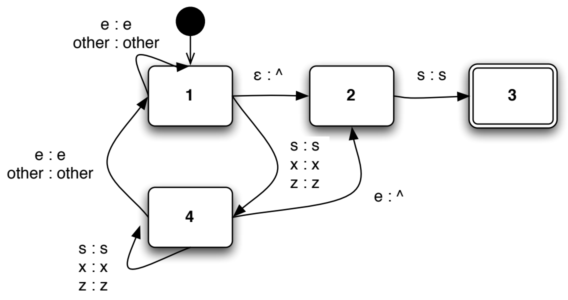
## Morphology

Words made up of *morphemes*. Morphemes which only occur with others are *affixes*. Words are *stems* along with some affixes. Have *infixes, circumfixes, prefixes and postfixes*. *Inflectional* morphology sets value of slots in some paradigm, and concerns e.g. tense, aspect, number, gender. *Derivational* morphology doesn't fit into neat paradigms (e.g. un-, anti-, re-), and hence is not similarly productive. Stems and affixes can be ambiguous (e.g. unionised).

Spelling rules show a mapping rule and a context in which the rule may be applied. The symbol ^ represents an affix boundary. An example is: $\epsilon \rightarrow e / \left\{ \begin{array}{c} s \\ x \\ z \end{array} \right\} \hat{}\underline{\ } s$

You can use a *full-form lexicon* to list all inflected forms and then treat derivational morphology as non-productive. Morphological analysis needs to know affixes, irregular forms and stems with syntactic categories. In English we can associate irregularity with words rather than meanings almost always (except e.g. hanged him / hung it out).

Analysis is typically done with *finite state transducers* (FSAs with symbols on transitions). For example, to recognize the affix "s":



## Part Of Speech Tagging

A *corpus* is a collection of text. A balanced corpus is a corpus gathered from many genres.

N-gram models are a type of Markov chain. A bi-gram model assigns probability to a word based only on the preceding word. To allow for sparse data we use *smoothing*: make an assumption about the probability of infrequent events, e.g. add-one smoothing.

A POS tagger resolves lexical ambiguities to give the most likely set of sentence tags (e.g. verb, singular noun, plural noun). This tagging is not necessarily globally coherent. A possible simple algorithm is one based on bi-grams, and $P(T|W) = \frac{P(T)P(W|T)}{P(W)}$ with $P(T) \approx P(t_i|t_{t-1})$ and $P(W|T) \approx P(w_i|t_i)$. Unknown words may be handled by guessing from a weighted set of open-class tags or fallback on morphological analysis (e.g. looking for -ed).

Evaluation is done by the percentage of correct tags. Typically 90% of the data is used to train and 10% for evaluation. Inter-annotator agreement is 96% and the baseline of choosing the most probable tag for a word is 90%. Most POS taggers have very uneven error distributions.

## Parsing And Generation

Two grammars are *weakly equivalent* if they generate the same strings, and *strongly equivalent* if they generate the same bracketings as well (constituents are the same).

Grammars in which all non-terminal daughters are the leftmost daughter are called *left associative*. Likewise grammars with rightmost terminals are *right associative*.

Grammars are typically both *lexically ambiguous* (some terminals correspond to multiple non-terminals) and *structurally ambiguous* (different bracketings).

### Chart Parsers

We can take advantage of the context-free nature of the grammar to record rules that we previously applied. We do this via a *chart*, which is a list of *edges*. Each edge has the structure [*id, left vertex, right vertex, mother category, daughters*]. Each vertex is an integer representing a point in the input string (e.g. between each word). The *mother category* is the rule which created the edge, and *daughters* are the list of daughter edges for this particular rule application. Parsing proceeds as:

**Parse:**

1. Create chart
2. For each *word* in the *input*, let *from* be the left vertex, *to* be the right vertex and *daughters* be *[word]*
   (a) For each *category* associated with *word*
      i. **Add new edge** *from, to, category, daughters*

**Add new edge** *from, to, category, daughters***:**

1. Put $[id, from, to, category, daughters]$ into the chart
2. For each rule $lhs \rightarrow cat_1 \ldots cat_{n-1}, category$
   (a) Find sets of contiguous edges $[id_1, from_1, to_1, cat_1, daughters_1] \ldots [id_{n-1}, from_{n-1}, from$ so that $to_i = from_{i+1}$
   (b) For each list of edges, **Add new edge** $from_1$,*to, lhs,* $(id_1 \ldots id)$

The algorithm is complete but may not terminate in the case of recursive rules.

We can make the algorithm faster by sharing edges. If we are about to add an edge that differs from an existing one only in the daughters, we can just modify the existing daughters to be the union of the two others. This is called *packing.* Note however that producing the output is still an exponential time process.

