Introduction to CL – CFG-Parsing

1 Left Corner relation – Transitive Closure

We do a least fixpoint computation to compute the closure. Initialize every set $LC(A)$ with the one-step left-corner relation items and add those that come in indirectly until no new candidates are found. This algorithm can be further optimized: In line 7, only those (non)terminals C should be considered which have been added in the last **while** round, or the initialization, if it's the first.

```
1 for A → Bβ ∈ P do LC(A) = {B}
2 changed = true
3 while changed do
4         changed = false
5         for A ∈ N do
6             for B ∈ LC(A) ∩ N do
7                 for C ∈ LC(B) do
8                     if C ∉ LC(A) then LC(A) = LC(A) ∪ {C}; changed = true
```

2 Extraction of complete parse trees

extract_trees extracts all trees rooted in the nonterminal N reaching from s to e in the chart. To get all full parse trees, call extract_trees$(S, 0, n)$ if S is in $\mathcal{C}[0, n]$. Otherwise, the input string is not in the language of the grammar.

```
extract_trees(N, s, e)
  if e = s + 1 ∧ N → a_e ∈ P return {tree(N)}      // preterminal leaf
  result_trees = {}
  for all k ∈ B[s, e]                               // check all split points
      for all A ∈ C[s, k]                           // check all possible left daughters
          for all B ∈ C[k, e]                       // check all possible right daughters
              if N → AB ∈ P                         // look for appropriate productions
                  left_trees = extract_trees(A, s, k)
                  right_trees = extract_trees(B, k, e)
                  for left in left_trees
                      for right in right_trees add tree(N, left, right) to result_trees
  return result_trees
```

3 Parse-tree extraction – run time

Because the number of parse trees may be exponential, this parse tree extraction algorithm has exponential worst case complexity.

4 Bottom-up vs. Earley/left corner parsing

Bottom-up parsing is advantageous in cases where all sub-constituents derived by a given grammar are useful, e.g., in robust parsing, where sub-constituents can be used to construct a partial representation of the input string's content.

Earley or left-corner parsing have a better average case run-time for cases where only complete parses are of interest and efficiency is important.