An Introduction to Natural Language Generation

Robert Dale
Microsoft Institute of Advanced Software Technology
and
School of Mathematics, Physics, Computing and Electronics
Macquarie University
Sydney
Australia
rdale@microsoft.com

1. An Overview of NLG
2. Linguistic Realization
3. Text Planning
4. Generating Referring Expressions

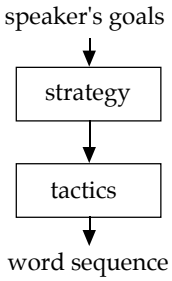
Topic 2:

Linguistic Realization

Overview

- The Nature of the Input
- Unification-based Approaches to Realisation
- Data-driven Approaches to Realisation
- Systemic Functional Grammar
- Towards a Synthesis

Realisation as Tactical Generation



Common Assumptions about the Input

1. Some other process has constructed the message.
2. Each message is realisable as a sentence.
3. The semantic content of referring expressions has been already determined.
4. The open-class lexical items have already been determined.

Example Input #1

Conventional First Order Predicate Calculus:

1. $\mathsf{gives}(m, j, b1)$
   $\Rightarrow$ Mary gave John a book.
2. $\forall x\ \mathsf{farmer}(x) \rightarrow \exists y\ \mathsf{donkey}(y) \wedge \mathsf{beats}(x, y)$
   $\Rightarrow$ Every farmer beats a donkey.

Example Input #2

Davidsonian logical forms:

- $\exists e, t, x, y, z$ event-type(e, giving)
  $\wedge$ time(e, t) $\wedge\ t <$ now
  $\wedge$ agent(e, x) $\wedge$ name(x, "John")
  $\wedge$ benefactor(e, y) $\wedge$ name(y, "Mary")
  $\wedge$ object(e, z) $\wedge$ isa(z, book)

Example Input #3

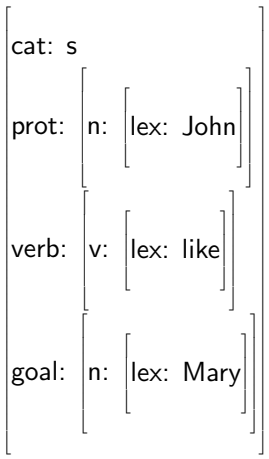
SPL expressions:

```
(p1 / class-ascription
    :domain (A2 / adder
                :identifiability-q identifiable)
    :range (B1 / binary-operator
                :identifiabilty-q notidentifiable))
```

$\Rightarrow$ The adder is a binary operator.

Example Input #4

Surge IRs:



⇒ John likes Mary.

Example Input #5

Mumble realisation specifications:

```
(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))))
```

⇒ Fluffy chases little mice.

Overview

- The Nature of the Input
- Unification-based Approaches
- Data-driven Approaches to Realisation
- Systemic Functional Grammar
- Towards a Synthesis

Unification-based Approaches

- PATR-II
- FUG
- PATR-II and FUG compared

Unification Grammar

- Basic idea: linguistic objects described by FEATURE STRUCTURES
- A feature structure is a collection of attribute–value pairs
- The value of an attribute can itself be a feature structure
- Feature structures are combined by means of grammar rules
- Grammar rules impose constraints on legal combinations of feature structures
- A grammar rule consists of a CONTEXT-FREE BACKBONE and a collection of PATH EQUATIONS

Unification Grammar in PATR-II

A simple feature structure:

$$\begin{bmatrix} \text{cat: np} \\ \text{agreement: } \begin{bmatrix} \text{number: singular} \\ \text{person: third} \end{bmatrix} \end{bmatrix}$$

Grammar Rules in PATR-II

1. $X0 \rightarrow X1\ X2$
   $\langle X0\ cat \rangle = s$
   $\langle X1\ cat \rangle = np$
   $\langle X2\ cat \rangle = vp$
2. $X0 \rightarrow X1\ X2$
   $\langle X0\ cat \rangle = s$
   $\langle X1\ cat \rangle = np$
   $\langle X2\ cat \rangle = vp$
   $\langle X1\ agreement \rangle = \langle X2\ agreement \rangle$

Grammar Rules in PATR-II

$X0 \rightarrow X1\ X2$
$\langle X0\ cat \rangle = s$
$\langle X1\ cat \rangle = np$
$\langle X2\ cat \rangle = vp$
$\langle X0\ head \rangle = \langle X2\ head \rangle$
$\langle X0\ head\ subj\ head \rangle = \langle X1\ head \rangle$

$X0 \rightarrow fred$
$\langle X0\ cat \rangle = np$
$\langle X0\ head\ agr\ num \rangle = sing$
$\langle X0\ head\ agr\ pers \rangle = 3$

$X0 \rightarrow sleeps$
$\langle X0\ cat \rangle = vp$
$\langle X0\ head\ vform \rangle = fin$
$\langle X0\ head\ subj\ head\ agr\ num \rangle = sing$
$\langle X0\ head\ subj\ head\ agr\ pers \rangle = 3$

Syntax–Semantics Mapping in PATR-II

Logical expressions as feature structures:

- *Uther storms Cornwall*

$$\Rightarrow \begin{bmatrix} \text{pred: storm} \\ \text{arg1: uther} \\ \text{arg2: cornwall} \end{bmatrix}$$

- *Uther persuades Arthur to sleep*

$$\Rightarrow \begin{bmatrix} \text{pred: persuade} \\ \text{arg1: uther} \\ \text{arg2: arthur} \\ \text{arg3:} \begin{bmatrix} \text{pred: sleep} \\ \text{arg1: arthur} \end{bmatrix} \end{bmatrix}$$

