8) Unification-based Grammars and Parsing

Paula Buttery

Dept of Computer Science & Technology, University of Cambridge
Reminder...

Last time we looked at lexicalisation and features to help us with:

- modelling structural dependency across the tree as a whole
  - e.g. correctly modelling \textit{NP} expansion

- modelling the structural behaviour specific to a lexical item:
  - pp-attachment
  - subcategorisation
  - co-ordination
Alternative approach represents features in \textbf{DAGs}

Re-conceptualise words, non-terminal nodes and parses as \textbf{Directed Acyclic Graphs} which may be represented as \textbf{Attribute Value Matrices}

We have \textbf{atomic values} at each of the terminal nodes and another \textbf{AVM/DAG} at all other nodes
Some grammars allow the AVMs to be typed.

Typing facilitates grammar building. Hierarchies of AVM types can be used to automatically populate attributes.

An shorthand notation uses angle bracket notation to indicate attribute paths: e.g. \texttt{<NP AGREEMENT PERSON>} would represent the attribute path leading to the atomic value 3\textit{rd}.

\[
NP \rightarrow \text{AGREEMENT} \rightarrow \bullet \rightarrow \text{NUMBER} \rightarrow \bullet \text{num}
\]

\[
\begin{tikzpicture}
  \node (np) {\textsc{NP}};
  \node (agreement) [below of=np] {\textsc{AGREEMENT}};
  \node (number) [below of=agreement] {\textsc{NUMBER}};
  \node (num) [right of=number] {\textsc{num}};
  \path (np) edge node [left] {\textbullet} (agreement);
  \path (agreement) edge node [left] {\textbullet} (number);
  \path (number) edge node [right] {\textbullet} (num);
  \node (person) [below of=number] {\textsc{PERSON}};
  \node (agreement Person) [right of=person] {\textsc{AGREEMENT}};
  \node (number Person) [below of=agreement Person] {\textsc{NUMBER}};
  \node (num Person) [right of=number Person] {\textsc{num}};
  \node (3rd Person) [below of=num Person] {\textsc{3rd}};
  \path (person) edge node [left] {\textbullet} (agreement Person);
  \path (agreement Person) edge node [left] {\textbullet} (number Person);
  \path (number Person) edge node [right] {\textbullet} (num Person);
  \path (num Person) edge node [right] {\textbullet} (3rd Person);
\end{tikzpicture}
\]
DAGs and AVMs may exhibit **re-entrancy**

Notice that re-entrancy indicates that a feature structure is being encountered twice (we have two ways of getting to the same node); for the purpose of grammar design this is subtly different to simply encountering two feature paths that eventually evaluate to the same atomic categories.

1. **Non re-entrant**:

   ![Non re-entrant diagram]

2. **Re-entrant**:

   ![Re-entrant diagram]
DAGs and AVMs may exhibit **re-entrancy**

1. Non re-entrant: \[
\begin{bmatrix}
\text{FEATURE1} & a \\
\text{FEATURE2} & a \\
\end{bmatrix}
\]

2. Re-entrant: \[
\begin{bmatrix}
\text{FEATURE1} & 1 & a \\
\text{FEATURE2} & 1 \\
\end{bmatrix}
\]

Note that re-entrancy indicates that a feature structure is encountered twice (i.e., we have two ways of reaching the same node). For the purpose of grammar design this is subtly different to simply encountering two feature paths that eventually evaluate to the same atomic categories.
Parsing with DAGs involves **Unification**

- The unification of two DAGs is the most specific DAG which contains all the information in both of the original attribute-value structures.
- Unification fails if the two DAGs contain conflicting information.
Parsing with DAGs involves **Unification**

- The unification of two DAGs is the most specific DAG which contains all the information in both of the original attribute-value structures.
- Unification fails if the two DAGs contain conflicting information.

\[
\begin{align*}
\text{[PERSON } 3\text{rd]} & \sqcap \text{[NUMBER plural]} = \text{[PERSON NUMBER } 3\text{rd plural]} \\
\end{align*}
\]
Parsing with DAGs involves **Unification**

- The unification of two DAGs is the most specific DAG which contains all the information in both of the original attribute-value structures.
- Unification fails if the two DAGs contain conflicting information.