*Active chart parsers* have edges that record the input they expect (i.e. a non-terminal symbol) in addition to daughters that have been seen. Each time a passive edge (one with no more input expected, e.g. for terminals) is added, the active edges are searched to see if it can be completed with the passive one.

## Parsing Comments

The parsing search space can be ordered by adding an explicit *schedule* which tells us which rules to try first. Indeed, very low probability rules can be excluded from consideration by this process (known as *beam search*).

FSAs cannot in general be used for natural languages due to the presence of centre embedding. However, the human brain has a limit to the amount of centre-embedding it can deal with.. this could open the door to FSA analysis. However, FSA techniques involve much redundancy and we cannot build up good semantic representations due to the lack of internal structure.

# Constraint Based Grammars

These can encode e.g. subject-verb agreement and subcategorization (verb arity) easily. We typically do this with *feature structures.* These are single-rooted directed acyclic graphs with arcs labeled by *features* and terminal nodes associated with values. No feature may appear on two or more edges leading out of a node. An *atomic valued* feature points to a terminal node, *complex valued* ones do not. A sequence of features is a *path.* A feature structure FS1 subsumes FS2 if and only if every path in FS1 exists in FS2 which preserves path atomic values, and every pair of paths which lead to the same node in FS1 (are *reentrant*) are also reentrant in FS2. Unification is defined by the most general FS which subsumes both FS1 and FS2.

Rules are encoded as FSs with features *MOTHER, DTR1, DTR2* etc. For example:

$$\begin{bmatrix} MOTHER & \begin{bmatrix} CAT & \mathbf{VP} \\ AGR & \mathbf{1} \end{bmatrix} \\ DTR1 & \begin{bmatrix} CAT & \mathbf{V} \\ AGR & \mathbf{1} \end{bmatrix} \\ DTR2 & \begin{bmatrix} CAT & \mathbf{NP} \\ AGR & [] \end{bmatrix} \end{bmatrix}$$

Complicated rules will typically have a *HEAD* feature which contains information shared between lexical entries and phrases of the same category (e.g. the category itself and the verb agreement), whereas things like object and subject are ancillary (in the case of intransitive verbs only one of these will be filled). For example:

$$\begin{bmatrix} HEAD & \mathbf{1} \\ OBJ & \textbf{filled} \\ SUBJ & \textbf{filled} \end{bmatrix} \rightarrow \mathbf{2} \begin{bmatrix} HEAD & \begin{bmatrix} AGR & \mathbf{3} \end{bmatrix} \\ OBJ & \textbf{filled} \\ SUBJ & \textbf{filled} \end{bmatrix}, \begin{bmatrix} HEAD & \mathbf{1} \\ OBJ & \\ SUBJ & \end{matrix}$$

Note that when defining FS grammars certain forms may occur over and over again (e.g. the FS for particular words). This is solved with *templates.* Note also that inflectional morphology can be encoded by considering stems and affixes as FSs. This will help e.g. exclude fee-ed as a valid analysis for feed.

## Parsing With Feature Structures

Such a system will build a parse structure which is subsumed by all the constraints encoded by the feature structures. Usually chart parsing is used. When we need to check a grammar rule matches an edge, the feature structures in the edges of the chart that correspond to the possible daughters are copied along with the grammar rule feature structure. The copied daughters are unified with the daughter positions in the copy of the rule and if successful the copied structure is associated with a new chart edge. Copying is necessary because unification pollutes the rules with extra information which may not be applicable to all uses.

Copying can be expensive but is improved by pretesting for likely unification, sharing unchanged parts of the FS and using linguistic locality in rule construction. Packing is poor since structures are rarely identical, but can be improved by testing if an edge can be packed using subsumption.

# Semantics

This can be done by augmenting your FS grammars with a *SEM* feature which contains an encoding of predicates such as:

$$SEM \begin{bmatrix} PRED & \textbf{and} \\ ARG1 & \mathbf{4} \\ ARG2 & \mathbf{5} \end{bmatrix}$$

These features can also contain *INDEX* subfeatures, which are the characteristic variables of the noun. It is used to "reach inside" the predicates which are applied in the noun FSs. A thus constructed semantic representation is a *logical form.* This is typically first order predicate calculus (FOPC) but this doesn't encode ambiguity well (not probabilistic and forces particular scopal relationships).

*Meaning postulates* are inference rules relating open classes. However, even if we constrain these to implication rather than (philosophically tricky) equality, it is hard to acquire these postulates and control the inferences they allow.

