CENTRE FOR LANGUAGE TECHNOLOGY
MACQUARIE UNIVERSITY - SYDNEY

Grammars, Parsers and Realisers
Session 5: Linguistic Realisation

Robert Dale
Robert.Dale@mq.edu.au

You can find these slides at:

http://www.ics.mq.edu.au/~rdale/teaching/tutorials/ali2008-07-11.pdf

The Aims of This Session

- To provide an overview of what's involved in natural language generation (NLG)
- To explain how grammars fit into the generation process
- To provide some examples of how grammars are used in NLG

The Agenda

- Looking at Grammars from The Other End
- The Big Picture: Natural Language Generation
- What's Involved in Linguistic Realisation
- Some Examples
- Concluding Remarks

Grammars, Parsers and Realizers

- A grammar is a declarative specification of well-formedness in a language [= data]
- A parser is a process that uses a grammar to provide a structural analysis of a well-formed sentence in the language [= algorithm]
- A realizer is a process that uses a grammar to produce a well-formed sentence in the language [= algorithm]

A Simple Prolog Definite Clause Grammar

```
s --> np, vp.
np --> det, n.
vp --> v, np.
det -->[the].
n -->[cat].
n -->[mouse].
v -->[chased].
```

Using a Definite Clause Grammar To Determine Well-Formedness

```
1 ?- s([the,cat,chased,the,mouse],[]).
true .
2 ?- s([chased,the,cat,the,mouse],[]).
fail.
3 ?-
```

Using a Definite Clause Grammar To Generate Well-Formed Sentences

```
1 ?- s(Sentence,[]).
Sentence = [the, cat, chased, the, cat] ;
Sentence = [the, cat, chased, the, mouse] ;
Sentence = [the, mouse, chased, the, cat] ;
Sentence = [the, mouse, chased, the, mouse].
2 ?-
```

A Slightly More Complex Prolog Definite Clause Grammar

```
s --> np, vp.
np --> det, n.
vp --> v, np.
vp --> v.
det -->[the].
n -->[cat].
n -->[mouse].
v -->[chased].
v -->[slept].
```

Using a Definite Clause Grammar To Determine Well-Formedness

```
1 ?- s([the,cat,slept],[]).
true .
2 ?-
```

Using a Definite Clause Grammar To Generate Well-Formed Sentences

```
1 ?- s(Sentence,[]).
Sentence = [the, cat, chased, the, cat] ;
Sentence = [the, cat, chased, the, mouse] ;
Sentence = [the, cat, slept, the, cat] ;
Sentence = [the, cat, slept, the, mouse] ;
Sentence = [the, cat, chased] ;
Sentence = [the, cat, slept] ;
Sentence = [the, mouse, chased, the, cat] ;
Sentence = [the, mouse, chased, the, mouse] ;
Sentence = [the, mouse, slept, the, cat] ;
Sentence = [the, mouse, slept, the, mouse] ;
Sentence = [the, mouse, chased] ;
Sentence = [the, mouse, slept].
3 ?-
```

An Improved Prolog Definite Clause Grammar

```
s --> np, vp.
np --> det, n.
vp --> tv, np.
vp --> iv.
det -->[the].
n -->[cat].
n -->[mouse].
tv -->[chased].
iv -->[slept].
```

Using a Definite Clause Grammar To Generate Well-Formed Sentences

```
1 ?- s(Sentence,[]).
Sentence = [the, cat, chased, the, cat] ;
Sentence = [the, cat, chased, the, mouse] ;
Sentence = [the, cat, slept] ;
Sentence = [the, mouse, chased, the, cat] ;
Sentence = [the, mouse, chased, the, mouse] ;
Sentence = [the, mouse, slept].

2 ?-
```

Using a Definite Clause Grammar to Return a Syntactic Analysis

```
s(s(NP,VP)) --> np(NP), vp(VP).
np(np(Det,N)) --> det(Det), n(N).
vp(vp(TV,NP)) --> tv(TV), np(NP).
vp(vp(IV)) --> iv(IV).
det(det(the)) -->[the].
n(moun(cat)) -->[cat].
n(noun(mouse)) -->[mouse].
tv(verb(chased)) -->[chased].
iv(verb(slept)) --> [slept].
```

Using a Definite Clause Grammar to Return a Syntactic Analysis

```
1 ?- s(Tree,[the,cat,chased,the,mouse],[]).
Tree = s(np(det(the), moun(cat)), vp(verb(chased), np(det(the), noun(mouse)))) .

2 ?-
```

Using a Definite Clause Grammar to Generate a Sentence Given a Structure

```
1 ?- s(s(np(det(the),noun(mouse)),vp(verb(slept))),Sentence,[]).
Sentence = [the, mouse, slept].
2 ?-
```

Fine, But …

- Q: What's the point of generating from a syntax tree?
- A: There isn't one.
- Q: Also fair to ask: what's the point of producing a syntax tree from a sentence?
- A: So that you can do something else … like generate a representation of the meaning.
- So:
  - parsing is about mapping from a sentence to its semantics
  - realisation is about mapping from semantics to a sentence