\[
\begin{align*}
\begin{bmatrix}
\text{PERSON} & 3\text{rd} \\
\end{bmatrix} & \sqcup \\
\begin{bmatrix}
\text{NUMBER} & \text{plural} \\
\end{bmatrix} & = \\
\begin{bmatrix}
\text{PERSON} & 3\text{rd} \\
\text{NUMBER} & \text{plural} \\
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\begin{bmatrix}
\text{PERSON} & 1\text{st} \\
\text{NUMBER} & \text{plural} \\
\end{bmatrix} & \sqcup \\
\begin{bmatrix}
\text{NUMBER} & \text{num} \\
\end{bmatrix} & = \\
\begin{bmatrix}
\text{PERSON} & 1\text{st} \\
\text{NUMBER} & \text{num} \\
\end{bmatrix}
\end{align*}
\]

unification fails

\[
\begin{bmatrix}
\text{feature1} \\
\text{feature2} \\
\text{feature3} \\
\end{bmatrix} & \sqcup \\
\begin{bmatrix}
\text{feature3} & a \\
\end{bmatrix} & = \\
\begin{bmatrix}
\text{feature1} \\
\text{feature2} & a \\
\text{feature3} & a \\
\end{bmatrix}
\]

unification fails
Parsing with DAGs involves **Unification**

- The unification of two DAGs is the most specific DAG which contains all the information in both of the original attribute-value structures.
- Unification fails if the two DAGs contain conflicting information.

\[
\begin{array}{ccc}
[\text{PERSON} \ 3rd] & \sqcup & [\text{NUMBER} \ plural] \\
[\text{PERSON} \ 1st] & \sqcup & [\text{NUMBER} \ num] \\
\end{array}
\]

\[
\begin{array}{ccc}
= & \text{unification fails} \\
\end{array}
\]
Parsing with DAGs involves **Unification**

- The unification of two DAGs is the most specific DAG which contains all the information in both of the original attribute-value structures.
- Unification fails if the two DAGs contain conflicting information.

\[
\begin{align*}
\text{[PERSON} & \text{3rd]} \sqcup [\text{NUMBER} \text{ plural}] = [\text{PERSON} \text{ 3rd} \text{ NUMBER} \text{ plural}] \\
\text{[PERSON} & \text{1st NUMBER} \text{ plural}] \sqcup [\text{NUMBER} \text{ num}] = [\text{PERSON} \text{ 1st NUMBER} \text{ plural}] \\
\text{[PERSON} & \text{1st NUMBER} \text{ sing}] \sqcup [\text{NUMBER} \text{ plural}] =
\end{align*}
\]
Parsing with DAGs involves **Unification**

- The unification of two DAGs is the most specific DAG which contains all the information in both of the original attribute-value structures.
- Unification fails if the two DAGs contain conflicting information.

\[
\begin{align*}
\left[ \text{PERSON} \ 3rd \right] & \sqcup \left[ \text{NUMBER} \ plural \right] = \left[ \text{PERSON} \ 3rd \right. \\
\left[ \text{PERSON} \ 1st \number \ plural \right] & \sqcup \left[ \text{NUMBER} \ num \right] = \left[ \text{PERSON} \ 1st \right. \\
\left[ \text{PERSON} \ 1st \number \ sing \right] & \sqcup \left[ \text{NUMBER} \ plural \right] = \text{unification fails}
\end{align*}
\]
 Parsing with DAGs involves **Unification**

- The unification of two DAGs is the most specific DAG which contains all the information in both of the original attribute-value structures.
- Unification fails if the two DAGs contain conflicting information.

\[
\begin{align*}
\begin{bmatrix}
\text{PERSON} & 3rd \\
\text{NUMBER} & \text{plural}
\end{bmatrix} \sqcup 
\begin{bmatrix}
\text{PERSON} & 3rd \\
\text{NUMBER} & \text{plural}
\end{bmatrix} &= 
\begin{bmatrix}
\text{PERSON} & 3rd \\
\text{NUMBER} & \text{plural}
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\begin{bmatrix}
\text{PERSON} & 1st \\
\text{NUMBER} & \text{plural}
\end{bmatrix} \sqcup 
\begin{bmatrix}
\text{PERSON} & 1st \\
\text{NUMBER} & \text{num}
\end{bmatrix} &= 
\begin{bmatrix}
\text{PERSON} & 1st \\
\text{NUMBER} & \text{plural}
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\begin{bmatrix}
\text{PERSON} & 1st \\
\text{NUMBER} & \text{sing}
\end{bmatrix} \sqcup 
\begin{bmatrix}
\text{PERSON} & 1st \\
\text{NUMBER} & \text{plural}
\end{bmatrix} &= 
\text{unification fails}
\end{align*}
\]

