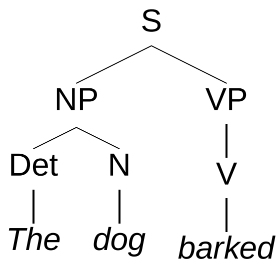


$$\begin{bmatrix} \texttt{LTOP} & h_1 \\ \texttt{INDEX} & e_2 \\ \texttt{RELS} & \left\langle \begin{bmatrix} \textit{prpstn\_m\_rel} & \\ \texttt{LBL} & h_1 \\ \texttt{MARG} & h_3 \end{bmatrix} \begin{bmatrix} \textit{def\_q\_rel} & \\ \texttt{LBL} & h_4 \\ \texttt{ARG0} & x_5 \\ \texttt{RSTR} & h_6 \\ \texttt{BODY} & h_7 \end{bmatrix} \begin{bmatrix} \textit{“dog\_n\_rel”} & \\ \texttt{LBL} & h_8 \\ \texttt{ARG0} & x_5 \end{bmatrix} \begin{bmatrix} \textit{“bark\_v\_rel”} & \\ \texttt{LBL} & h_9 \\ \texttt{ARG0} & e_2 \\ \texttt{ARG1} & x_5 \end{bmatrix} \right\rangle \\ \texttt{HCONS} & \langle h_3 =_q h_9,\, h_6 =_q h_8 \rangle \end{bmatrix}$$

Algorithms for AI and NLP (INF4820 — PCFGs)

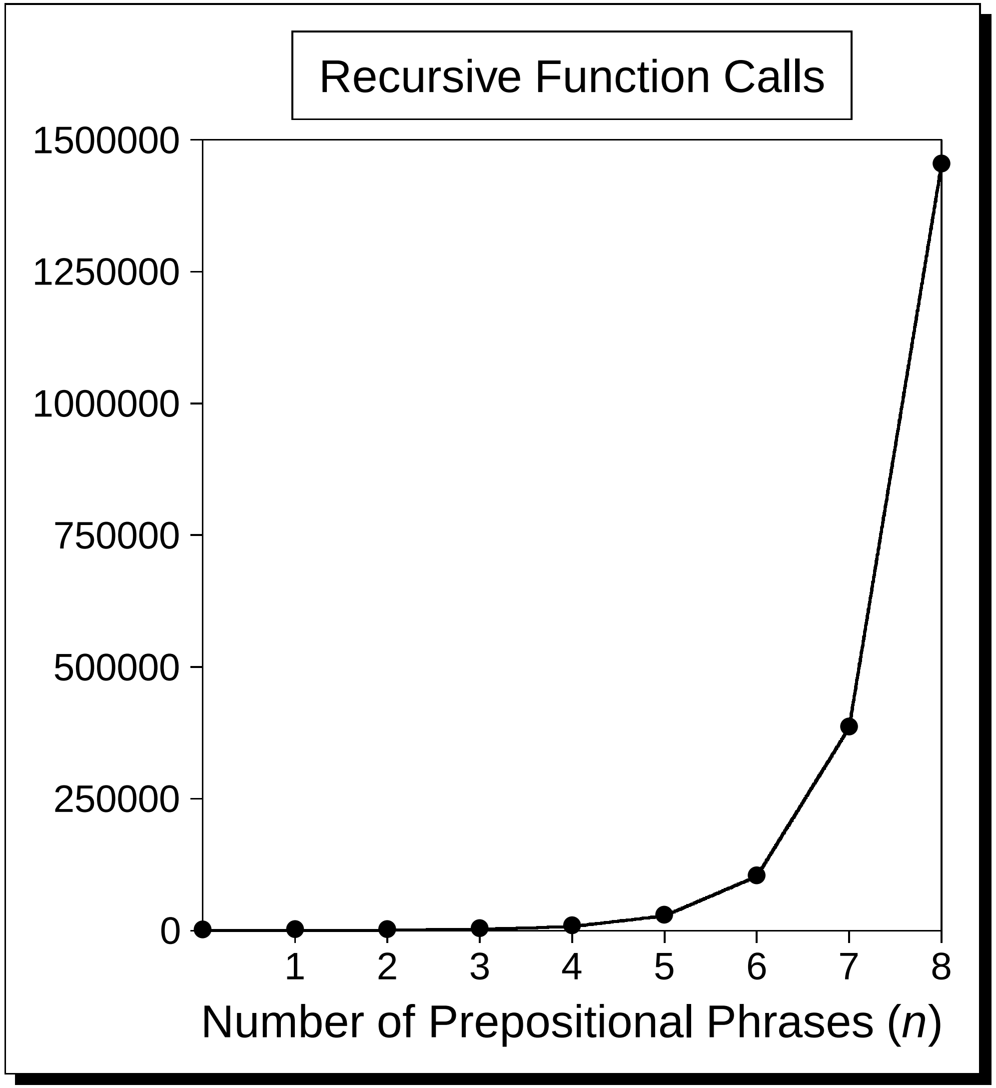
P(S $\longrightarrow$ *NP VP)* = *1.0; P(NP* $\longrightarrow$ *Det N)* = *0.6*