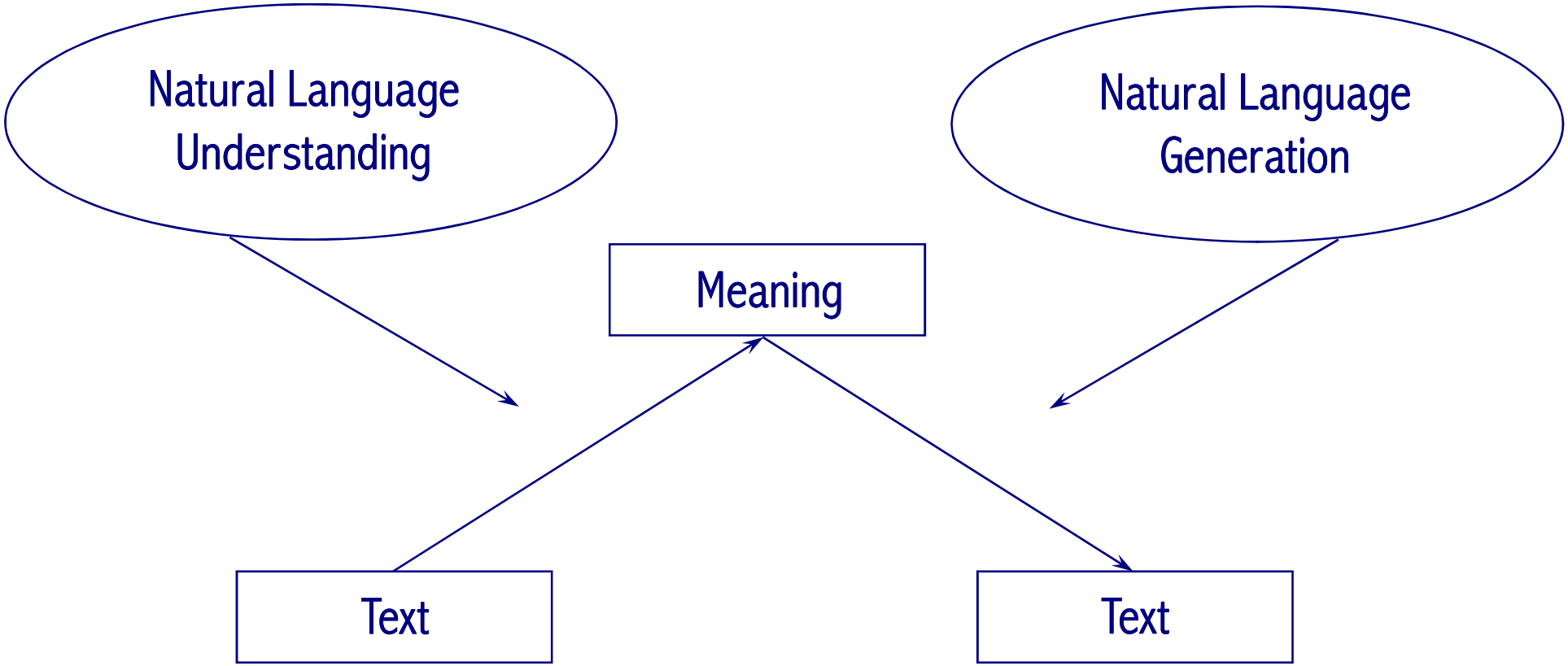
The Agenda

- Looking at Grammars from The Other End
- The Big Picture: Natural Language Generation
- What's Involved in Linguistic Realisation
- Some Examples
- Concluding Remarks

What is NLG?

- Goal:
  - computer software which produces understandable texts in English or other human languages
- Input:
  - some underlying non-linguistic representation of information
- Output:
  - documents, reports, explanations, help messages, and other kinds of texts

NLP = NLU + NLG


Natural Language Understanding
Natural Language Generation
Meaning
Text
Text

Inputs and Outputs

The inputs to NLG:

- A knowledge source
- A communicative goal
- A user model
- A discourse model

The output of NLG:

- A text, possibly embodied as part of a document or within a speech stream

Component Tasks in NLG

1. Content determination
2. Discourse planning
3. Sentence aggregation
4. Lexicalisation
5. Referring expression generation
6. Syntactic and morphological realization
7. Orthographic realization

1 Content Determination

- The process of deciding what to say
- Can be viewed as the construction of a set of MESSAGES from the underlying data source
- Messages are aggregations of data that are appropriate for linguistic expression: each may correspond to the meaning of a word or a phrase
- Messages are based on domain entities, concepts, and relations

2 Discourse Planning

- A text is not just a random collection of sentences
- Texts have an underlying structure in which the parts are related together
- Two related issues:
  - conceptual grouping
  - rhetorical relationships

3 Sentence Aggregation

- A one-to-one mapping from messages to sentences results in disfluent text
- Messages need to be combined to produce larger and more complex sentences
- The result is a sentence specification or SENTENCE PLAN

4 Lexicalisation

- So far we have determined text content and the structuring of the information into paragraphs and sentences, but the raw material is still assumed to be in the form of a conceptual representation
- Lexicalisation determines the particular words to be used to express domain concepts and relations

5 Referring Expression Generation

- Referring expression generation is concerned with how we describe domain entities in such a way that the hearer will know what we are talking about
- Do we use a proper name? A definite or indefinite description? A pronoun?