Syntax–Semantics Mapping in PATR-II

Lexical entries as syntax–semantics correspondences:

Uther $\longmapsto$

$$\begin{bmatrix} \text{cat: NP} \\ \text{head:} \begin{bmatrix} \text{agreement:} \begin{bmatrix} \text{number: singular} \\ \text{person: third} \end{bmatrix} \\ \text{trans: uther} \end{bmatrix} \end{bmatrix}$$

Syntax–Semantics Mapping in PATR-II

Syntax–semantics correspondence in a lexicalised grammar:

storms $\longmapsto$

$$\begin{bmatrix} \text{cat: V} \\ \text{head:} \begin{bmatrix} \text{form: finite} \\ \text{trans:} \begin{bmatrix} \text{pred: storm} \\ \text{arg1: } \boxed{2}[] \\ \text{arg2: } \boxed{1}[] \end{bmatrix} \end{bmatrix} \\ \text{subcat:} \begin{bmatrix} \text{first:} \begin{bmatrix} \text{cat: NP} \\ \text{head:} \begin{bmatrix} \text{trans: } \boxed{1} \end{bmatrix} \end{bmatrix} \\ \text{rest:} \begin{bmatrix} \text{first:} \begin{bmatrix} \text{cat: NP} \\ \text{head:} \begin{bmatrix} \text{agreement:} \begin{bmatrix} \text{number: singular} \\ \text{person: third} \end{bmatrix} \\ \text{trans: } \boxed{2} \end{bmatrix} \end{bmatrix} \\ \text{rest: end} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Unification-based Approaches

- PATR-II
- FUG
- PATR-II and FUG compared

Functional Unification Grammar

- formulated by Kay [1979]
- intended to be neutral between generation and analysis
- can be used to flesh out minimal, conceptually derived functional descriptions
- provides a modular, independent way of supplying purely linguistic information
- imposes no specific demands on the generator's control structure

Motivating Views

> Language is a system for encoding and transmitting ideas. A theory that seeks to explain linguistic phenomena in terms of this fact is a *fuctional* theory.
>
> ...a theory that shows how the sentences of a language are all generable by rules of a particular formal system ...does not explain anything.
>
> ...any reasonable linguistic theory will be functional.
>
> [Kay 1982:251]

What Makes FUG Functional

- primary status given to functional aspects of language; logical aspects are not priveleged
- linguistic structures described in terms of the function a part fills in the whole, rather than in terms of parts of speech and ordering relations
- grammars are required to function: to support language generation and analysis

Functional Notions

- given and new
- focus
- speech acts
- ...

Functional Unification Grammar

- each linguistic object (word, phrase, or clause) is represented by a FUNCTIONAL DESCRIPTION (FD), also called a FEATURE STRUCTURE or ATTRIBUTE–VALUE MATRIX
- an FD is a collection of FEATURES (or ATTRIBUTES or LABELS) where each feature has a value that can either be atomic or another functional description
- the order of feature–value pairs is not significant
- a unification grammar is itself a large FD that characterizes the features of every possible sentence in the language

Unification

- combines two FDs into a single structure
- result is an FD that contains all attributes of both the original FDs provided they are compatible
- simple atomic values are compatible iff they are identical
- values which are complex FDs are compatible iff all their attributes are compatible
- if a particular attribute is present in both FDs, then its values from both FDs are unified
- attributes which are present only in one FD are retained in the result
- unification is recursive, terminating when all embedded attributes have been unified
- if any attributes fail to unify, then the entire unification fails

A Simple Functional Description

$$\begin{bmatrix} \text{cat: determiner} \\ \text{lex: "the"} \end{bmatrix}$$

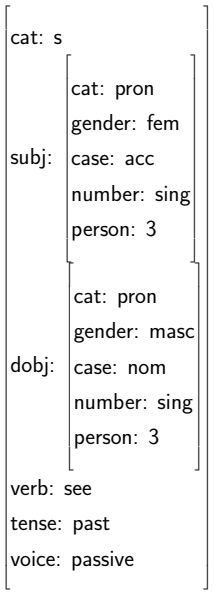
An FD for a Sentence

He saw her.

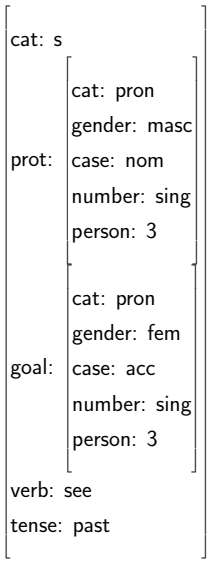
$$\begin{bmatrix} \text{cat: s} \\ \text{subj:} \begin{bmatrix} \text{cat: pron} \\ \text{gender: masc} \\ \text{case: nom} \\ \text{number: sing} \\ \text{person: 3} \end{bmatrix} \\ \text{dobj:} \begin{bmatrix} \text{cat: pron} \\ \text{gender: fem} \\ \text{case: acc} \\ \text{number: sing} \\ \text{person: 3} \end{bmatrix} \\ \text{verb: see} \\ \text{tense: past} \\ \text{voice: active} \end{bmatrix}$$

An FD for a Sentence

She was seen by him.



