-
  append(T1,L,T2).
```

Das *append*, das wir verwenden, benutzt aber zusätzlich noch Delay-Anweisungen. Diese stehen im `when`-Teil. Nur wenn der Ausdruck

```
(X=(e_list;ne_list);
 Z=(e_list;ne_list))
```

erfüllt ist, d. h., wenn X vom Typ *e_list* oder *ne_list* ist, oder wenn das für Z gilt, dann wird *undelayed_append* aufgerufen. Der `when`-Teil ist nicht erfüllt, wenn sowohl X als auch Z einen Typ haben, der allgemeiner als die angegebenen Typen ist. Das heißt, wenn wir von einem Objekt wissen, dass es eine Liste ist, aber nicht wissen, ob die Liste Elemente enthält oder nicht, dann können wir auch keine Listenverknüpfung durchführen und verzögern deshalb die Ausführung so lange, bis genügend Information vorliegt.

undelayed_append arbeitet den Listenanfang ab und ruft dann wieder *append* auf. Je nach Situation ist *append* dann wieder blockiert, bis genügend Information vorhanden ist, d. h. bis der `when`-Teil erfüllt wird.

Die *append*-Relation besteht zwischen drei Listen. Die Anweisung `fun append(+,+,-).` sagt, wie die Relation als Funktion in Merkmalbeschreibungen verwendet werden kann. Ein '+' steht für ein Argument, das angegeben werden muss, und '−' steht an der Stelle des Wertes, der Funktionswert sein kann. Weil *append* so deklariert wurde, können wir schreiben `sem:rels:append(HRels,NHRels)`. Dieser Ausdruck besagt, dass das dritte Element in der *append*-Relation (das Ergebnis der Verknüpfung der beiden angegebenen Listen) der Wert des Pfades `sem:rels` ist.

(c) Flexionsregeln verändern die Relation nicht, die von der Tochter beigesteuert wird. Stellen Sie die Verfügbarkeit des SEM-Wertes am Mutterknoten durch entsprechende Koindizierung der SEM-Werte von Mutter und Tochter im Makro für Flexionslexikonregeln in der Datei lexrules.pl sicher.

(d) Laden Sie die Grammatik neu und entfernen Sie Fehler (falls es welche gibt). Stellen Sie sicher, dass die Abdeckung gleich bleibt und dass der RELS-Wert am 'S'-Knoten die Relationen aller Wörter im Satz enthält.

Was jetzt noch erreicht werden muss, ist die Verbindung von syntaktischen Argumenten und semantischen Rollen (*Linking*). Wir werden den INDEX-Wert von Argumenten (auch einfach *Index* genannt) benutzen, um diese Verbindung explizit zu machen. Dazu müssen wir alle Lexeme erweitern, die mit Argumenten kombiniert werden, d. h., die eine nicht-leere SUBCAT-Liste oder einen nicht-leere MOD-Liste haben: Wir müssen den INDEX-Wert von jedem Argument mit einer semantischen Rolle in der Relation des Funktors (dem semantischen Kopf) koindizieren.

(a) Nomina identifizieren den INDEX des Determinierers mit ihrem eigenen INDEX (und damit auch mit ihrem ARG0).

(b) Alle Verben identifizieren den INDEX ihres ersten Arguments mit der ARG1-Rolle. Zusätzlich identifizieren bivalente Verben den INDEX ihres zweiten Arguments mit ihrem ARG2. Bei ditransitiven Verben kommt dann entsprechend noch eine Verbindung zu ARG3 hinzu.

(c) Modifizierende Präpositionen identifizieren den INDEX der *syn-struc*, die über MOD selegiert wird mit ihrem eigenen Index. Der INDEX des Elements in der SUBCAT-Liste wird mit ARG1 identifiziert. (Komplementpräpositionen werden vorerst ignoriert.)

(d) Laden Sie die Grammatik neu, stellen Sie sicher, dass alles funktioniert und bewundern Sie die Schönheit der semantischen Komposition. Überprüfen Sie die folgenden Wohlgeformtheitsbedingungen in den Semantikrepräsentation von Sätzen wie *Der Hund bellt.* und *Die Katze gab dem Hund das Schaf.*: (i) Alle Indizes sind maximal spezifisch, d. h. entweder *event* oder *object*, (ii) in allen Nominalphrasen teilen Determinator und Nomen die ARG0-Variable, (iii) alle Rollen von verbalen Relationen sind an die Indizes der korrespondierenden Argumente gebunden und (iv) die ARG0-Variable einer modifizierenden Präpositionalphrase ist an das ARG0 des *event* bzw. des *object* gebunden, das modifiziert wird.

3. Das Trale-System enthält einen Chart-basierten Generator, der von einer semantischen Formel ausgehend alle Wortfolgen generieren kann, die diese semantische Repräsentation haben.

   Man kann Kanten in der Chart anklicken und bekommt dann einen Menüpunkt „Generiere“. Wenn man diesen Punkt anklickt, werden alle Strukturen generiert, die in den in `gen_pathes/1` vorgegeben Pfaden entsprechen. `gen_pathes/1` ist für die Grammatik 7 in `setup.pl` wie folgt definiert: `gen_pathes([[head],[subcat],[sem]]).` Das heißt beim Generieren werden Merkmalbeschreibungen erzeugt, die denselben HEAD-Wert, dieselbe SUBCAT-Liste und denselben SEM-Wert haben wie das linguistisch Objekt, das zur angeklickten Kante gehört.

   Parsen Sie den Satz *Den Hund jagt die Katze.*. Generieren Sie von der entsprechenden Bedeutungsrepräsentation ausgehend alle möglichen Sätze. Überlegen Sie, woran es liegen könnte, dass so viele ungrammatische Sätze von der Grammatik erzeugt werden.

References

Copestake, Ann, Daniel P. Flickinger, Carl J. Pollard & Ivan A. Sag. 2005. Minimal Recursion Semantics: An introduction. *Research on Language and Computation* 3(2–3). 281–332. `https://doi.org/10.1007/s11168-006-6327-9`.