Stephan Oepen and Jan Tore Lønning

Universitetet i Oslo

`{ oe | jtl }@ifi.uio.no`

Quantifying the Complexity of the Parsing Task



Kim adores snow (*in Oslo*)^n

n	trees	calls
0	1	46
1	2	170
2	5	593
3	14	2,093
4	42	7,539
5	132	27,627
6	429	102,570
7	1430	384,566
8	4862	1,452,776
⋮	⋮	⋮

Chart Parsing — Specialized Dynamic Programming

Basic Notions

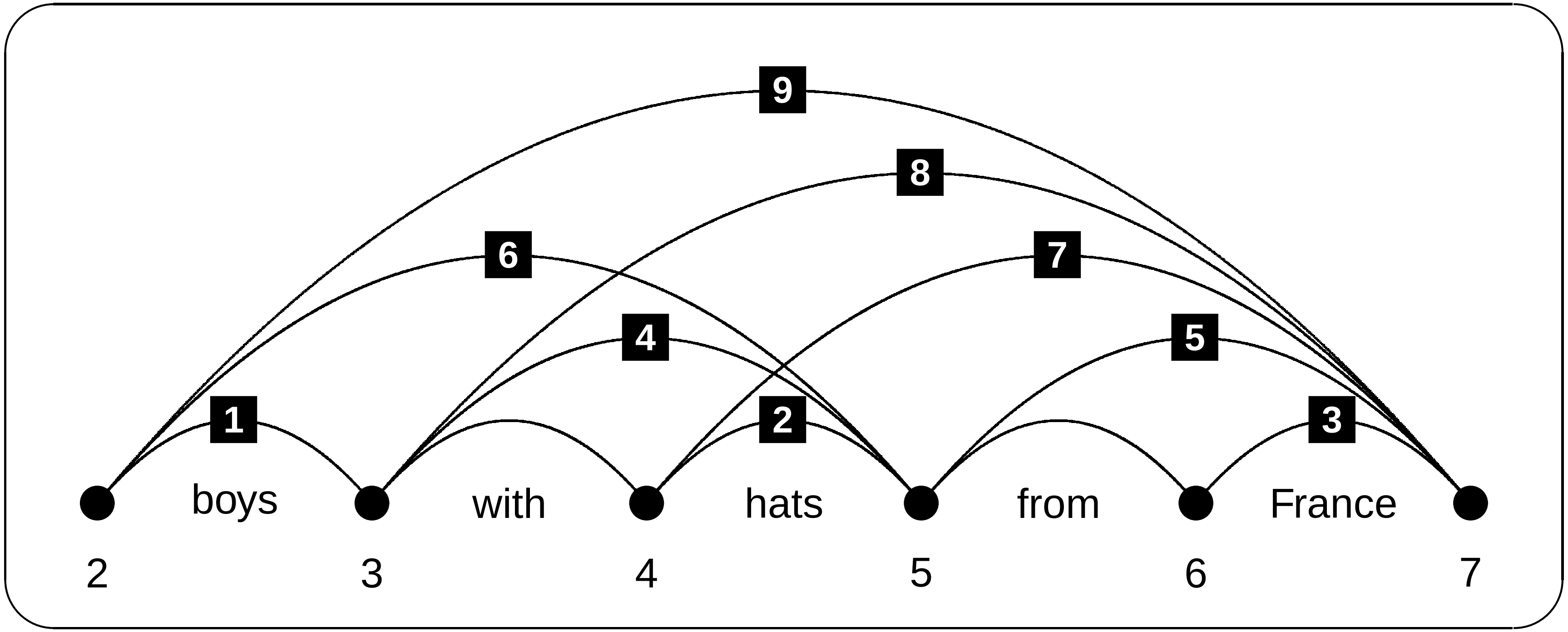
- Use *chart* to record partial analyses, indexing them by string positions;
- count inter-word vertices; CKY: chart row is *start*, column *end* vertex;
- treat multiple ways of deriving the same category for some substring as *equivalent*; pursue only once when combining with other constituents.

