Unzusammenhängende Ausdrücke

In natürlichen Sprachen können *unzusammenhängende* Ausdrücke auftreten, z.B.

- `Er will` <u>`das Gen`</u> `vermarkten,` <u>`das er entdeckt hat`</u>.
- `ein` <u>`nützliches`</u> `Gen` <u>`für die Fortpflanzung`</u>

Wir betrachten *gespaltene Wörter* $(u, v) \in \Sigma^* \times \Sigma^*$, d.h. unzusammenhängende Ausdrücke mit nur zwei Teilen.

Aus einem gespaltenen Wort (u, v) entsteht durch *Einfügung*

$$(u, v) : w \ := \ uvw$$

eines Worts $w \in \Sigma^*$ wieder ein (zusammenhängendes) Wort. Die Einfügung verallgemeinert man auf Mengen durch

$$R : A \ := \ \{ (u, v) : w \mid (u, v) \in R, w \in A \}.$$

Die Bildung eines gespaltenen Worts (u, v) als Paar von Wörtern wird durch das *kartesische Produkt*

$$A_1 \times A_2 \ := \ \{ (u, v) \mid u \in A_1, v \in A_2 \}$$

auf Mengen verallgemeinert.

Wir betrachten Grammatiken, die Kategorien A zusammenhängender Wörter sowie Kategorien R gespaltener Wörter verwenden (Wortmengen und Wortrelationen).

Der Aufbau der Ausdrücke erfolgt aus atomaren Ausdrücken $u, v, w \in \Sigma^*$ durch die Verkettung $u \cdot v$, die Paarbildung (u, v), und die Einfügung $(u, v) : w$.

Grammatik mit gespaltenen Ausdrücken

Die Bindungsstärke der Operationen sei abnehmend in $\cdot, +, : \times$ (bzw. `,` `;` `:` `x` in Prolog).

Beisp. 13 Kategorien zusammenhängender Ausdrücke sind `s,rel,pp,n,a,pr,vpart`, Kategorien gespaltener Ausdrücke sind `np,ap`.

⟨*discont.grm*⟩≡

```
start_symbol(s).

s --> np:[], aux, (np:vpart).
np --> (pron x []).
np --> (det, (n;(ap:n))) x (rel;[]).
ap --> a x (pp;[]).
rel --> relpron, (np:[]), vpart, aux.
pp --> pr, (np:[]).

word(der, det).    word(er, pron).
word(die, det).    word(das, relpron).
word(ein, det).

word(gen, n).
word(forscher, n).
word(fortpflanzung, n).

word(nuetzliches, a).
word(fuer, pr).

word(hat, aux).
word(vermarktet, vpart).
word(entdeckt, vpart).
```

Top-Down-Parser

für DCGs mit gespaltenen Konstituenten

⟨*Zusammengesetzte Kategorien*⟩≡

```
Cat := AtomicCat | (Cat,Cat) | (Cat;Cat) | {Goal}
     | (SpCat:Cat)
SpCat:= (Cat x Cat) | (SpCat;SpCat) | {Goal}
```

⟨*Baumkodierung*⟩≡

```
[Wurzelkategorie|Teilbäume]
```

⟨*discont-top-down-dcg.pl*⟩≡

```
:- op(600,yfy,'x').
:- op(1100,xfx,'-->').

parse(Atoms,Tree) :-
        start_symbol(S),
        parse(S,Tree,Atoms,[]), printTree(Tree,3,2).

parse((B,C),Tree,I,J) :-
        !, (parse(B,TreeB,I,K), parse(C,TreeC,K,J)),
        Tree = [*,TreeB,TreeC].
parse((B;C),Tree,I,J) :-
        !, (parse(B,Tree,I,J) ; parse(C,Tree,I,J)).
parse({Goal},Tree,I,J) :-
        !, call(Goal), Tree = [], I = J.
parse((B:C),Tree,I,J) :-
        !, (parse(B,TreeB,I,K,L,J),
            parse(C,TreeC,K,L)),
        Tree = [:,TreeB,TreeC].
parse([],Tree,I,J) :-
        !, Tree = [], I = J.
parse([W],Tree,I,J) :-
```

```
        !, Tree = [W], I = [W|J].

parse(A,Tree,I,J) :-
        (A --> Body),
        parse(Body,TreeBody,I,J),
        Tree = [A,TreeBody].
parse(A,Tree,I,J) :-
        I = [W|J],
        word(W,A),
        Tree = [A,W].

parse((B;C),Tree,I,K,L,J) :-
        !, (parse(B,Tree,I,K,L,J)
           ; parse(C,Tree,I,K,L,J)).
parse({Goal},Tree,I,K,L,J) :-
        !, call(Goal), Tree = [],
        I = K, L = J, K = L.
parse((B x C),Tree,I,K,L,J) :-
        !, (parse(B,TreeB,I,K),
            append(_,L,K), % K <= L
            parse(C,TreeC,L,J)),
        Tree = [x,TreeB,TreeC].

parse(A,Tree,I,K,L,J) :-
        (A --> Body),
        parse(Body,TreeBody,I,K,L,J),
        Tree = [A,TreeBody].
```

$\langle printTree \rangle$

Beispielanalyse

Mit der Grammatik <*discont.grm*> kann man den folgenden Satz mit zwei gespaltenen Ausdrücken wie folgt analysieren:

⟨*Beispiel*⟩≡

```
  ?- parse([der,forscher,hat,ein,nuetzliches,gen,
             fuer,die,fortpflanzung,entdeckt,
             das,er,vermarktet,hat],Tr).
      s
        *
          :
            np
              x
                *
                  det  der
                  n  forscher
                []
            []
          *
            aux  hat
            :
              np
                x
                  *
                    det  ein
                    :
                      ap
                        x
                          a  nuetzliches
                          pp
                            *
```

```
                                pr  fuer
                                :
                                  np
                                    x
                                      *
                                        det  die
                                        n  fortpflanzung
                                      []
                                  []
                        n  gen
                    rel
                      *
                        relpron  das
                        *
                          :
                            np
                              x
                                pron  er
                                []
                            []
                          *
                            vpart  vermarktet
                            aux  hat
                vpart  entdeckt
```

```
Tr = [s, [*, [:, [np, [x|...]], []],
          [*, [aux, hat], [:|...]]]]
```

Mit dem leeren Wort (`[]` in Prolog) verketten wir die beiden Komponenten eines gespaltenen Worts zu $(u, v) : \epsilon = uv$.

Die Einfügungen erscheinen im Syntaxbaum rückgängig gemacht, sind aber genau zu rekonstruieren.