6 Syntactic and Morphological Realization

- Every natural language has grammatical rules that govern how words and sentences are constructed
  - Morphology: rules of word formation
  - Syntax: rules of sentence formation

7 Orthographic Realization

- Orthographic realization is concerned with matters like casing and punctuation
- This also extends into typographic issues: font size, column width …
- … and there are spoken language correlates: intonational phrasing, pauses, emphasis …

Tasks and Architecture in NLG

• Content determination • Discourse planning	Document Planning
• Sentence aggregation • Lexicalisation • Referring expression generation	Micro Planning
• Syntax + morphology • Orthographic realization	Linguistic Realization

The Agenda

- Looking at Grammars from The Other End
- The Big Picture: Natural Language Generation
- What's Involved in Linguistic Realisation
- Some Examples
- Concluding Remarks

The Input to Realisation

- Often referred to as 'sentence plans'
- The choice of representational level:
  - Skeletal Propositions
  - Meaning Specifications
  - Lexicalised Case Frames
  - Abstract Syntactic Structures

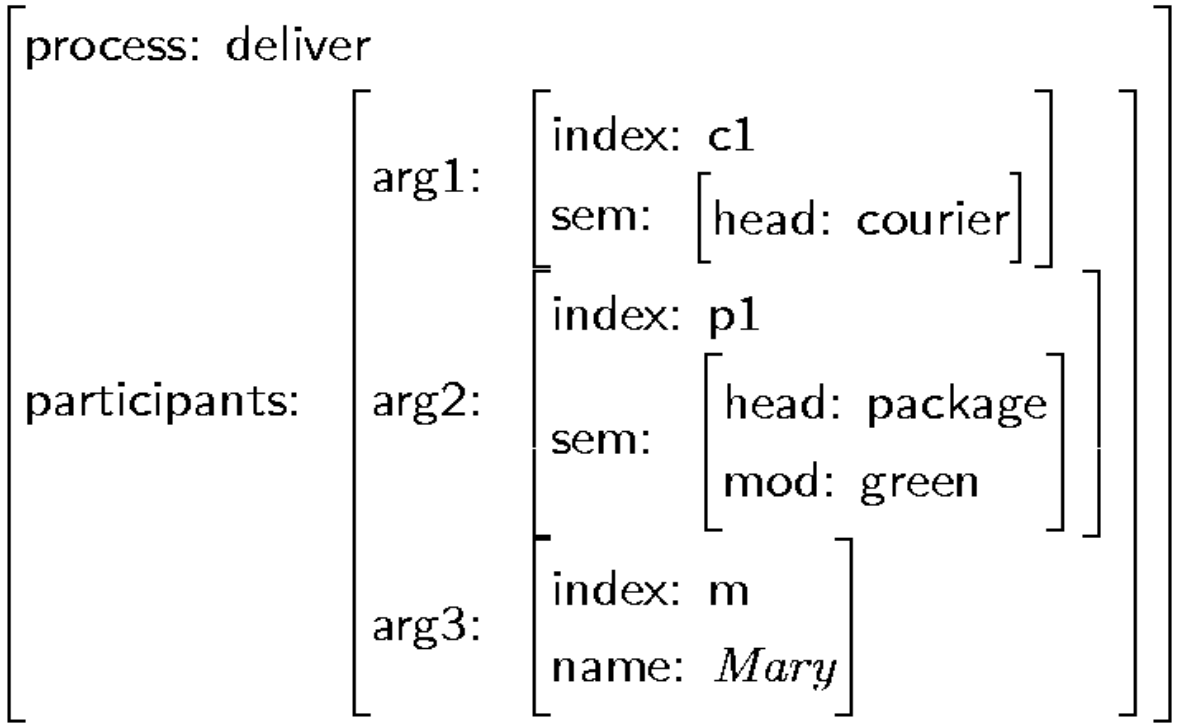
Propositional Content

- The target sentence to generate:
  - The courier delivered the green package to Mary
- Propositional content:
  - $\exists c1\, \exists p1\, \exists m\ deliver(c1, p1, m)$

Skeletal Propositions

$$\begin{bmatrix} \text{predicate: deliver} & \\ \text{arguments:} & \begin{bmatrix} \text{arg1: c1} \\ \text{arg2: p1} \\ \text{arg3: m} \end{bmatrix} \end{bmatrix}$$