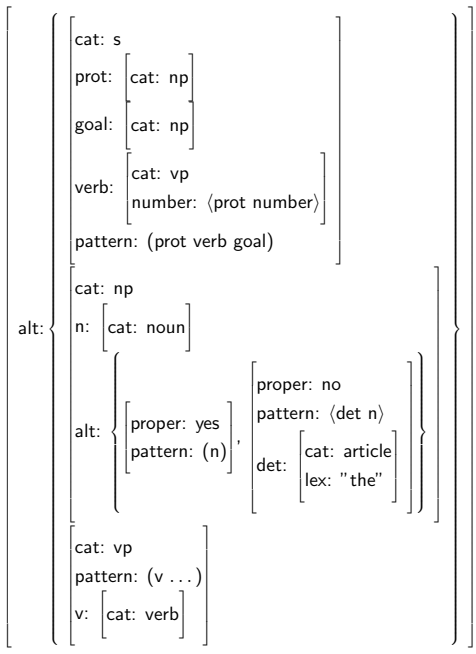
An FD for Both Sentences



Additional Machinery

- patterns
- paths

A Simple Grammar



Generation by Unification

- The input FD represents the semantic content of the message to be realised
- The grammar FD describes the space of grammatical alternatives and the correspondences of these to semantic elements
- Generation involves unifying the input FD and the grammar FD
- The output is a sentence expressing this meaning according to the grammatical constraints of the language

Generation by Unification

Two component processes:

- Unification enriches the input FD with word order, syntactic constructions, number agreement ...
- Linearisation includes morphology

Unification-based Approaches

- PATR-II
- FUG
- PATR-II and FUG compared

A Simple Grammar Rule

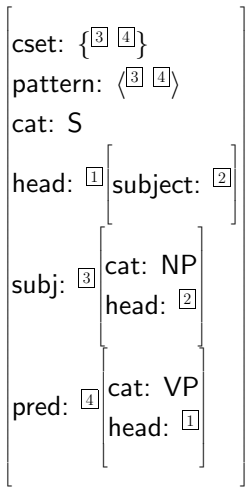
In PATR-II:

$$\begin{aligned} &S \rightarrow NP\ VP \\ &\quad \langle S\ head \rangle = VP\ head \\ &\quad \langle S\ head\ subject \rangle = \langle NP\ head \rangle \end{aligned}$$

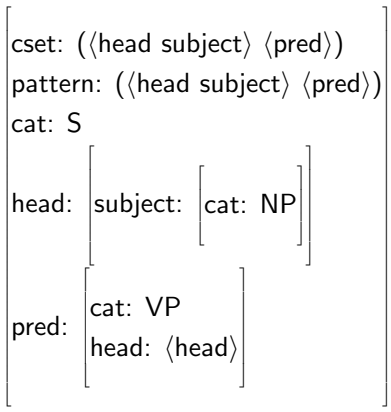
In FUG:

$$\begin{bmatrix} \text{cat: S} \\ \text{head: } \boxed{1} \begin{bmatrix} \text{subject: } \boxed{2} \end{bmatrix} \\ \text{subj: } \begin{bmatrix} \text{cat: NP} \\ \text{head: } \boxed{2} \end{bmatrix} \\ \text{pred: } \begin{bmatrix} \text{cat: VP} \\ \text{head: } \boxed{1} \end{bmatrix} \end{bmatrix}$$

Constituency and Order



A More FUG-like Rendering



Michael Elhadad's FUF

- Extends FUG with a variety of other control elements
- Uses typed feature structures
- Comes with SURGE, a large grammar of English
- Written in Common Lisp
- Available from `elhadad@cs.bgu.ac.il`

Unification-Based Approaches

- Benefits of declarative representation:
  - perspicuous formalism for the grammar writer
  - independence from process allows varied processing strategies
  - independence from process supports bidirectionality
- Inefficiency is a disadvantage
- Complex grammars are still very complex

Overview

- The Nature of the Input
- Unification-based Approaches to Realisation
- Data-driven Approaches to Realisation
- Systemic Functional Grammar
- Towards a Synthesis

Direct Replacement in SHRDLU

Example dialogue:

User: Stack up both of the red blocks and either a green cube or a pyramid.

SHRDLU: Okay.

User: How did you do it?

SHRDLU: By putting a large red block on the table; then letting go of it; then putting a large green cube on it; then letting go of that cube; then putting the red cube on that cube; then letting go of that cube.

Direct Replacement in SHRDLU

Example output:

> By putting a large red block on the table; then letting go of it; then putting a large green cube on it; then letting go of that cube; then putting the red cube on that cube; then letting go of that cube.

The underlying message:

```
(#puton :B6 :table)
(#ungrasp :B6)
(#puton :B3 :B6)
(#ungrasp :B3)
(#puton :B1 :B3)
(#ungrasp :B1)
```


Direct Replacement in SHRDLU

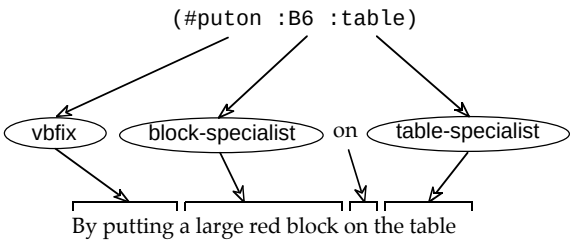
How it works:

- generation achieved by specialist code fragments associated with terms in the internal representation
- the blocks and the table have special programs associated with them that cause the appropriate sequence of words to be determined
- the generation program for `#puton` is

  ```
  (append (vbfix 'put) obj1 'on obj2)
  ```

- `vbfix` (verb fix) is a special purpose routine that attunes the verb to the grammatical context

Direct Replacement in SHRDLU



Direct Replacement Approaches

Summary:

- the coherence of the text comes from the coherence of the message
- the generator 'executes' the chosen message
- linguistic resources are chosen by linking objects in the message to their *realization specialists*, which are in most cases simply templates

Direct Replacement Approaches

Limitations:

- direct link between perception and action
- no explicit notion of function
- there is nowhere that linguistic generalizations can be stated

Approaches to Linguistic Realization

In generation, the purpose of a grammar is to define and constrain linguistic choices. Two perspectives:

Grammar-directed: Taking the language as a whole, what information-bearing distinctions does it possess, and what are the dependencies on their co-occurrence?