\[
\begin{align*}
\begin{bmatrix}
\text{FEATURE1} \\
\text{FEATURE2} [1] \\
\text{FEATURE3} [1]
\end{bmatrix} \sqcup 
\begin{bmatrix}
\text{FEATURE2} [1] \\
\text{FEATURE3} a
\end{bmatrix} &= 
\begin{bmatrix}
\text{FEATURE1} \\
\text{FEATURE2} [1] \\
\text{FEATURE3} a
\end{bmatrix}
\end{align*}
\]
The unification of two DAGs is the most specific DAG which contains all the information in both of the original attribute-value structures.

Unification fails if the two DAGs contain conflicting information.
Unification examples…
Unification algorithm requires extra graph structure

As with all recursive algorithms, the next step is to test for the various base cases of the recursion before proceeding on to a recursive call involving some part of the original arguments. In this case, there are three possible base cases:

- The arguments are identical
- One or both of the arguments has a null value
- The arguments are non-null and non-identical

If the structures are identical, then the pointer of the first is set to the second and the second is returned. It is important to understand why this pointer exchange is done in this case. After all, since the arguments are identical, returning either one would appear to suffice. This might be true for a single unification but recall that we want the two arguments to the unification operator to be truly unified. The pointer change is necessary since we want the arguments to be truly identical, so that any subsequent unification that adds information to one will add it to both.

In the case where either of the arguments is null, the pointer field for the null argument is changed to point to the other argument, which is then returned. The result is that both structures now point at the same value.

If neither of the preceding tests is true then there are two possibilities: they are non-identical atomic values, or they are non-identical complex structures. The former

From Jurafsky and Martin version 2
Unification algorithm requires extra graph structure

Figure 15.8 The final result of unifying F1 and F2.

\[
\begin{bmatrix}
\text{AGREEMENT 1} \\
\text{SUBJECT} \\
\end{bmatrix}
\sqcup
\begin{bmatrix}
\text{AGREEMENT 1} \\
\text{PERSON 3rd} \\
\end{bmatrix}
\]

Figure 15.10 shows the extended representations for the arguments to this unification. These original arguments are neither identical, nor null, nor atomic, so the main loop is entered. Looping over the features of \( f_2 \), the algorithm makes a recursive attempt to unify the values of the corresponding \text{SUBJECT} features of \( f_1 \) and \( f_2 \).

\[
\begin{bmatrix}
\text{NUMBER sg} \\
\text{PERSON null} \\
\end{bmatrix}
\sqcup
\begin{bmatrix}
\text{PERSON 3rd} \\
\end{bmatrix}
\]

In looping over the features of the second argument, the fact that the first argument lacks a \text{PERSON} feature is discovered. A \text{PERSON} feature initialized with a \text{NULL} value is, therefore, added to the first argument. This, in effect, changes the previous unification to the following.

\[
\begin{bmatrix}
\text{NUMBER sg} \\
\text{PERSON null} \\
\end{bmatrix}
\sqcup
\begin{bmatrix}
\text{PERSON 3rd} \\
\end{bmatrix}
\]

From Jurafsky and Martin version 2
DAGs can be straightforwardly associated with the lexicon.

\[
\begin{align*}
\text{N} & \quad \rightarrow \quad \{\text{fish, rivers, pools, they}\} \\
\text{AGREEMENT} & \quad \rightarrow \\
\begin{bmatrix}
\text{PERSON} & 3rd \\
\text{NUMBER} & \text{plural}
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{V} & \quad \rightarrow \quad \{\text{cans, fishes}\} \\
\langle \text{V AGREEMENT PERSON} \rangle & = 3\text{rd} \\
\langle \text{V AGREEMENT NUMBER} \rangle & = \text{sing}
\end{align*}
\]