Meaning Specifications



Lexicalised Case Frames

$$\begin{bmatrix} \text{process:} & \begin{bmatrix} \text{lex:} & \textit{deliver} \end{bmatrix} & \\ \text{participants:} & \begin{bmatrix} \text{arg1:} & \begin{bmatrix} \text{head:} & \begin{bmatrix} \text{lex:} & \textit{courier} \end{bmatrix} \end{bmatrix} \\ \text{arg2:} & \begin{bmatrix} \text{head:} & \begin{bmatrix} \text{lex:} & \textit{package} \end{bmatrix} \\ \text{mod:} & \begin{bmatrix} \text{lex:} & \textit{green} \end{bmatrix} \end{bmatrix} \\ \text{arg3:} & \begin{bmatrix} \text{head:} & \begin{bmatrix} \text{lex:} & \textit{Mary} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Abstract Syntactic Structures

$$\begin{bmatrix} \text{verb: deliver} & \\ \text{arguments:} & \begin{bmatrix} \text{subject:} & \begin{bmatrix} \text{head: } courier \end{bmatrix} \\ \text{object:} & \begin{bmatrix} \text{head: } package \\ \text{mod: } green \end{bmatrix} \\ \text{indirectobject:} & \begin{bmatrix} \text{head: } Mary \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Sentence Planning Language

```
(S1/ThereBe
   :object (O1/train
            :cardinality 20
            :relations
              ((R1/period :value daily)
               (R2/source :value Aberdeen)
               (R3/destination :value Glasgow))))
```

There are 20 trains daily from Aberdeen to Glasgow.

A Realisation Specification in MUMBLE

```
(discourse-unit
  :head (general-clause
          :head (chase
                  (general-np
                    :head (np-proper-name "Fluffy")
                    :accessories
                       (:number singular
                        :determiner-policy no-determiner))
                  (general-np
                    :head (np-common-noun "mouse")
                    :accessories
                       (:number singular
                        :determiner-policy kind))
                    :further-specifications
                       ((:specification
                           (predication_to-be *self*
                              (adjective "little"))
                         :attachment-function
                            restrictive-modifier)))))
          :accessories (:tense-modal present
                        :progressive
                        :unmarked))))
```

$\Rightarrow$ Fluffy chases little mice.

The Agenda

- Looking at Grammars from The Other End
- The Big Picture: Natural Language Generation
- What's Involved in Linguistic Realisation
- Some Examples
  - Functional Unification Grammar
  - Systemic Functional Grammar
- Concluding Remarks

FUF/SURGE

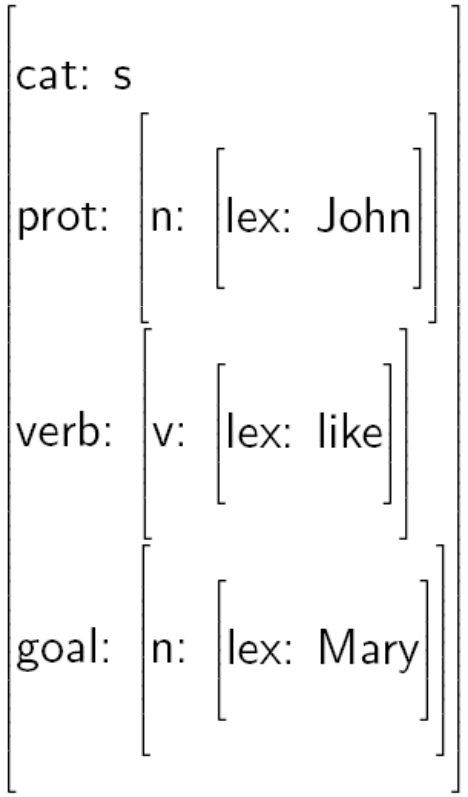
- FUF: a unification-based linguistic realisation toolkit
- SURGE: a unification grammar of English

FUF/SURGE

Basic idea:

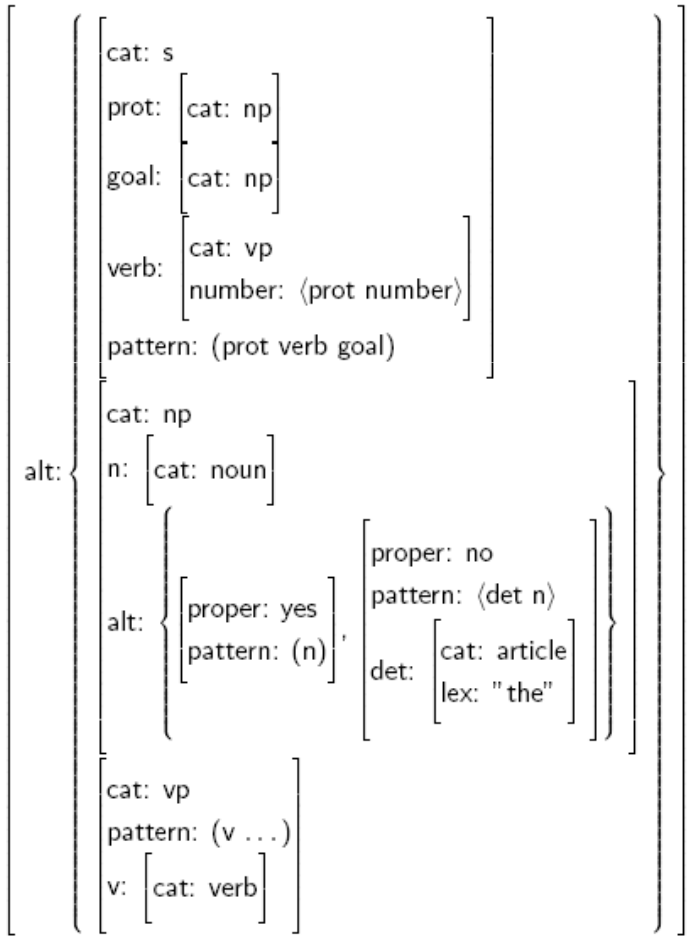
- input specification in the form of a FUNCTIONAL DESCRIPTION, a recursive attribute–value matrix
- the grammar is a large functional description with alternations representing choice points
- realisation is achieved by unifying the input FD with the grammar FD