Message-directed: for a given element of the content or intent, what are the alternative linguistic resources available for its realization and the constraints on their use?

McDonald's MUMBLE

1. The basic ideas
2. Data structures:
   - The Message Specification
   - Realisation Classes
3. MUMBLE's control structure
4. A worked example

McDonald's MUMBLE: Basic Ideas

MUMBLE provides an input specification language to be used by a text planner, which must specify:

- the units from which the utterance is to be composed
- the functional relationships between the units
- choice of lexical heads

Main characteristics:

- description directed
- indelible process
- psychologically motivated

McDonald's MUMBLE: Basic Ideas

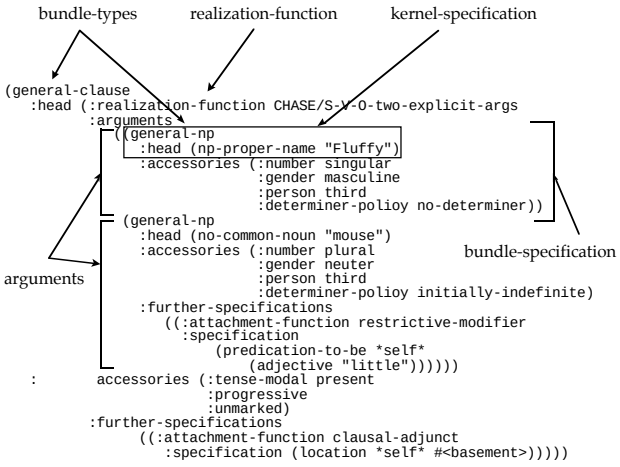
The input to generation: the MESSAGE LEVEL

- specifies what is to be said
- constrains how it is to be said
- consists of REALIZATION SPECIFICATIONS

What MUMBLE does:

- assembles text, guaranteeing grammaticality and expression of the indicated functional relationships
- maintains syntactic context and morphological specifications
- defines and applies contextual constraints on realization

An Example Bundle Specification



The Input Specification Language

bundle	⟶	bundle-type head accessories further-specs
bundle-type	⟶	`discourse-unit` | `general-np` | `general-clause` | `conjunction-bundle`
head	⟶	`:head` r-spec
r-spec	⟶	bundle | kernel
kernel	⟶	realization-function argument* | realization-function word
realization-function	⟶	`chase` | ...
argument	⟶	r-spec
accessories	⟶	`:accessories` accessory*
accessory	⟶	`:aspect` | `:number` | ...
further-specs	⟶	further-spec*
further-spec	⟶	`:specification :attachment-function`

Bundle Specifications in MUMBLE

- Bundles belong to one of four types: GENERAL-CLAUSE, GENERAL-NP, DISCOURSE-UNIT, and CONJUNCTION
- Each bundle type has a different driver and a different set of possible accessories: for example, GENERAL-CLAUSE has associated accessories TENSE-MODAL and QUESTION, and GENERAL-NP has accessories NUMBER and GENDER

McDonald's MUMBLE: Realization Classes

- the same internal objects and relations may be realized in different ways in different situations, depending on the context
- in MUMBLE, the process of choice is managed by grouping the alternatives according to the type of object involved
- these groupings are called REALIZATION CLASSES

McDonald's MUMBLE

A realization class:

```
(define-realization-class LOCATIVE-RELATION
  :parameters (Relation Arg1 Arg2)
  :choice
     ((Arg1-is-Relation-Arg2)
      ;; The driveway is next to the house
      clause focus(Arg1))
     ((Arg2-has-Arg1-Relation-Arg2)
      ;; The house has a driveway in front of it
      clause focus(Arg1))
     ((There-is-a-Arg1-Relation-Arg2)
      ;; There is a driveway next to the house
      root-clause shifts-focus-to(Arg1))
     ((Relation-Arg2-is-Arg1)
      ;; Next to the house is a driveway
      root-clause shifts-focus-to(Arg1)
      final-position(Arg1))
     ((with-Arg1-Relation-Arg2)
      ;; ... with a driveway next to it
      prepp modifier-to(Arg1)))
```

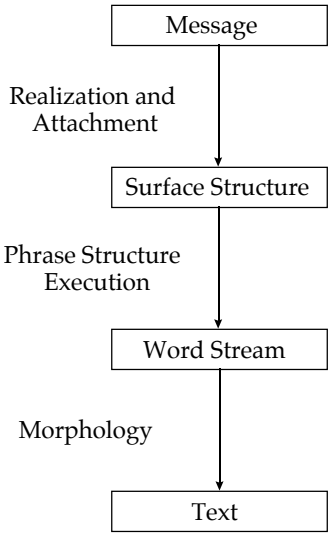
McDonald's MUMBLE: Control Structure

Basic mode of operation:

- two cascaded transducers driven by the input data structure
- first transducer replaces part of the message with some structural realization
- second transducer walks around the tree, producing words or invoking the first transducer

It is always the message that determines what happens next; at any time the message is a mixture of syntactic, semantic and pragmatic elements.