\[
\begin{align*}
\langle \text{they,} \text{AGREEMENT} \rangle & \quad \rightarrow \\
\begin{bmatrix}
\text{PERSON} & 3rd \\
\text{NUMBER} & \text{sing}
\end{bmatrix}
\end{align*}
\]
DAGs can be straightforwardly associated with the lexicon

\[
\begin{align*}
\text{N} & \quad \rightarrow \quad \{\text{fish, rivers, pools, they}\} \\
\text{AGREEMENT} & \quad \rightarrow \quad \{\text{cans, fishes}\} \\
\text{V} & \quad \rightarrow \quad \{\text{cans, fishes}\} \\
\quad & \quad \rightarrow \quad \{\text{fish, rivers, pools, they}\}
\end{align*}
\]

Now we have methods for representing feature structures and can use feature path notation:

\[
\text{V} \rightarrow \{\text{cans, fishes}\} \\
\langle\text{V AGREEMENT person}\rangle = 3rd \\
\langle\text{V AGREEMENT number}\rangle = \text{sing}
\]

Recall the mini ruleset from earlier in the lecture that became at least three times larger when we tried to add simple agreement (reproving the over-generation problem without inflating the rule set). The next step is to integrate them into our CFGs to see if we might improve the right or wrong way of expressing the non-terminal categories and over-generate as rules with feature path constraints:

\[
\text{V} \rightarrow \{\text{cans, fishes}\}
\]

As rule based AVMs

Now we have a rule notation that allows us to succinctly express things: they list out all the feature constraints beneath the rules and are able to simple match feature structures against those in our new grammar rules as:

\[
\text{V} \rightarrow \{\text{cans, fishes}\} \\
\langle\text{V AGREEMENT person}\rangle = 3rd \\
\langle\text{V AGREEMENT number}\rangle = \text{sing}
\]

They, they would be there would be \{\text{fish, rivers, pools, they}\} to indicate that there is of course intentional since it allows there might be \{\text{it, fish, rivers, pools, I, you, December, Scotland, they}\}.

The lexical elements of the grammar may also be specified in several ways:

- By feature structures, is is compatible:
  \[
  \text{N} \quad \rightarrow \quad \{\text{fish, rivers, pools, they}\} \\
  \text{AGREEMENT} \quad \rightarrow \quad \{\text{fish, rivers, pools, they}\} \\
  \text{V} \quad \rightarrow \quad \{\text{cans, fishes}\} \\
  \langle\text{V AGREEMENT person}\rangle = 3rd \\
  \langle\text{V AGREEMENT number}\rangle = \text{sing}
  \]

- Or as a tuple to be looked up in a lexicon (this is how the lexicon is implemented): a tuple is simply \(\langle\text{N AGREEMENT person}\rangle, \langle\text{N AGREEMENT number}\rangle\) to be the unification operator. The following are some unification examples:

- As rules with feature path constraints:
  \[
  \text{V} \rightarrow \{\text{cans, fishes}\} \\
  \langle\text{V AGREEMENT person}\rangle = 3rd \\
  \langle\text{V AGREEMENT number}\rangle = \text{sing}
  \]
DAGs can be straightforwardly associated with the lexicon.

\[
\begin{align*}
N & \rightarrow \{\text{fish, rivers, pools, they}\} \\
\text{AGREEMENT} & \left[ \begin{array}{c}
\text{PERSON} \\
\text{NUMBER}
\end{array} \right] \\
\end{align*}
\]

\[
\begin{align*}
V & \rightarrow \{\text{cans, fishes}\} \\
\langle V \text{ AGREEMENT PERSON} \rangle & = 3rd \\
\langle V \text{ AGREEMENT NUMBER} \rangle & = \text{sing}
\end{align*}
\]

\[
\begin{align*}
\langle \text{they,} \rangle \\
\text{AGREEMENT} & \left[ \begin{array}{c}
\text{PERSON} \\
\text{NUMBER}
\end{array} \right] \\
\end{align*}
\]
We can modify CFG algorithms to parse with DAGs

- We can use any CFG parsing algorithm if:
  - associate attribute paths with CFG rules
  - unify DAGs in the states

\[
S \to NP \ VP \\
< NP \ HEAD \ AGREEMENT > = < VP \ HEAD \ AGREEMENT > \\
< S \ HEAD > = < VP \ HEAD >
\]

- We would have items like \([X, \{a, b\}, \text{DAG}]\) on the agenda or at each cell
Parsing example...