## Semantic Relationships

*Hyponomy* is the is-a relationship. Classically we may have a tree structure (a taxonomy), though you can allow multiple inheritance. This only really makes sense for concrete nouns, and dealing with quantization can be tricky. It can be used for semantic classification (e.g. the object of eat has to be edible), shallow inference (X murdered Y implies X killed Y), for coarse grained statistics (classify on hyponyms instead of base words), machine translation (substitute hypernym for untranslatable words) and information retrieval query expansion.

*Meronomy* is the part-of relationship.

*Synonymy* is the same-meaning relationship.

*Antonymy* is the opposite-meaning relationship. Only relevant for some adjective classes.

*Polysymy* is the state of a word having more than one sense (e.g. dance, which can be a noun or a verb).

*Homonymy* refers to polysemous words whose senses are entirely unrelated (e.g. bank).

## Word Sense Disambiguation

This selects the correct sense of a word in some context. Classically done by hand crafting rules, but nowadays sense frequency, collocations and selectional restrictions are used with machine learning to avoid this. Success rates can be as high as 95% for homonyms, but may be 70% for general polysyms.

*Collocations* are groups of multiple words that occur together more often than chance would suggest. This can be used for WSD (e.g. striped bass vs. bass guitars).

Yarowsky's algorithm for WSD is a machine learning technique exploiting collocates. This proceeds by:

1. Identify all examples of the word to be disambiguated and store their contexts.
2. Identify some seeds which reliable disambiguate a few of these uses according to our initial knowledge: tag these uses with the right class and count the rest as residual.
3. Then until we have convergence:
   (a) Train a decision list classifier on the examples we have identified the class of (ordered list of criterion tagged with reliability, may include criterion like some concrete surrounding words or rules like "animal" being within 10 words).
   (b) Apply the classifier to the training set and add examples tagged with greater than a threshold reliability to the appropriate classes.

Yarowsky also demonstrated the one sense per discourse principle, which can be used to refine the algorithm above.

# Discourse

This refers to relationships between phrases, which may be e.g. *explanatory* or *narratory*. Discourses must have connectivity to be coherent, which can be hard for strategic generation algorithms. Things affecting discourse interpretation include:

- Coherence assumptions(e.g. John likes Bill. He gave him an expensive present).
- Cue phrases (e.g. and).
- Punctuation (e.g. parentheses enclosing an explanation, lists encoding narration).
- Real world content (e.g. Max fell. John pushed him as he lay on the ground).
- Tense and aspect (e.g. Max fell. John had pushed him).

By discarding superfluous phrases in a discourse tree structure (e.g. explanations) we can obtain a summary.

## Referring Expressions

*Referents* are real word entities that text refers to. *Referring expressions* are language that performs reference. *Antecedents* are texts evoking a referent (e.g. "Niall" is the antecedent of "the historian"). *Anaphora* are references to an antecedent (e.g. "the historian"). Finally, *cataphora* appear before their referents are introduced.

Pronoun number and gender can be used to help resolve anaphora. We can also avoid resolving *pleonastic* pronouns which are semantically empty (e.g. "it is snowing"). There are also a number of salience effects:

- Recency
- Grammatical role*:* subjects > objects > everything else.
- Repeated mention
- Parallelism*:* entities which share the same role as the pronoun in the same sort of sentence are preferred.
- Coherence

The Lappin and Leass algorithm provides a means to resolve anaphora:

1. For each sentence:
   (a) Divide all global salience factors for each equivalence class by 2
   (b) Identify referring NPs in the sentence (i.e. exclude pleonastic pronouns)

(c) Calculate per-NP salience factors and add those as referents to the global salience table

(d) For each pronoun in the sentence:

  i. Collect potential referents from most recent 4 sentences

  ii. Filter referents by binding and agreement

  iii. Obtain per-pronoun adjustments for each referent

  iv. Select the most salient referent taking into account the adjustment, preferring the closest referent in case of a tie

  v. Add the pronoun into the referent equivalence class, incrementing the salience factor by the relevant non-duplicate salience factors

Global salience factors are:

| Factor | Weight | Remarks |
|---|---|---|
| Recency | 100 | For current sentence |
| Subject | 80 | |
| Existential sentence object | 70 | |
| Direct object | 50 | |
| Indirect object | 40 | e.g. “Sandy” in “give a present to Sandy” |
| Oblique complement | 40 | |
| Non-embedded noun | 80 | Not a part of another noun phrase |
| Other non-adverbial | 50 | |

Per-pronoun salience factors are:

| Factor | Weight | Remarks |
|---|---|---|
| Cataphora | -175 | Can be negative since not added to global salience table |
| Same role | 35 | |