Chart parsing with non-atomic categories

Three options for chart parsing with grammars employing non-atomic categories:

1. Expand the grammar into a CFG with atomic categories
2. Parse using an atomic CFG backbone with reduced information
3. Incorporate special mechanisms into the parser

Idea 1: Expand the grammar into a CFG with atomic categories

- number of categories grows exponentially, e.g., 3^n is size of category set with n binary features (plus, minus, unspecified)
- leads to a potentially huge set of rules
- grammar size relevant for time and space efficiency of parsing

Idea 2: Parse using an atomic CFG backbone with reduced information

- idea:
  - parse using a property defined for all categories
  - use other properties to filter solutions from set of parses
- downside:
  - parsing with partial information can significantly enlarge the search space

Idea 3: Incorporate special mechanism into parser

- The equality check used for atomic categories has to be replaced by **unification**.
- Every active and inactive edge in a chart may be used for different uses. So for each time an edge is used, a new **copy** needs to be made.
- Revise the duplication check: only add an edge if it is not **subsumed** by an edge already in the chart.

- Two efficiency issues:
  - intelligent **indexing** of edges in chart
  - **packing** of similar edges in chart (cf. Tomita parser)

Earley parser with atomic categories

Prediction: for each ${}_i[A \;\rightarrow\; \alpha \;\bullet_j\; B\;\beta]$ in chart
for each $B \rightarrow \gamma$ in rules
add ${}_j[B \;\rightarrow\; \bullet_j\; \gamma]$ to chart

Scanning: let $w_1 \ldots w_j \ldots w_n$ be the input string
for each ${}_i[A \;\rightarrow\; \alpha \;\bullet_{j-1}\; w_j\;\beta]$ in chart
add ${}_i[A \;\rightarrow\; \alpha\; w_j \;\bullet_j\; \beta]$ to chart

Completion (fundamental rule of chart parsing):

for each ${}_i[A \;\rightarrow\; \alpha \;\bullet_k\; B\;\beta]$ and ${}_k[B \;\rightarrow\; \gamma \;\bullet_j\;]$ in chart
add ${}_i[A \;\rightarrow\; \alpha\; B \;\bullet_j\; \beta]$ to chart

Earley parser with unification

Prediction:
for each ${}_i[A \;\rightarrow\; \alpha \;\bullet_j\; B\;\beta]$ in chart
for each $B' \rightarrow \gamma$ in rules
add ${}_j[\sigma(B \;\rightarrow\; \bullet_j\; \gamma)]$ with $\sigma = \text{mgu}(B, B')$ to chart

Completion (fundamental rule of chart parsing):

for each ${}_i[A \;\rightarrow\; \alpha \;\bullet_k\; B\;\beta]$ and ${}_k[B' \;\rightarrow\; \gamma \;\bullet_j\;]$ in chart
add ${}_i[\sigma(A \;\rightarrow\; \alpha\; B \;\bullet_j\; \beta)]$ with $\sigma = \text{mgu}(B, B')$ to chart

Using restriction to prevent prediction loops

- Prediction terminates for grammars with atomic categories, since a new item is only added to the chart if not already there and there is a finite number of atomic categories.
- Moving beyond atomic categories, there can be an infinite number of non-atomic categories.
- Prediction loop on left-recursive rules can be problem again.
- Solution: use **restriction** on prediction substitution to limit to finite number of cases

An example for a problematic grammar

Shieber/Shabes/Pereira (1994, p. 13): Grammar accepting ab^n with N being instantiated to the successor representation of n.

$$\begin{array}{l} \mathbf{start} \rightarrow \mathbf{r}(0, N) \\ \mathbf{r}(X, N) \rightarrow \mathbf{r}(s(X), N)\ \mathbf{b} \\ \mathbf{r}(N, N) \rightarrow \mathbf{a} \end{array}$$

Prediction step with unification will loop:

$$\begin{array}{l} {}_0[\mathbf{start} \rightarrow\ \bullet_0\ \mathbf{r}(0, N)] \\ {}_0[\mathbf{r}(0, N) \rightarrow\ \bullet_0\ \mathbf{r}(s(0), N)\ \mathbf{b}] \\ {}_0[\mathbf{r}(s(0), N) \rightarrow\ \bullet_0\ \mathbf{r}(s(s(0)), N)\ \mathbf{b}] \\ {}_0[\mathbf{r}(s(s(0)), N) \rightarrow\ \bullet_0\ \mathbf{r}(s(s(s(0))), N)\ \mathbf{b}] \\ \ldots \end{array}$$

Prediction with restriction

for each ${}_i[A\ \rightarrow\ \alpha\ \bullet_j\ B\ \beta]$ in chart
  for each $B' \rightarrow \gamma$ in rules
    add ${}_j[\sigma(B\ \rightarrow\ \bullet_j\ \gamma)]$ with $\sigma =$ restriction(mgu(B, B')) to chart

– restriction(mgu(B, B')) can be any operation reducing the number of possible substitutions to finite classes:

(a) depth bound on term complexity
(b) elimination of terms that are known to grow indefinitely
(c) use only of selected terms known not to grow indefinitely

– sound since predicted edge only step towards completion!