They like Peter

```
S
  AGREEMENT 1
→ [NP
    AGREEMENT 1
  ] [VP
    AGREEMENT 1
  ]

VP
  AGREEMENT 1
→ [V
    AGREEMENT 1
  ]

V
  AGREEMENT [PERSON 3rd plural]
→ {like}

V
  AGREEMENT [PERSON 3rd sing]
→ {likes}

NP
  AGREEMENT [PERSON 3rd plural]
→ {fish, rivers, pools, they}

NP
  AGREEMENT [PERSON 3rd sing]
→ {it, fish, Peter}
```
They like Peter

\[ S \]
\[ \text{AGREEMENT 1} \]
\[ \rightarrow \]
\[ \text{NP} \]
\[ \text{AGREEMENT 1} \]
\[ \text{VP} \]
\[ \text{AGREEMENT 1} \]
\[ \rightarrow \]
\[ \text{V} \]
\[ \text{agreement 1} \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \text{VP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{V} \]
\[ \text{agreement 1} \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \text{VP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{V} \]
\[ \text{agreement 1} \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \text{VP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{V} \]
\[ \text{agreement 1} \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \text{VP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{V} \]
\[ \text{agreement 1} \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \text{VP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{V} \]
\[ \text{agreement 1} \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \text{VP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{V} \]
\[ \text{agreement 1} \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \text{VP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{V} \]
\[ \text{agreement 1} \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \text{VP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{V} \]
\[ \text{agreement 1} \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \text{VP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{V} \]
\[ \text{agreement 1} \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \text{VP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{V} \]
\[ \text{agreement 1} \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \text{VP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{V} \]
\[ \text{agreement 1} \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \text{VP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{V} \]
\[ \text{agreement 1} \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{NP} \]
\[ \text{agreement 1} \]
\[ \text{VP} \]
\[ \text{agreement 1} \]
\[ \rightarrow \]
\[ \text{V} \]
\[ \text{agreement 1} \]
\[ \text{NP} \]
They like Peter

\[
\begin{align*}
S \quad \text{AGREEMENT} & \quad \{1\} \\
VP \quad \text{AGREEMENT} & \quad \{1\} \\
V \quad \text{AGREEMENT} \quad \{\text{PERSON}, 3rd, \text{NUMBER}, \text{plural}\} & \quad \{\text{like}\} \\
V \quad \text{AGREEMENT} \quad \{\text{PERSON}, 3rd, \text{NUMBER}, \text{sing}\} & \quad \{\text{likes}\} \\
NP \quad \text{AGREEMENT} \quad \{\text{PERSON}, 3rd, \text{NUMBER}, \text{plural}\} & \quad \{\text{fish, rivers, pools, they}\} \\
NP \quad \text{AGREEMENT} \quad \{\text{PERSON}, 3rd, \text{NUMBER}, \text{sing}\} & \quad \{\text{it, fish, Peter}\}
\end{align*}
\]
They like Peter

They like Peter

\[
\begin{align*}
S &\rightarrow NP \ [AGREEMENT \ 1] \ VP \ [AGREEMENT \ 1] \\
VP &\rightarrow V \ [AGREEMENT \ 1] \ NP \ [AGREEMENT \ 1] \\
V &\rightarrow \{\text{like}\} \\
NP &\rightarrow \{\text{fish, rivers, pools, they}\} \\
\end{align*}
\]
They like Peter

[$S$
  AGREEMENT [1]] -> [$NP$
  AGREEMENT [1] [$VP$
  AGREEMENT [1]]

[$VP$
  AGREEMENT [1]]

[$V$
  AGREEMENT
  [PERSON
    3rd
    number
    plural]]

[$V$
  AGREEMENT
  [PERSON
    3rd
    number
    sing]]

[$NP$
  AGREEMENT
  [PERSON
    3rd
    number
    plural]]

[$NP$
  AGREEMENT
  [PERSON
    3rd
    number
    sing]]