Key Benefits

- Dynamic programming (memoization): avoid recomputation of results;
- efficient indexing of constituents: no search by start or end positions;
- compute *parse forest* with exponential 'extension' in *polynomial* time.

Bounding Ambiguity — The Parse Chart

- For many substrings, more than one way of deriving the same category;
- NPs: **1** | **2** | **3** | **6** | **7** | **9**; PPs: **4** | **5** | **8**; **9** $\equiv$ **1** + **8** | **6** + **5**;
- *parse forest* — a single item represents multiple trees [Billot & Lang, 89].



The CKY (Cocke, Kasami, & Younger) Algorithm

for $(0 \leq i < |\textit{input}|)$ do
  $\textit{chart}_{[i,i+1]} \leftarrow \{\alpha \,|\, \alpha \rightarrow \textit{input}_i \in P\}$;
for $(1 \leq l < |\textit{input}|)$ do
  for $(0 \leq i < |\textit{input}| - l)$ do
    for $(1 \leq j \leq l)$ do
      if $(\alpha \rightarrow \beta_1\,\beta_2 \in P \wedge \beta_1 \in \textit{chart}_{[i,i+j]} \wedge \beta_2 \in \textit{chart}_{[i+j,i+l+1]})$ then
        $\textit{chart}_{[i,i+l+1]} \leftarrow \textit{chart}_{[i,i+l+1]} \cup \{\alpha\}$;

$[0,2] \leftarrow [0,1] + [1,2]$
$\ldots$
$[0,5] \leftarrow [0,1] + [1,5]$
$[0,5] \leftarrow [0,2] + [2,5]$
$[0,5] \leftarrow [0,3] + [3,5]$
$[0,5] \leftarrow [0,4] + [4,5]$

	1	2	3	4	5
0	NP		S		S
1		V	VP		VP
2			NP		NP
3				P	PP
4					NP

Generalized Chart Parsing

- The parse *chart* is a two-dimensional matrix of *edges* (aka chart items);
- an edge is a (possibly partial) rule instantiation over a substring of input;
- the chart indexes edges by start and end string position (aka vertices);
- dot in rule RHS indicates degree of completion: $\alpha \rightarrow \beta_1 ... \beta_{i-1} \bullet \beta_i ... \beta_n$
- *active* edges (aka *incomplete* items) — partial RHS: $[1, 2, \mathrm{VP} \rightarrow \mathrm{V} \bullet \mathrm{NP}]$;
- *passive* edges (aka *complete* items) — full RHS: $[1, 3, \mathrm{VP} \rightarrow \mathrm{V}\ \mathrm{NP} \bullet]$;

The Fundamental Rule

$$[i,\ j,\ \alpha \rightarrow \beta_1 ... \beta_{i-1} \bullet \beta_i ... \beta_n] + [j,\ k,\ \beta_i \rightarrow \gamma^+ \bullet]$$
$$\mapsto [i,\ k,\ \alpha \rightarrow \beta_1 ... \beta_i \bullet \beta_{i+1} ... \beta_n]$$

