Finite-state parsing
- Deterministic bottom-up parsing (Abney 1996)
- Finite-State Intersection Grammar (Koskenneimi 1990)
- Incremental Finite-State Parsing (Aït-Mokhtar & Chanod 1997)
- Parsing top-down (Roche 1993) (not in this course)
- Light Parsing as Finite-State Filtering (Grefenstette 1996) (not in this course)
- FS-approximations of context-free grammars (Pereira & Wright 1997) (not in this course)
- Etc.

Some parsing issues
- Accuracy
- Speed
- Space requirements
- Robustness
- Learning
- Partiality, Underspecificaton, Shallowness
- Declarativity
- Order-independence of rules

Problems with traditional parsers
- Correct lowlevel parses are often rejected because they do not fit into a global parse -> brittleness
- Ambiguity -> indeterminism -> search -> slow parsers (e.g. the complexity of parsing by means of CFGs is $O(n^3)$)
- Ambiguity -> sometimes hundreds of thousands of parse trees, and what can we do with these? (But see the work of Tomita...)

Another strategy (Abney)
- Start with the simplest constructions (easy-first parsing) and be as careful as possible when parsing them -> 'islands of certainty'
- 'islands of certainty' -> do not reject these parses even if they do not fit into a global parse -> robustness
- When you are almost sure of how to resolve an ambiguity, do it! -> determinism
- When you are uncertain of how to resolve an ambiguity, don’t even try! -> ‘containment of ambiguity’ -> determinism
- determinism -> no search -> speed ( $O(n)$ complexity)

Shallow syntax
- analyses less complete than conventional parser output
- identifies some phrasal constituents (e.g. NPs), without indicating their internal structure and their function in the sentence
- or identifies the functional role of some of the words, such as the main verb, and its direct arguments.
Two kinds of approaches

- The constructive approach
  - Identifies basic phrases (chunks) through pattern-matching
  - Builds complex phrases (up to a certain level) on top of these
  - Examples: Abney’s Partial Parsing, Hobbs et al.’s FASTUS, Grefenstette’s ‘Light Parsing’, ...

- The reductionist approach
  - Starts from a large number of alternative analyses (obtained through lexical lookup and morphological analysis), that gets reduced through the application of constraints.
  - Examples: Karlsson et al.’s ‘Constraint grammar’, Koskennemi’s ‘Finite-State Intersection Grammar’, Roches ‘Negative Grammar’, ...

Deterministic bottom-up parsing

- Described in Abney 1996
- Uses a constructive strategy
- Analysis takes place on different levels
- Chunks = non-recursive, basic NPs, VPs, PPs, APs, AdvPs
- Easy-first parsing
- Islands of certainty
- Longest-match strategy
- Containment of ambiguity

Deterministic bottom-up parsing with XFST

- Adapted from Karttunen 1996:

  define NP {[d] a* n} :
  regex NP @-> "[NP] ... "]
  v "[NP] NP ... ] @-> "[VP] ... "] :
  apply down dannvaan
  [NP dann][VP v [NP aan]]

  Note the use of the longest-match operator!

Finite-State Intersection Grammar

- First described in Koskenniemi 1990
- Uses a reductionist strategy
- Relates to (and predates?) Negative Grammar

Sentence:

Grammar:

Analysis:

Sentence:

Grammar:

Analysis:
Finite-State Intersection Grammar in XFST

```plaintext
define Grammar ~$[subj ?* subj] & ~$[obj ?* obj];
define Sentence [Lisa [subj|obj] älskar pred Slugge [subj|obj]]; # let's parse!
regex Sentence & Grammar;
print net
```

Relating Negative Grammar and Finite-State Intersection Grammar

- Negative grammar
  ```plaintext
define S [he Pro hopes [N|V] that [Pro|Conj|Det] this [Det|Pro] works [N|V];
define C1 [that Det this Det];
define C2 [this Det ? V];
define S'' ~$ - [?* [C1|C2] ?*];
print net S''
```

- Finite-State Intersection Grammar
  ```plaintext
define S [he Pro hopes [N|V] that [Pro|Conj|Det] this [Det|Pro] works [N|V];
define C1 ~$[that Det this Det];
define C2 ~$[this Det ? V];
define S'' [S & C1 & C2];
print net S''
```

The restriction operator in Finite-State Intersection Grammar

Example:

```
Prep => _
Coord Prep |
~$[NounHead | VerbHead | Prep ]PPObj |
~$[NounHead | VerbHead | Prep ][ Inf | PresPart ] |
Adverbial |
NounPsMod
```

Incremental Finite-State Parsing

- AP example:
  ```plaintext
define AP [ ADV* ADJ] ;
read regex AP @-> "[AP " ... "AP"] ;
```

- NP example:
  ```plaintext
  [DET|NUM|PRO] -> "NP" ... 
  [N|NUM|PRO] ... ] -> "NP" :
  ["NP" ~$["NP"] "NP"] @.@ "NP" ... "NP" :
  ["NP" | "NP"] -> [ ];
```

- SUBJ example:
  ```plaintext
  ["NP" ~$["NP"] "NP"] @.@ Subj :
  Subj -> [ ] || Subj VFin ~$[Subj] _;

  Note that the Subject example uses a reductionist strategy (although implemented differently than in Finite-State Intersection Grammar)
```
A tip

- define NUM [Letter+ %+ M TagChar* SP*];
- define ADJ [Letter+ %+ A TagChar* SP*];