} {like}

} {likes}

} {fish, rivers, pools, they}

} {it, fish, Peter}
They like Peter

\[
\begin{align*}
S & \quad \text{AGREEMENT 1} \\
\text{VP} & \quad \text{AGREEMENT 1} \\
V & \quad \text{AGREEMENT [PERSON 3rd number plural]} \\
\text{NP} & \quad \text{AGREEMENT [PERSON 3rd number plural]} \\
\text{NP} & \quad \text{AGREEMENT [PERSON 3rd number sing]} \\
\end{align*}
\]
They like Peter

[\textbf{S} \text{AGREEMENT 1}] → [\textbf{NP} \text{AGREEMENT 1} [\textbf{VP} \text{AGREEMENT 1}]]

[\textbf{V} \text{AGREEMENT 1} [\text{PERSON NUMBER 3rd \textit{plural}}]] → \{\text{like}\}

[\textbf{V} \text{AGREEMENT 1} [\text{PERSON NUMBER 3rd \textit{singular}}]] → \{\text{likes}\}

[\textbf{NP} \text{AGREEMENT 1} [\text{PERSON NUMBER 3rd \textit{plural}}]] → \{\text{fish, rivers, pools, they}\}

[\textbf{NP} \text{AGREEMENT 1} [\text{PERSON NUMBER 3rd \textit{singular}}]] → \{\text{it, fish, Peter}\}
They like Peter

\[
\begin{align*}
\text{They} & \rightarrow [\text{NP}] \\
\text{like} & \rightarrow \{\text{like}\} \\
\text{Peter} & \rightarrow \{\text{it, fish, Peter}\}
\end{align*}
\]
They like Peter

```
S
 AGREEMENT 1

NP
 AGREEMENT 1
 VP
 AGREEMENT 1
  V
 AGREEMENT 1
 NP
 AGREEMENT 1
 NP
 AGREEMENT 1
```

→ {like}

→ {likes}

→ {fish, rivers, pools, they}

→ {it, fish, Peter}
They like Peter

```
[\textbf{S} \hspace{1cm} \text{AGREEMENT 1}] \rightarrow [\textbf{NP} \hspace{1cm} \text{AGREEMENT 1} [\text{PERSON NUMBER 3rd plural}]] \rightarrow [\textbf{VP} \hspace{1cm} \text{AGREEMENT 1} [\text{PERSON NUMBER 3rd plural}]] \rightarrow [\textbf{NP} \hspace{1cm} \text{AGREEMENT 1} [\text{PERSON NUMBER 3rd plural}]]
```

```
[\textbf{VP} \hspace{1cm} \text{AGREEMENT 1} [\text{PERSON NUMBER 3rd plural}]] \rightarrow [\textbf{V} \hspace{1cm} \text{AGREEMENT 1} [\text{PERSON NUMBER 3rd plural}]] 
```

```
[\textbf{V} \hspace{1cm} \text{AGREEMENT 1} [\text{PERSON NUMBER 3rd sing}]] \rightarrow \{\text{like}\}
```

```
[\textbf{V} \hspace{1cm} \text{AGREEMENT 1} [\text{PERSON NUMBER 3rd sing}]] \rightarrow \{\text{likes}\}
```

```
[\textbf{NP} \hspace{1cm} \text{AGREEMENT 1} [\text{PERSON NUMBER 3rd plural}]] \rightarrow \{\text{fish, rivers, pools, they}\}
```

```
[\textbf{NP} \hspace{1cm} \text{AGREEMENT 1} [\text{PERSON NUMBER 3rd plural}]] \rightarrow \{\text{it, fish, Peter}\}
```

```
[\textbf{NP} \hspace{1cm} \text{AGREEMENT 1} [\text{PERSON NUMBER 3rd sing}]] \rightarrow \{\text{it, fish, Peter}\}
```
They like Peter