Backpointers: Recording the Derivation History

	0	1	2	3
0	2: S $\rightarrow$ • NP VP 1: NP $\rightarrow$ • NP PP 0: NP $\rightarrow$ • kim	10: S $\rightarrow$ 8 • VP 9: NP $\rightarrow$ 8 • PP 8: NP $\rightarrow$ kim •		17: S $\rightarrow$ 8 15 •
1		5: VP $\rightarrow$ • VP PP 4: VP $\rightarrow$ • V NP 3: V $\rightarrow$ • adored	12: VP $\rightarrow$ 11 • NP 11: V $\rightarrow$ adored •	16: VP $\rightarrow$ 15 • PP 15: VP $\rightarrow$ 11 13 •
2			7: NP $\rightarrow$ • NP PP 6: NP $\rightarrow$ • snow	14: NP $\rightarrow$ 13 • PP 13: NP $\rightarrow$ snow •
3				

- Use edges to record derivation trees: backpointers to daughters;
- a single edge can represent multiple derivations: backpointer sets.

Ambiguity Packing in the Chart

General Idea

- Maintain only one edge for each α from i to j (the 'representative');
- record alternate sequences of daughters for α in the representative.

Implementation

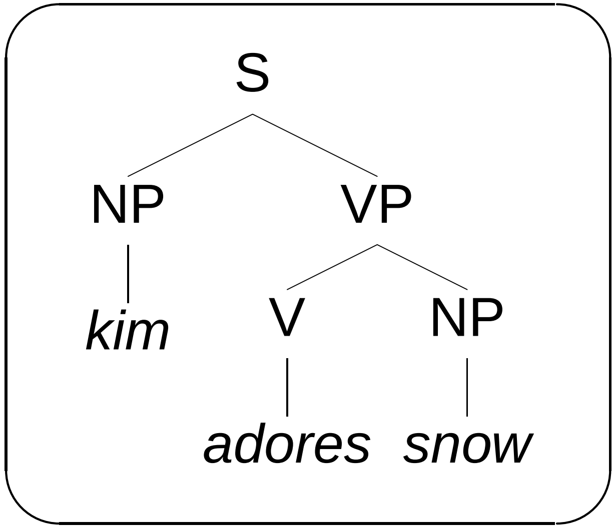
- Group passive edges into *equivalence classes* by identity of α, i, and j;
- search chart for existing equivalent edge (h, say) for each new edge e;
- when h (the 'host' edge) exists, *pack* e into h to record equivalence;
- e *not* added to the chart, no derivations with or further processing of e;

$\rightarrow$ *unpacking* multiply out all alternative daughters for all result edges.

Background: Trees as Bracketed Sequences

- Trees can be encoded as sequences (*dominance* plus *precedence*):

```
(S (NP kim)
   (VP (V adored)
       (NP snow)))
```



- the `first()` element (at each level) represents the tree root (or mother);
- all other elements (i.e. the `rest()`) correspond to immediate daughters.

Ambiguity Resolution Remains a (Major) Challenge

The Problem

- With broad-coverage grammars, even moderately complex sentences typically have multiple analyses (tens or hundreds, rarely thousands);
- unlike in grammar writing, exhaustive parsing is useless for applications;
- identifying the 'right' (intended) analysis is an 'AI-complete' problem;
- inclusion of (non-grammatical) sortal constraints is generally undesirable.

Typical Approaches

- Design and use statistical models to select among competing analyses;
- for string S, some analyses T_i are more or less likely: maximize $P(T_i|S)$;

$\rightarrow$ Probabilistic Context Free Grammar (PCFG) is a CFG plus probabilities.

Probability Theory and Linguistics?

The most important questions of life are, for the most part, really only questions of probability. (Pierre-Simon Laplace, 1812)

Probability Theory and Linguistics?

The most important questions of life are, for the most part, really only questions of probability. (Pierre-Simon Laplace, 1812)

Special wards in lunatic asylums could well be populated with mathematicians who have attempted to predict random events from finite data samples. (Richard A. Epstein, 1977)

Probability Theory and Linguistics?

The most important questions of life are, for the most part, really only questions of probability. (Pierre-Simon Laplace, 1812)

Special wards in lunatic asylums could well be populated with mathematicians who have attempted to predict random events from finite data samples. (Richard A. Epstein, 1977)

But it must be recognized that the notion 'probability' of a sentence is an entirely useless one, under any known interpretation of this term. (Noam Chomsky, 1969)

Probability Theory and Linguistics?