An Input Functional Descriptor in SURGE



$\Rightarrow$ John likes Mary.

A Simple Grammar in SURGE



The Agenda

- Looking at Grammars from The Other End
- The Big Picture: Natural Language Generation
- What's Involved in Linguistic Realisation
- Some Examples
  - Functional Unification Grammar
  - Systemic Functional Grammar
- Concluding Remarks

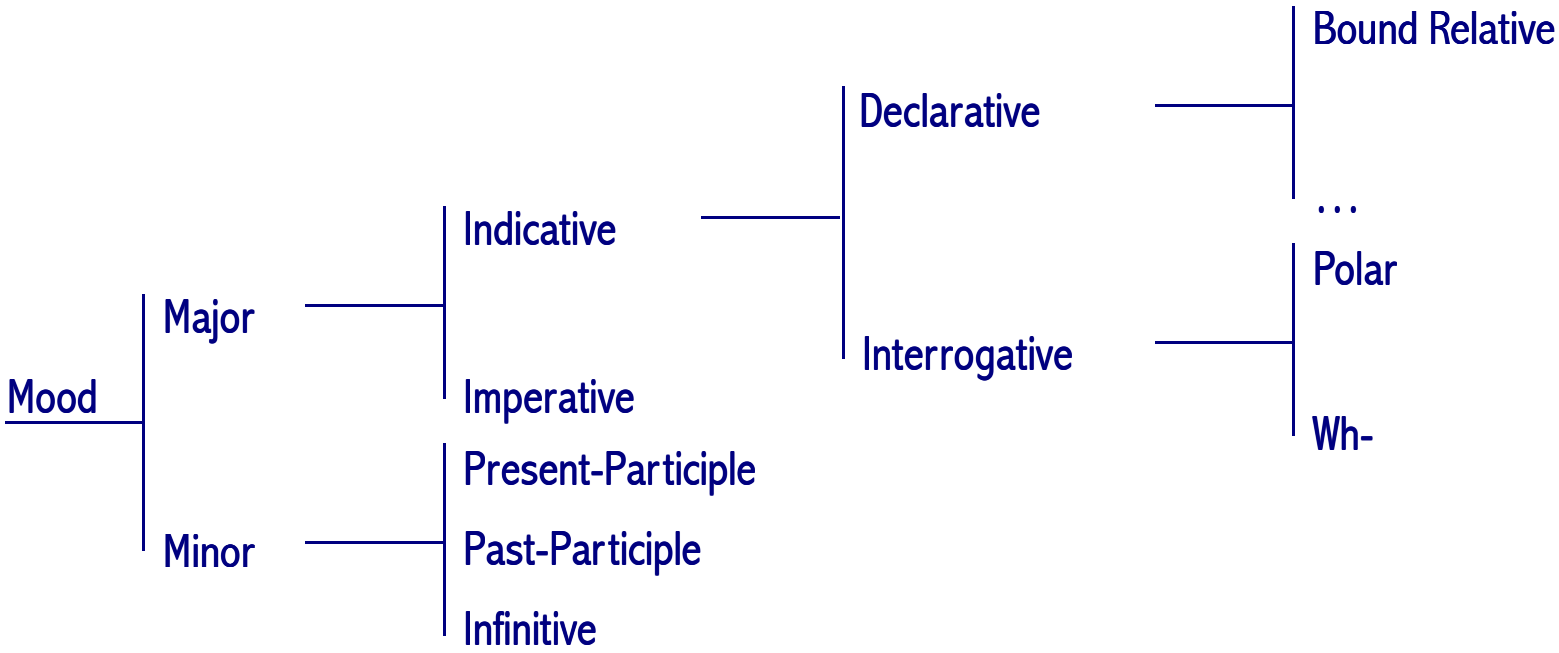
Grammatical Representations for Generation

- Phrase structure grammars are essentially concerned with mapping from form to meaning
- Systemic functional grammar is essentially concerned with mapping from meaning (or function) to form
- These are different ways of organising the available lexicogrammatical resources: like the difference between a contents page and a back-of-the-book index