The Control Structure in MUMBLE

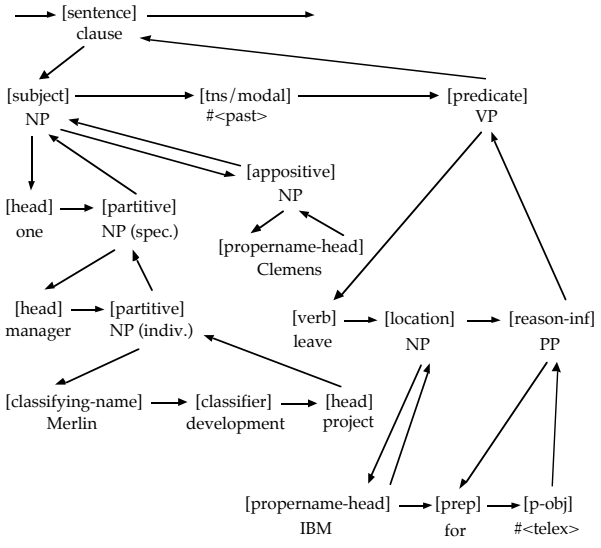


The Surface Structure in MUMBLE

Surface structure is represented in POSITION PATH NOTATION:

- linked list of positions corresponding to depth first left to right traversal of the tree
- positions annotated with labels carrying grammatical constraints, specifications of action to be carried out, and indications of where phrases may be attached
- positions have contents: a position directly dominates a word, an unrealized specification, or a rooted phrase

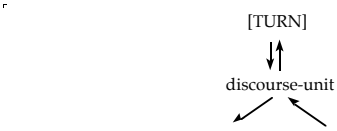
The Surface Structure in MUMBLE



A Worked Example: The Input Expression

```
(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))))
```


Initializing the Linguistic Context




```
(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)))
```


Phrase Structure Execution

Traverses the tree so far.

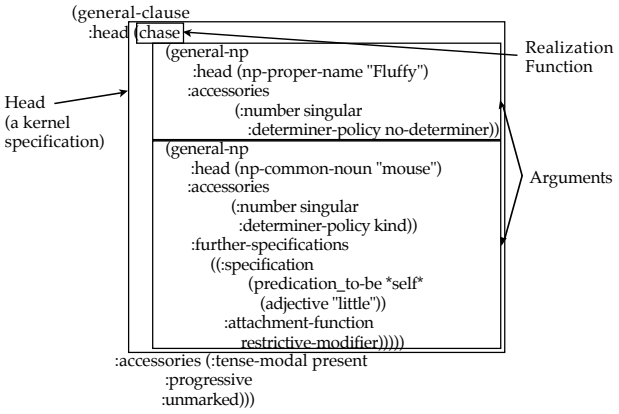
- Notes a grammatical constraint that the unit should be realized as a clause.
- Adds tokens to indicate that the unit should begin with a capital letter and terminate with a period.
- Passes control to REALIZATION.

Realization

The specification is a bundle, so the appropriate driver is called. This defines the order of realization of the parts.

1. Realize the head of the bundle.
2. Process the accessories.
3. Realize any further specifications.

Realizing the Head



A Realization Class

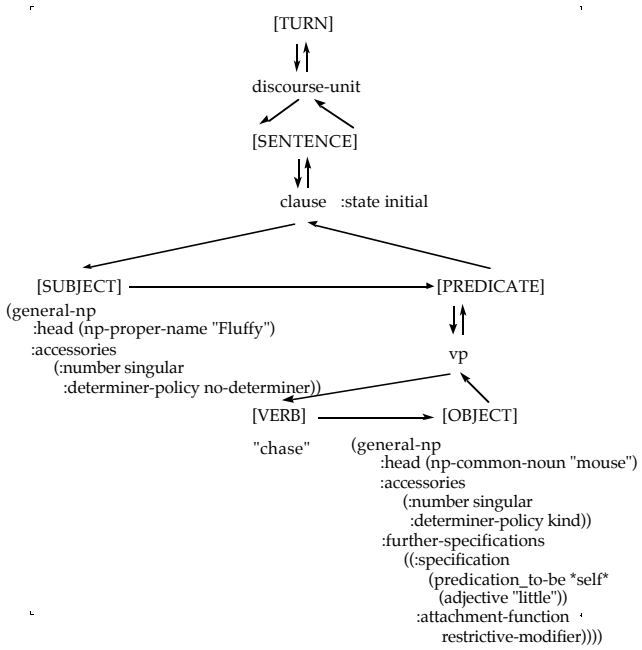
```
(define-realization-class
   TRANSITIVE-VERB\_TWO-EXPLICIT-ARGS (verb agent patient)
1  ((SVO agent verb patient)
    :grammatical-characteristics (clause)
    :required-acccessories (:unmarked))
2  ((SVO-subj-rel agent (agent :trace) verb patient)
    :grammatical-characteristics (relative-clause)
    :argument-characteristics (identical-with-root agent))
3  ((SVO-obj-rel patient agent verb (patient :trace))
    :grammatical-characteristics (clause)
    :argument-characteristics (identical-with-root patient))
4  ((SVO-for-inf agent verb patient)
    :grammatical-characteristics (for-infinitive))
5  ((SVO-for-inf (agent :trace) verb patient)
    :grammatical-characteristics (for-infinitive)
    :required-acccessories (:purpose-clause-object agent))
6  ((SVO-for-inf (agent :trace) verb patient)
    :grammatical-characteristics (for-infinitive)
    :argument-characteristics (available agent))
7  ((SVO-for-inf agent verb (patient :trace))
    :grammatical-characteristics (for-infinitive)
    :required-acccessories (:purpose-clause-object patient))
8  ((SVO-for-inf (agent :trace) verb (patient :trace))
    :grammatical-characteristics (for-infinitive)
    :required-acccessories (:purpose-clause-object patient)
    :argument-characteristics (available agent))
9  ((SVO-subj-whq agent (agent :trace) verb patient)
    :grammatical-characteristics (clause)
    :required-acccessories (:wh agent))
10 ((SVO-obj-whq patient agent verb (patient :trace))
    :grammatical-characteristics (clause)
    :required-acccessories (:wh patient))
11 ((SVO (agent :trace) verb patient)
    :grammatical-characteristics (clause)
    :required-acccessories (:command)))
```