The most important questions of life are, for the most part, really only questions of probability. (Pierre-Simon Laplace, 1812)

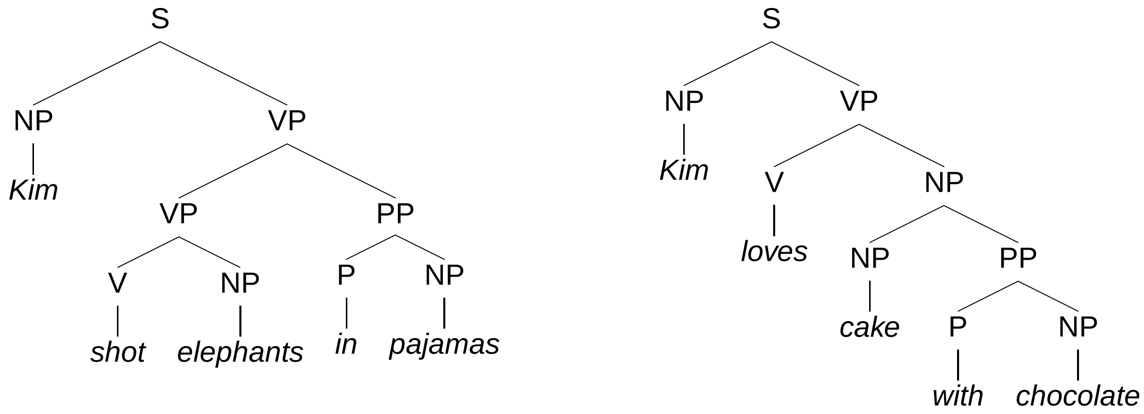
Special wards in lunatic asylums could well be populated with mathematicians who have attempted to predict random events from finite data samples. (Richard A. Epstein, 1977)

But it must be recognized that the notion 'probability' of a sentence is an entirely useless one, under any known interpretation of this term. (Noam Chomsky, 1969)

Every time I fire a linguist,
system performance improves. (Fredrick Jelinek, 1980s)

Probabilistic Context-Free Grammars

A (Simplified) PCFG Estimation Example



P(RHS|LHS)	CFG Rule
	S $\longrightarrow$ NP VP
	VP $\longrightarrow$ VP PP
	VP $\longrightarrow$ V NP
	PP $\longrightarrow$ P NP
	NP $\longrightarrow$ NP PP
	VP $\longrightarrow$ V

- Estimate rule probability from observed distribution;

$\longrightarrow$ conditional probabilities:

$$P(\mathrm{RHS}|\mathrm{LHS}) = \frac{C(\mathrm{LHS}, \mathrm{RHS})}{C(\mathrm{LHS})}$$

Formally: Probabilistic Context-Free Grammars

- Formally, a *context-free grammar* (CFG) is a quadruple: $\langle C, \Sigma, P, S \rangle$

...

- P is a set of category rewrite rules (aka *productions*), each with a conditional probability P(RHS|LHS), e.g.

...

NP $\rightarrow$ Kim [0.6]
NP $\rightarrow$ snow [0.4]

...

- for each rule '$\alpha \rightarrow \beta_1, \beta_2, ..., \beta_n$' $\in P$: $\alpha \in C$ and $\beta_i \in C \cup \Sigma$; $1 \leq i \leq n$;

...

- for each $\alpha \in C$, the probabilities of all rules R '$\alpha \rightarrow$ ...' must sum to 1.

Limitations of Context-Free Grammar

Agreement and Valency (For Example)

That dog barks.

**That dogs barks.*

**Those dogs barks.*

The dog chased a cat.

**The dog barks a cat.*

**The dog chased.*

**The dog chased a cat my neighbours.*

The cat was chased by a dog.

**The cat was chased of a dog.*

...