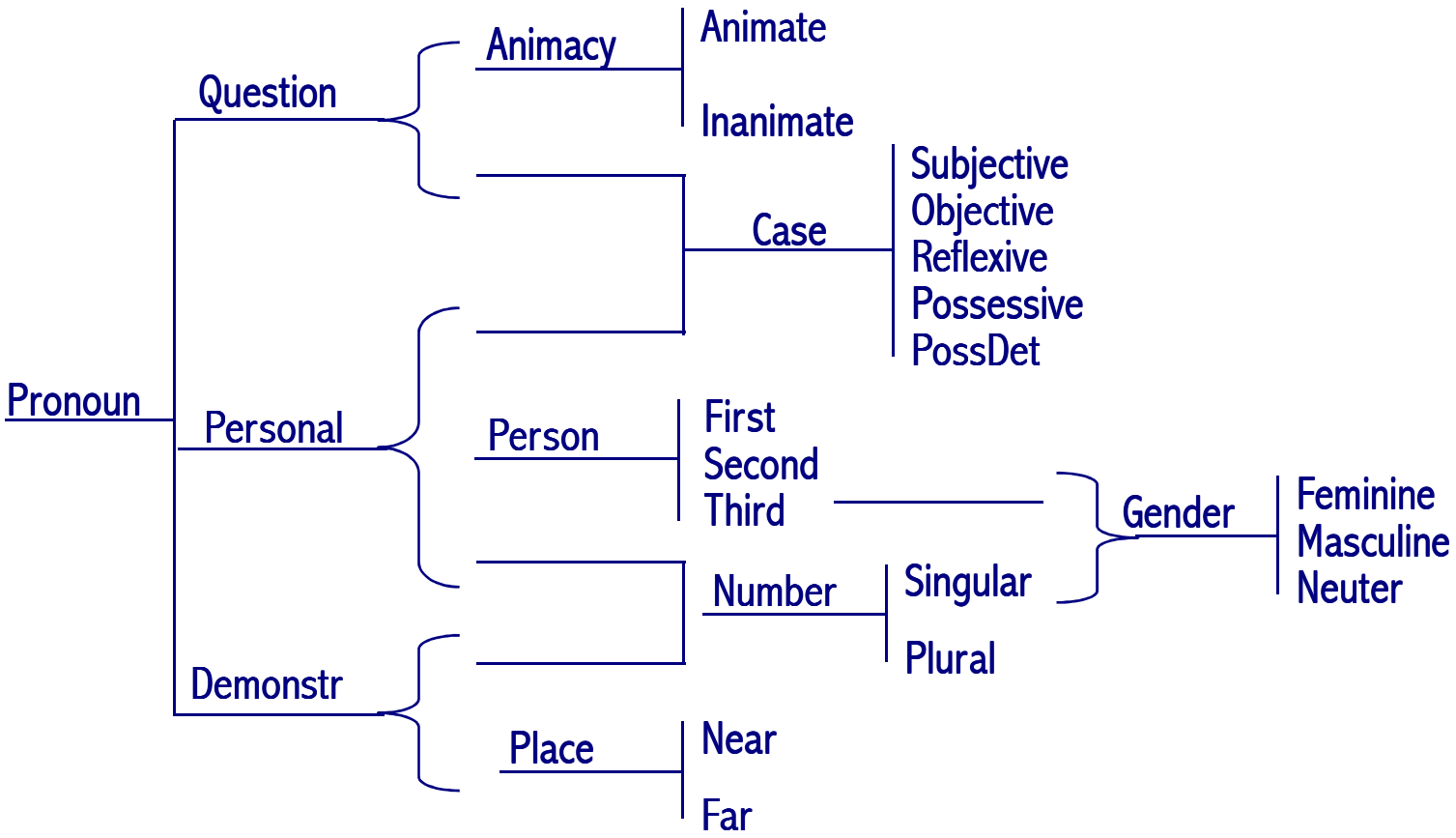
Systemic Grammar

- Emphasises the functional organisation of language
- Surface forms are viewed as the consequences of selecting a set of abstract functional features
- Choices correspond to minimal grammatical alternatives
- The interpolation of an intermediate abstract representation allows the specification of the text to accumulate gradually

Systemic Grammar: The Clause



A Grammar for Pronouns



Realisation Rules for Pronouns

question animate subjective	→ *who*
question animate objective	→ *whom*
question animate possessive	→ *whose*
question inanimate	→ *what*
demonstr singular near	→ *this*
demonstr singular far	→ *that*
personal first singular subjective	→ *I*
personal first singular objective	→ *me*
personal first singular reflexive	→ *myself*