Realization Classes

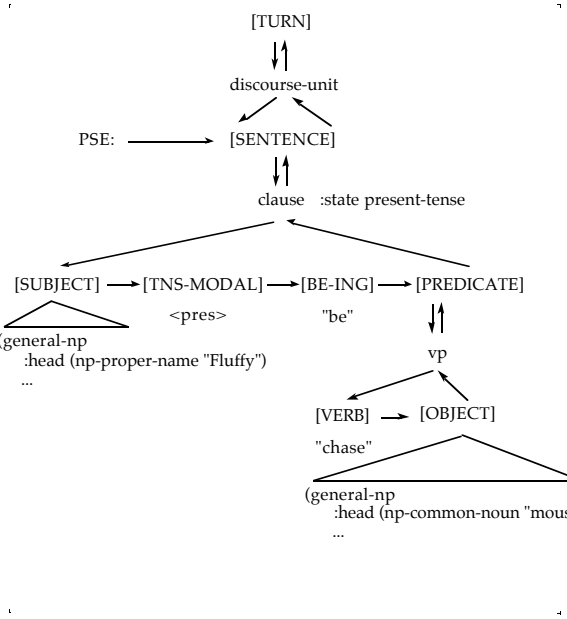
A predefined set of alternatives annotated by the characteristics that distinguish them.

- Grammatical characteristics compared with grammatical constraints on the current position.
- Required accessories compared with the current bundle's accessories.
- First of the remaining phrases is chosen.

Realization of the Head



Processing the Accessories



Realizing the Subject

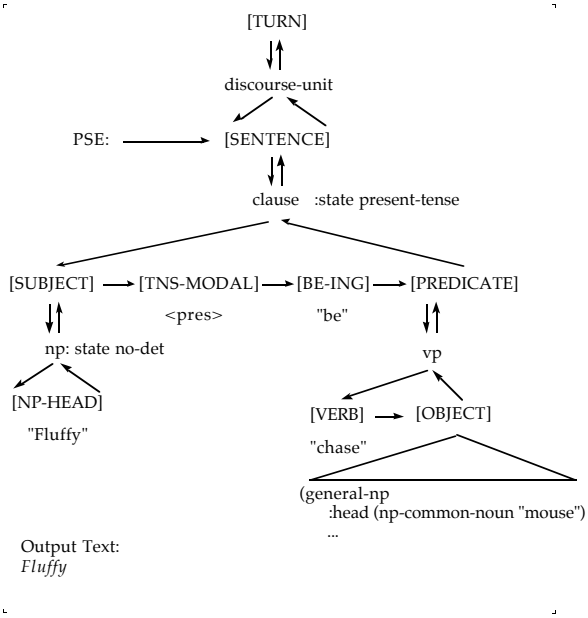
PSE continues from where it was interrupted.

- Reaches the SUBJECT slot.
- Passes control to realization.
- The GENERAL-NP bundle driver is called.
- The driver first checks if pronominalization is required; otherwise follows same strategy as before: realize head, process accessories, attach further specifications.

Realizing the Subject

- Realization here is a single choice.
- Grammatical constraints are checked.
- Processing the accessories involves setting the state of the NP in the phrasal context.
- No more specifications, so the NP is spliced into tree.
- Control is returned to PSE.

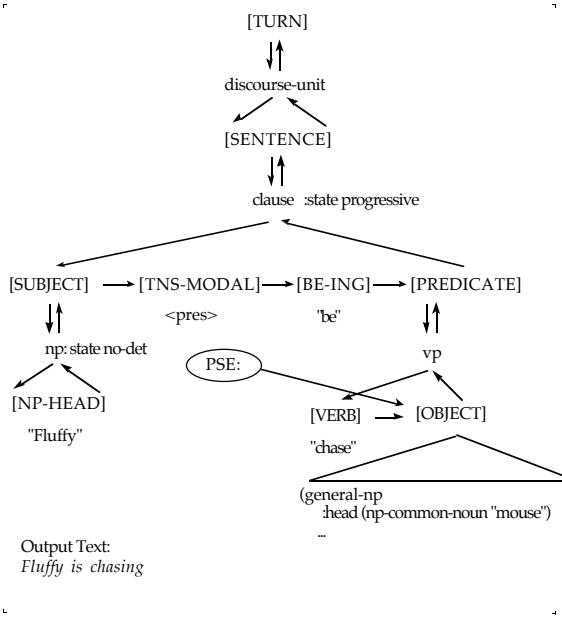
Realizing the Subject



Realizing the Verb Group

- PSE passes through the [TNS-MODAL] and [BE+ING] slots.
- Morphological routines are called to utter the verb group.