\[
\begin{align*}
S & \rightarrow NP \\
& \quad \rightarrow VP \\
& \quad \rightarrow V \\
& \quad \rightarrow \{\text{like}\} \\
& \quad \rightarrow \{\text{likes}\} \\
& \quad \rightarrow \{\text{fish, rivers, pools, they}\} \\
& \quad \rightarrow \{\text{it, fish, Peter}\}
\end{align*}
\]
They likes Peter

```
S
AGREEMENT [1] → NP
AGREEMENT [1] [PERSON NUMBER 3rd plural] VP
AGREEMENT [1] [PERSON NUMBER 3rd sing] NP
AGREEMENT [1] [PERSON NUMBER 3rd sing]

VP
AGREEMENT [1] [PERSON NUMBER 3rd sing] → V
AGREEMENT [1] [PERSON NUMBER 3rd sing] NP
AGREEMENT [1] [PERSON NUMBER 3rd sing]

V
AGREEMENT [PERSON NUMBER 3rd sing] → {like}

V
AGREEMENT [PERSON NUMBER 3rd sing] → {likes}

NP
AGREEMENT [PERSON NUMBER 3rd sing] → {fish, rivers, pools, they}

NP
AGREEMENT [PERSON NUMBER 3rd sing] → {it, fish, Peter}
```
They likes Peter — UNIFICATION FAILS BECAUSE OF CO-INDEXATION

- S
- VP
- V
- NP

AGREEMENT → NP

AGREEMENT → VP

AGREEMENT → V

AGREEMENT → NP

AGREEMENT → NP

AGREEMENT → NP
Subcategorization is captured by the feature constraints

\[
\begin{align*}
S & \rightarrow \begin{cases} 
\text{NP} & \text{AGREEMENT}, \\
\text{VP} & \text{AGREEMENT,}
\end{cases} \\
\text{VP} & \rightarrow \begin{cases} 
\text{V}, \\
\text{NP} & \text{AGREEMENT,}
\end{cases}
\end{align*}
\]
Subcategorization is captured by the feature constraints

\[
\begin{align*}
[S \quad \text{HEAD} \quad 1] & \rightarrow \quad [NP \quad \text{HEAD} \quad \text{AGREEMENT} \quad 3] \quad [VP \quad \text{HEAD} \quad \text{AGREEMENT} \quad 3] \\
[VP \quad \text{HEAD} \quad 1 \quad \text{SUBJ} \quad 3] & \rightarrow \quad [V \quad \text{HEAD} \quad 1 \quad \text{OBJ} \quad 2 \quad \text{SUBJ} \quad 3] \\
\langle \text{can,} \quad \text{OBJ} \quad \text{SUBJ} \quad \text{NP} \quad \text{HEAD} \quad \text{Agreement} \quad \rangle & \quad [VP \quad \text{HEAD} \quad ] \\
\end{align*}
\]
Subcategorization is captured by the feature constraints:

\[
\begin{align*}
S & \rightarrow NP \text{ HEAD} 1 \text{ AGREEMENT} 3 \text{ VP} \\
& \rightarrow V \text{ HEAD} 1 \text{ OBJ} 2 \text{ SUBJ} 3 \\
& \rightarrow \text{ can, VP} \text{ HEAD} 2 \text{ NP} \text{ HEAD} 1 \text{ OBJ} 2 \text{ SUBJ} 3
\end{align*}
\]
Alternatively use **unification as the parsing operation** instead of just for feature checking:

- $X_0 \rightarrow X_1 X_2$
  - $< X_1 \text{ HEAD AGREEMENT } > = < X_2 \text{ HEAD AGREEMENT } >$
  - $< X_0 \text{ HEAD } > = < X_1 \text{ HEAD } >$

- $X_0 \rightarrow X_1 X_2$
  - $< X_0 \text{ HEAD } > < X_1 \text{ HEAD } >$
  - $< X_2 \text{ CAT } > = PP$

- $X_0 \rightarrow X_1 \text{ and } X_2$
  - $< X_0 \text{ CAT } > < X_1 \text{ CAT } >$
  - $< X_1 \text{ CAT } > < X_2 \text{ CAT } >$
Alternatively use **unification as the parsing operation** instead of just for feature checking:

- $X_0 \rightarrow X_1 X_2$
  
  \[
  \begin{align*}
  &< X_1 \text{ HEAD AGREEMENT } >= < X_2 \text{ HEAD AGREEMENT } > \\
  &< X_0 \text{ HEAD } >= < X_1 \text{ HEAD } >
  \end{align*}
  \]

- $X_0 \rightarrow X_1 X_2$
  
  \[
  \begin{align*}
  &< X_0 \text{ HEAD } >= < X_1 \text{ HEAD } > \\
  &< X_2 \text{ CAT } >= PP
  \end{align*}
  \]