Systemic Grammar

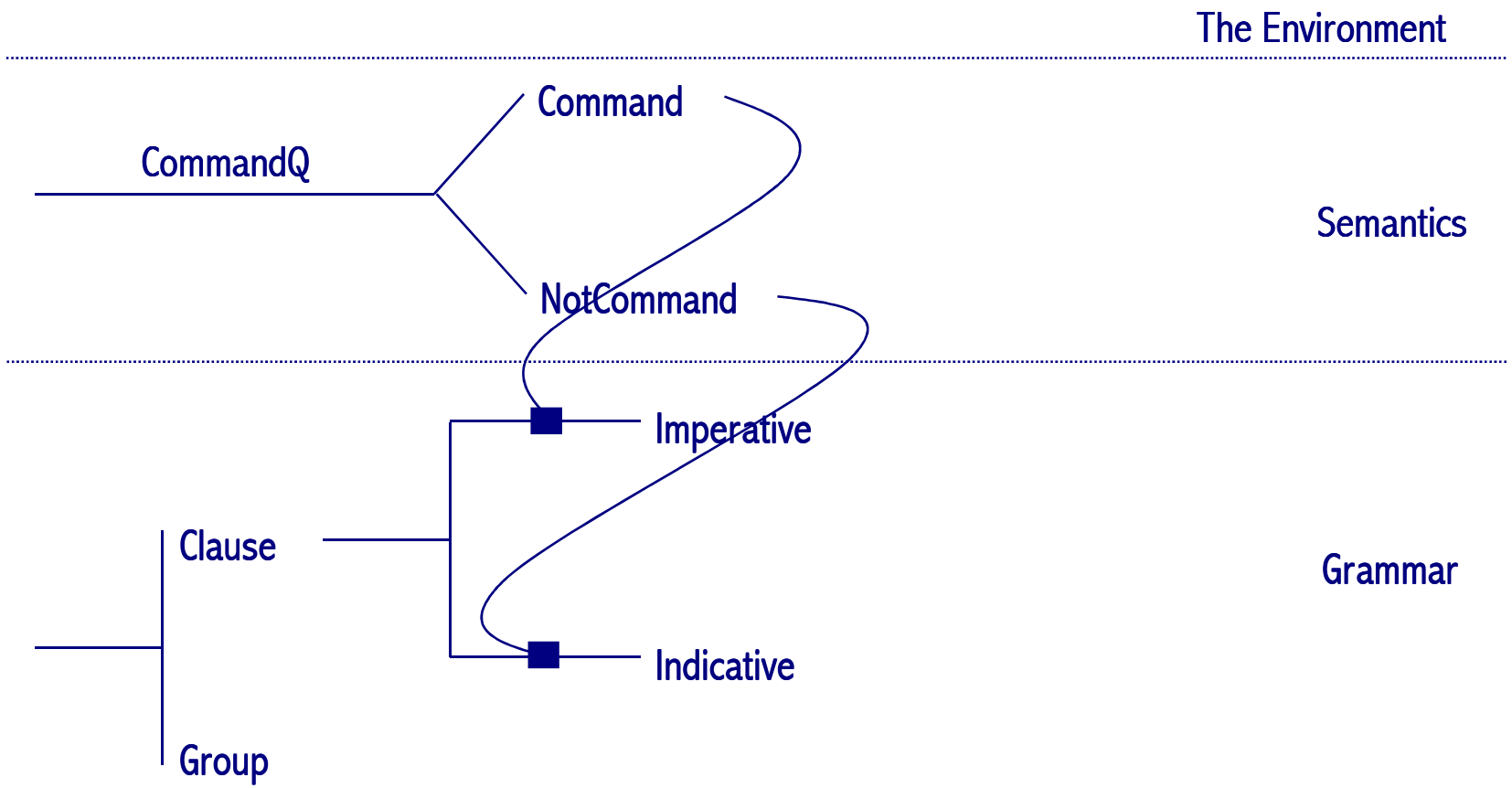
- So far this is just a particular taxonomisation of the resources in a language
- Two things needed to make this do work for us:
  - Choices in the network need to result in grammatical characteristics
  - Choices in the network need to be motivated by the NLG system's intentions

The Penman Model

How it works:

- choices are made using INQUIRY SEMANTICS
- for each choice system in the grammar, a set of predicates known as CHOOSERS are defined
- these tests are functions from the internal state of the realiser and host generation system to one of the features in the system the chooser is associated with