Unification-Based Grammar: Structured Categories

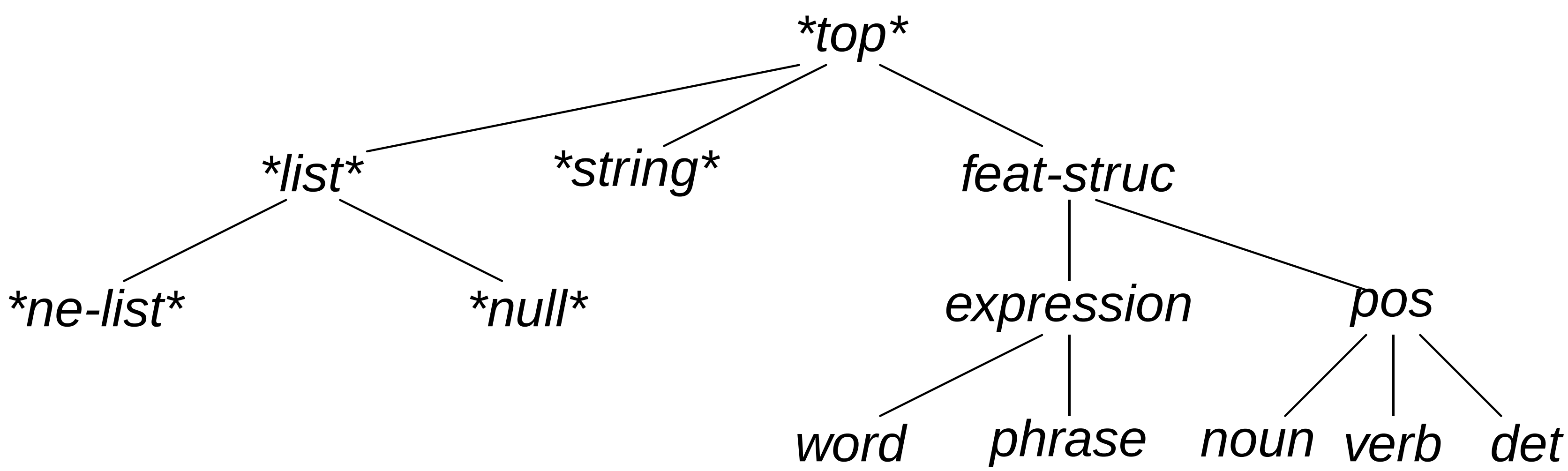
- All (constituent) categories in the grammar are *typed feature structures*;
- feature structures are recursive, record-like objects: attribute – value sets;
- typing very similar to OO programming: a multipe-inheritance hierarchy;
- specific TFS configurations may correspond to ‘traditional’ categories;
- labels like ‘S’ or ‘NP’ are mere abbreviations, not elements of the theory.

$$\mathit{word}\begin{bmatrix} \mathrm{HEAD} & \mathit{noun} \\ \mathrm{SPR} & \left\langle \begin{bmatrix} \mathrm{HEAD} & \mathit{det} \end{bmatrix} \right\rangle \\ \mathrm{COMPS} & \langle\rangle \end{bmatrix} \qquad \mathit{phrase}\begin{bmatrix} \mathrm{HEAD} & \mathit{verb} \\ \mathrm{SPR} & \langle\rangle \\ \mathrm{COMPS} & \langle\rangle \end{bmatrix} \qquad \mathit{phrase}\begin{bmatrix} \mathrm{HEAD} & \mathit{verb} \\ \mathrm{SPR} & \left\langle \begin{bmatrix} \mathrm{HEAD} & \mathit{noun} \end{bmatrix} \right\rangle \\ \mathrm{COMPS} & \langle\rangle \end{bmatrix}$$

‘N’ **‘S’** **‘VP’**

The Type Hierarchy: Fundamentals

- Types 'represent' groups of entities with similar properties ('classes');
- types ordered by specificity: subtypes inherit properties of (all) parents;
- type hierarchy determines which types are compatible (and which not).