- $X_0 \rightarrow X_1$ and $X_2$
  
  \[
  \begin{align*}
  &< X_0 \text{ CAT } >= < X_1 \text{ CAT } > \\
  &< X_1 \text{ CAT } >= < X_2 \text{ CAT } >
  \end{align*}
  \]
Alternatively use **unification as the parsing operation**

Alternatively we can use **unification as the parsing operation** instead of just for feature checking:

- $X_0 \rightarrow X_1 X_2$
  - $< X_1 \text{ HEAD AGREEMENT } > = < X_2 \text{ HEAD AGREEMENT }>$
  - $< X_0 \text{ HEAD } > = < X_1 \text{ HEAD } >$

- $X_0 \rightarrow X_1 X_2$
  - $< X_0 \text{ HEAD } > < X_1 \text{ HEAD } >$
  - $< X_2 \text{ CAT } > = PP$

- $X_0 \rightarrow X_1$ and $X_2$
  - $< X_0 \text{ CAT } > < X_1 \text{ CAT } >$
  - $< X_1 \text{ CAT } > < X_2 \text{ CAT } >$
Lexical AVMs may be derived through unification

- We have assumed we have a lexicon entry for all the inflected forms of a word.
- With a morphological analysis step we can return a word its stem and affixes and then build the AVM from the pieces: \( \text{foxes} \rightarrow \text{fox}^\text{s} \)

\[
\begin{align*}
\langle \text{s}, \quad \text{HEAD} \quad \begin{bmatrix} \text{N} \\ \text{AGREEMENT} \quad \text{pl} \end{bmatrix} \rangle \\
\langle \text{fox}, \quad \text{HEAD} \quad \begin{bmatrix} \text{N} \\ \text{AGREEMENT} \quad [] \end{bmatrix} \rangle
\end{align*}
\]
Lexical AVMs may be derived through unification

- We have assumed we have a lexicon entry for all the inflected forms of a word.
- With a morphological analysis step we can return a word its stem and affixes and then build the AVM from the pieces: foxes → fox^s

\[
\begin{align*}
\langle \, ^s, & \left[ \begin{array}{c}
{\text{HEAD}} \\
{\text{AGREEMENT}} & \text{pl}
\end{array} \right] \rangle \\
\langle \, \text{fox}, & \left[ \begin{array}{c}
{\text{HEAD}} \\
{\text{AGREEMENT}} & \text{[]}
\end{array} \right] \rangle \\
\left[ \begin{array}{c}
{\text{HEAD}} \\
{\text{AGREEMENT}} & \text{pl}
\end{array} \right] & \sqcup \left[ \begin{array}{c}
{\text{HEAD}} \\
{\text{AGREEMENT}} & \text{[]}
\end{array} \right] = \left[ \begin{array}{c}
{\text{HEAD}} \\
{\text{AGREEMENT}} & \text{pl}
\end{array} \right]
\end{align*}
\]
Lexical AVMs may be derived through unification

- We have assumed we have a lexicon entry for all the inflected forms of a word.
- With a morphological analysis step we can return a word its stem and affixes and then build the AVM from the pieces: $\text{foxes} \rightarrow \text{fox}^\wedge s$

$$\langle ^\wedge s, \begin{bmatrix} \text{HEAD} \\ \text{N} \\ \text{AGREEMENT} \\ pl \end{bmatrix} \rangle$$

$$\langle \text{fox}, \begin{bmatrix} \text{HEAD} \\ \text{N} \\ \text{AGREEMENT} \end{bmatrix} \rangle$$

$$\begin{bmatrix} \text{HEAD} \\ \text{N} \\ \text{AGREEMENT} \\ pl \end{bmatrix} \sqcup \begin{bmatrix} \text{HEAD} \\ \text{N} \\ \text{AGREEMENT} \end{bmatrix} = \begin{bmatrix} \text{HEAD} \\ \text{N} \\ \text{AGREEMENT} \\ pl \end{bmatrix}$$
Focus on adequacy for a wide range of languages as well as tractable for parsing.

Examples include **Lexical Functional Grammar, LFG** (Bresnan and Kaplan) and **Head-driven Phrase Structure Grammar, HPSG** (Pollard and Sag).

Grammars tend to incorporate aspects of morphology, syntax and compositional semantics:

If you are interested see: [http://www.delph-in.net](http://www.delph-in.net)