Choosers and Inquiries



Choosers: An Example

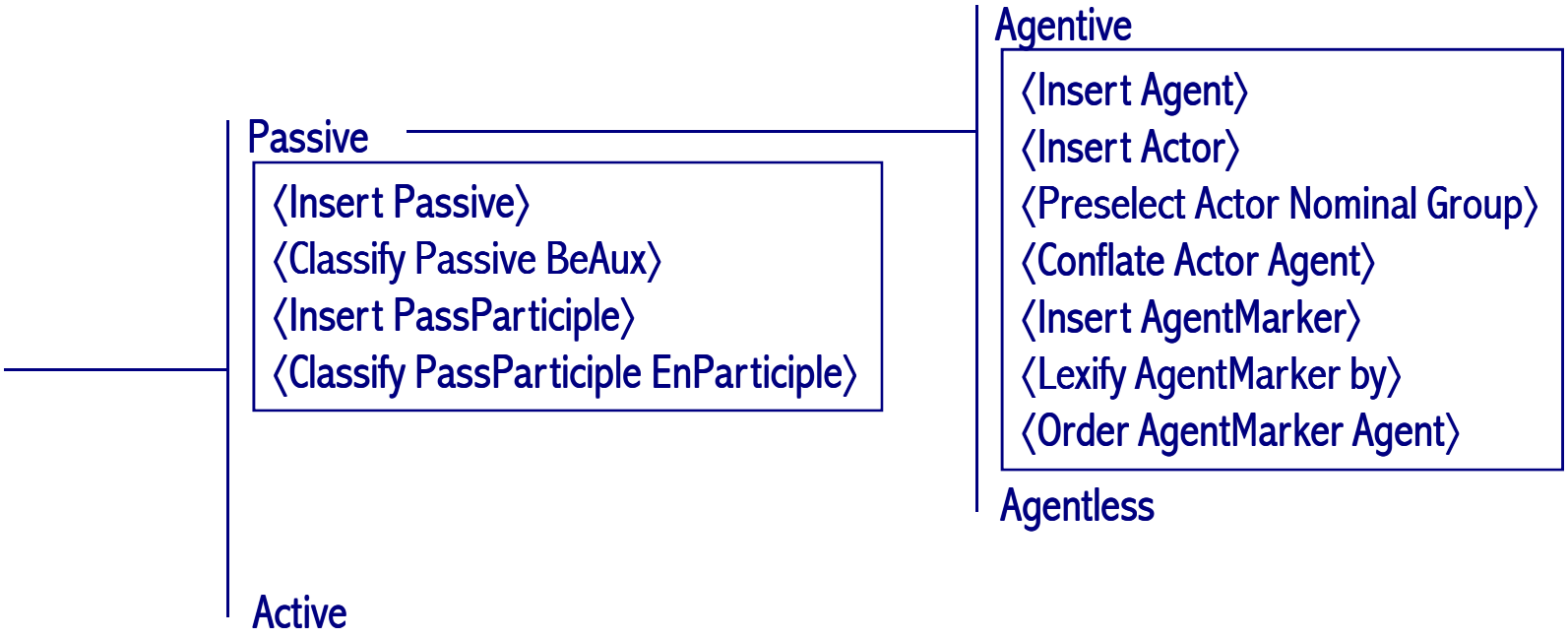
- To choose between a definite and an indefinite article, a chooser might query:
  - the knowledge base to determine whether the head of the NP refers to a generic or individual concept
  - the discourse model to determine whether the object has been previously mentioned

The Penman Model

Realisation Statements:

- small grammatical constraints at each choice point build up to a grammatical specification
- ⟨Insert SUBJECT⟩: an element functioning as subject will be present
- ⟨Conflate SUBJECT ACTOR⟩: the constituent functioning as SUBJECT is the same as the constituent that functions as ACTOR
- ⟨Order FINITE SUBJECT⟩: FINITE must immediately precede SUBJECT

Realisation Statements



An SPL Input

```
(rst / rst-concessive
  :domain
    (l / greater-than-comparison
      :tense past  :exceed-q (l a) exceed
      :domain (m / one-or-two-d-time :name June)
      :standard (a / quality :lex average)
      :range ((wa / sense-and-measure-quality :lex warm)
              (we / sense-and-measure-quality :lex wet)))
  :range
    (sp / existence
      :tense past
      :domain (s / abstraction
                :lex spell
                :property-ascription (d / quality :lex dry))
      :source (2nd / one-or-two-d-time
                :lex 2nd
                :destination (5th / one-or-two-d-time
                                  :lex 5th    :determiner the))))
```

Advantages of SFG for NLG

- May be more natural and economical to state syntactic regularities in a functional framework
- Cross-language generalisations may be better stated in functional terms
- The analysis embodies several aspects of meaning:
  - ideational
  - interpersonal
  - textual

The Agenda

- Looking at Grammars from The Other End
- The Big Picture: Natural Language Generation
- What's Involved in Linguistic Realisation
- Functional Unification Grammar
- Systemic Functional Grammar
- Concluding Remarks

Finding Out More: Natural Language Generation in General

- E Reiter and R Dale [2000] Building Natural Language Generation Systems. Cambridge University Press. [Paperback edition 2006]
- http://www.ics.mq.edu.au/~rdale/teaching/tutorials.html

Finding Out More: Unification Grammars

- Prolog and Definite Clause Grammars:
  - F. C. N. Pereira and S. M. Shieber [1987] Prolog and Natural-Language Analysis. Volume 10 of CSLI Lecture Notes Series, Center for the Study of Language and Information, Stanford U.
- Unification:
  - S. M. Shieber [1989] An Introduction to Unification-Based Approaches to Grammar. Volume 4 of CSLI Lecture Notes Series, Center for the Study of Language and Information, Stanford U.

Finding Out More: Implemented Realisers

- FUF/SURGE:
  - http://www.cs.bgu.ac.il/surge/index.html
- KPML, an SFG generator:
  - http://www.fb10.uni-bremen.de/anglistik/langpro/kpml/README.html

Finding Out More: Realisers for Other Formalisms

- Categorial Grammar:
  - http://openccg.sourceforge.net/
- Tree Adjoining Grammar:
  - http://wiki.loria.fr/wiki/GenI
- HPSG:
  - http://lingo.stanford.edu/