Realizing the Verb Group



Realizing the Object

The realization specification:

```
(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))))
```

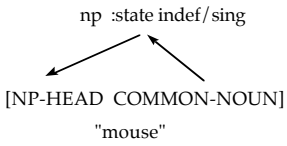

Realizing the Object

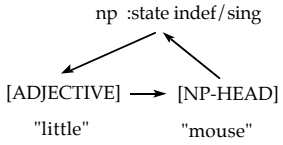
A further specification has two parts:

- a kernel or bundle specification
- an attachment function, which defines the possible points in the surface structure where a specification may be attached.

```
(define-attachment-class
  RESTRICTIVE-MODIFIER ()
    ((ADJECTIVE
     (RESTRICTIVE-APPOSITIVE)
     (RESTRICTIVE-RELATIVE-CLAUSE)
     (NP-PREP-COMPLEMENT)))
```

Realizing the Object

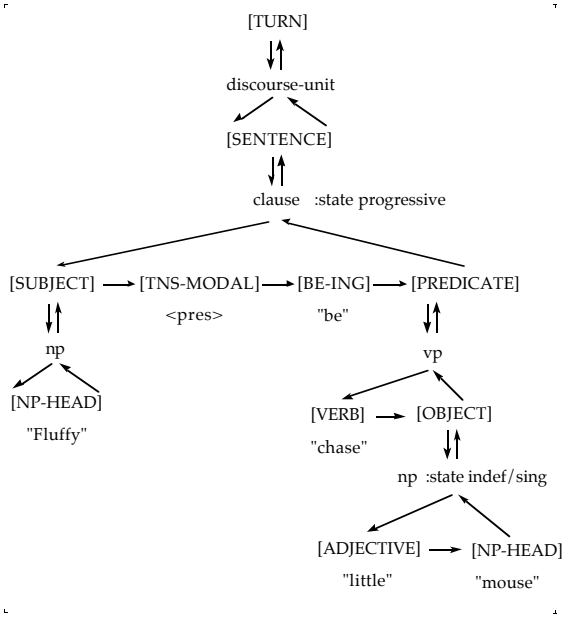




Finishing

- The result of realization is knit into the surface structure.
- PSE takes over from the NP node.
- A determiner is printed.
- The contents of the NP are printed.
- PSE returns to the SENTENCE node.
- The final full stop is printed.

The Final Surface Structure



McDonald's MUMBLE

Summary:

- MUMBLE represents surface structure explicitly and has it interpreted
- conceptual items or item types control the selection and instantiation of the appropriate surface forms directly, through the realization classes that the planner associates with them
- there is no distinct grammar in the sense of a set of rules for deriving linguistic forms from primitive features

McDonald's MUMBLE

- The lack of an explicit grammar can be seen as a deficiency.
- McDonald sees it as a strong hypothesis about the character of linguistic knowledge: the space of valid feature configurations is smaller, less arbitrary and more structured than a feature-heap notation can express.

Overview

- The Nature of the Input
- Unification-based Approaches to Realisation
- Data-driven Approaches to Realisation
- Systemic Functional Grammar
- Towards a Synthesis

Systemic Functional Grammar

1. Background
2. The formalism
3. Its use in NLG systems

Systemic Grammar: General Orientation

- emphasises the FUNCTIONAL ORGANIZATION of language: how language presents speakers with systems of meaningful options as a basis for communication
- 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 Metafunctions

SFG emphasises the use of multiple descriptive dimensions:

ideational: the traditional notion of meaning, as expressed in the transitivity structure of a clause

interpersonal: why the utterance is there: primarily embodied in the mood structure

textual: the glue that holds the communication together, based on information packaging needs

Systemic Grammar: History

- roots in anthropology and sociology
- Firth, Halliday, Hudson

Early themes in the development of systemic grammar:

- What are the social functions of language?
- How does language fulfill these social functions?
- How does language work?

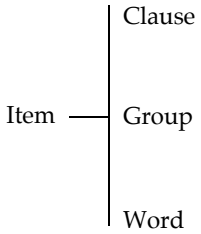
Language as DOING rather than language as KNOWING: LINGUISTIC BEHAVIOUR POTENTIAL as a property of a speech community.

Systemic Grammar: The Formalism

- The grammar is composed of CHOICE SYSTEMS.
- Each system is a set of simultaneous alternatives.
- Each alternative is named; these are referred to as TERMS, OUTPUT FEATURES, or GRAMMATICAL FEATURES.
- Each alternative corresponds to a minimal grammatical alternation.
- Each system has an ENTRY CONDITION.

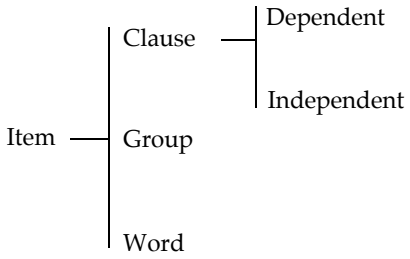
Systemic Grammar: The Formalism

Every item is either a CLAUSE, a GROUP or a WORD:



Systemic Grammar: The Formalism

The selection of one alternative determines what further systems may be entered:

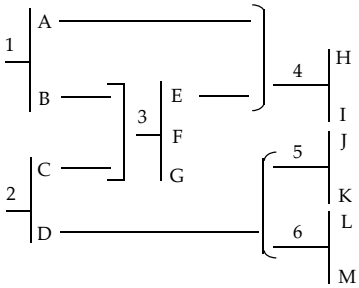


The second system is more DELICATE than the first because it doesn't apply to Groups or Words.

Systemic Grammar: The Formalism

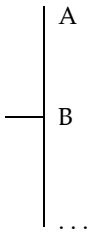
Additional expressive power offered by square OR brackets and round AND brackets:

- facing right means 'select one' or 'select all'
- facing left means one or all of the entry conditions must be satisfied.