Typed Feature Structure Subsumption

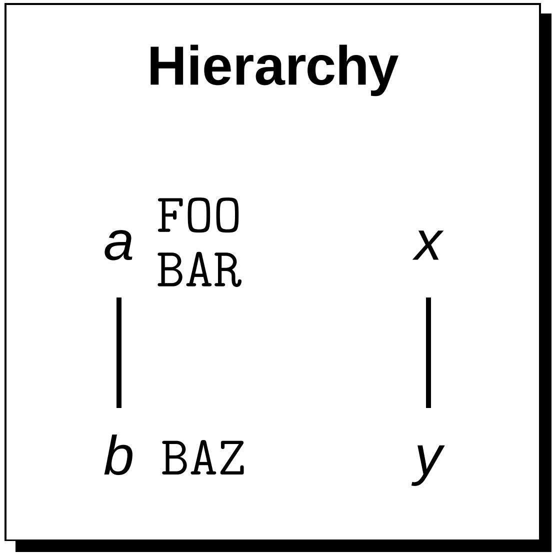
- Typed feature structures can be partially ordered by information content;
- a more general structure is said to *subsume* a more specific one;
- $\textit{*top*}\begin{bmatrix}\ \end{bmatrix}$ is the most general feature structure (while $\perp$ is inconsistent);
- $\sqsubseteq$ ('square subset or equal') conventionally used to depict subsumption.

Feature structure F subsumes feature structure G ($F \sqsubseteq G$) iff: (1) if path p is defined in F then p is also defined in G and the type of the value of p in F is a supertype or equal to the type of the value of p in G, and (2) all paths that are reentrant in F are also reentrant in G.

Feature Structure Subsumption: Examples

$$\text{TFS}_1: \; {}_{a}\!\begin{bmatrix} \text{FOO } x \\ \text{BAR } x \end{bmatrix} \qquad \text{TFS}_2: \; {}_{a}\!\begin{bmatrix} \text{FOO } x \\ \text{BAR } y \end{bmatrix}$$

$$\text{TFS}_3: \; {}_{b}\!\begin{bmatrix} \text{FOO } y \\ \text{BAR } x \\ \text{BAZ } x \end{bmatrix} \qquad \text{TFS}_4: \; {}_{a}\!\begin{bmatrix} \text{FOO } \boxed{1}\, x \\ \text{BAR } \boxed{1} \end{bmatrix}$$



Feature structure F subsumes feature structure G ($F \sqsubseteq G$) iff: (1) if path p is defined in F then p is also defined in G and the type of the value of p in F is a supertype or equal to the type of the value of p in G, and (2) all paths that are reentrant in F are also reentrant in G.

Typed Feature Structure Unification

- Decide whether two typed feature structures are mutually compatible;
- determine combination of two TFSs to give the most general feature structure which retains all information which they individually contain;
- if there is no such feature structure, unification fails (depicted as $\perp$);
- unification *monotonically* combines information from both 'input' TFSs;
- *relation to subsumption* the unification of two structures *F* and *G* is the most general TFS which is subsumed by both *F* and *G* (if it exists).
- $\sqcap$ ('square set intersection') conventionally used to depict unification.

Typed Feature Structure Unification: Examples

TFS$_1$: $\; {}_{a}\begin{bmatrix}\texttt{FOO}\ x \\ \texttt{BAR}\ x\end{bmatrix}$

TFS$_2$: $\; {}_{a}\begin{bmatrix}\texttt{FOO}\ x \\ \texttt{BAR}\ y\end{bmatrix}$

TFS$_3$: $\; {}_{b}\begin{bmatrix}\texttt{FOO}\ y \\ \texttt{BAR}\ x \\ \texttt{BAZ}\ x\end{bmatrix}$

TFS$_4$: $\; {}_{a}\begin{bmatrix}\texttt{FOO}\ \boxed{1}\, x \\ \texttt{BAR}\ \boxed{1}\end{bmatrix}$

Hierarchy

$$\begin{array}{lll} a & \begin{array}{l}\texttt{FOO}\\ \texttt{BAR}\end{array} & \qquad x \\ | & & \qquad | \\ b & \texttt{BAZ} & \qquad y \end{array}$$

$$\text{TFS}_1 \sqcap \text{TFS}_2 \equiv \text{TFS}_2 \qquad \text{TFS}_1 \sqcap \text{TFS}_3 \equiv \text{TFS}_3 \qquad \text{TFS}_3 \sqcap \text{TFS}_4 \equiv {}_{b}\begin{bmatrix}\texttt{FOO}\ \boxed{1}\, y \\ \texttt{BAR}\ \boxed{1} \\ \texttt{BAZ}\ x\end{bmatrix}$$