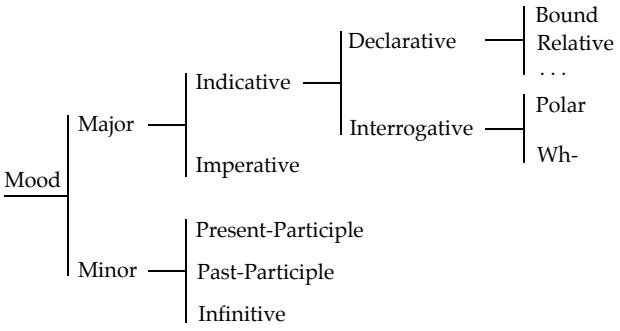
Systemic Grammar: The Formalism

The unmarked choice:



A Systemic Fragment

Mood in the English clause:



Clause Choices

Major Indicative Declarative : The cat is on the mat.

Major Indicative Declarative Relative : [He didn't see the cat] that chased the rat.

Major Indicative Declarative Bound : [It only hurts] when I laugh.

Major Indicative Interrogative Polar : Has anybody seen my gull?

Major Indicative Interrogative Wh- : When will they ever learn?

Major Imperative : Don't be ridiculous.

Minor Present-Participle : [You'll enjoy] having more free time.

Minor Past-Participle : [He had a face] weathered by the years.

Minor Infinitive : [The hard part is] to do it without smiling.

Ranks in Systemic Grammar

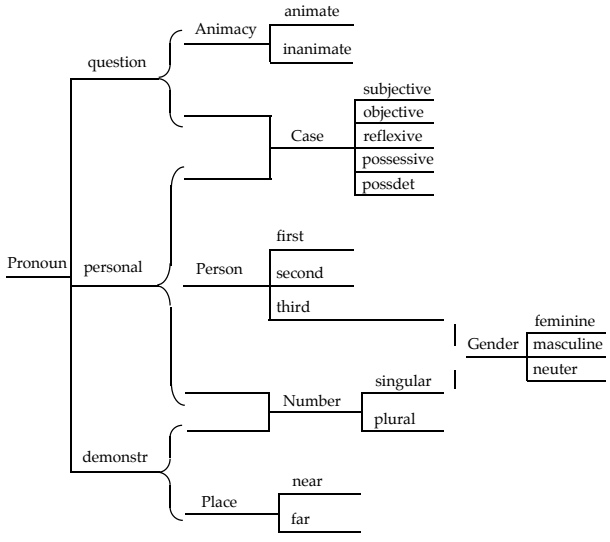
- clause/sentence
- group/phrase
- word
- morpheme

How a Systemic Grammar is Used

- Start with rank of least delicacy
- Make choices until maximally delicate distinctions offered have been drawn
  - result is a complete description of a linguistic unit at that rank
  - this set of features is called the SELECTION EXPRESSION
  - the selection expression will classify a linguistic unit in terms of all three metafunctions
- Repeat for next rank

A Systemic Fragment

Pronominal resources:



A Systemic Fragment

Realization rules		
question animate subjective	→	*who*
question animate objective	→	*whom*
question animate possessive	→	*whose*
question inanimate	→	*what*
demonstr singular near	→	*this*
demonstr singular far	→	*that*
demonstr plural near	→	*these*
demonstr plural far	→	*those*
personal first singular subjective	→	*I*
personal first singular objective	→	*me*
personal first singular reflexive	→	*myself*
personal first singular possessive	→	*mine*
personal first singular possdet	→	*my*
personal second singular subjective	→	*you*
personal second singular objective	→	*you*
personal second singular reflexive	→	*yourself*
personal second singular possessive	→	*yours*
personal second singular possdet	→	*your*
...		
personal third plural possdet	→	*their*

Systemic Grammar: Advantages

Why might systemic grammar be better than immediate constituent approaches?

- may be more natural and economical to state syntactic regularities in a functional framework: a constituent framework may require additional levels of structure to capture functional similarity
- cross-language generalizations may be better stated in functional terms
- the analysis embodies several aspects of meaning: propositional content (transitivity), the speaker's focus and goals (theme), and the discourse context (information structure)

Systemic Grammar

What we still need:

- some way of making choices in systems
- some way of producing text from sets of features

The Development of Nigel

The Nigel grammar = a component of the Penman Text Generation project.

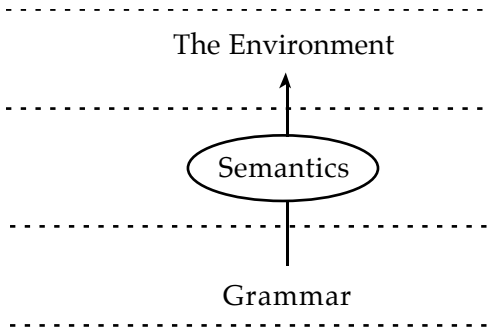
- Penman, a continuation of the work on KDS at ISI, started around 1979–1980
- Nigel was the first part of Penman to be worked on: first version provided by Halliday in 1980
- Subsequently developed by Matthiessen and Bateman and others
- Originally about 80 systems, now contains 600–650 systems

The Nigel Systemic Grammar

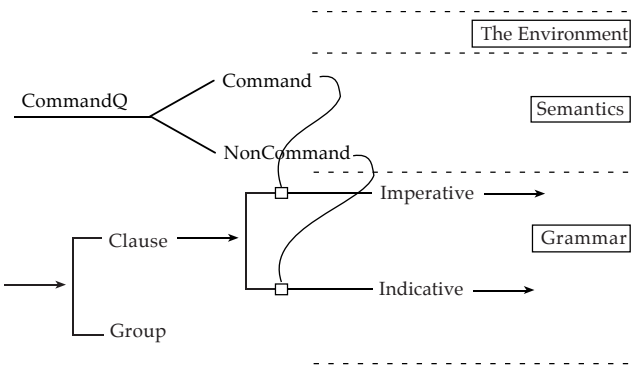
How it works:

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

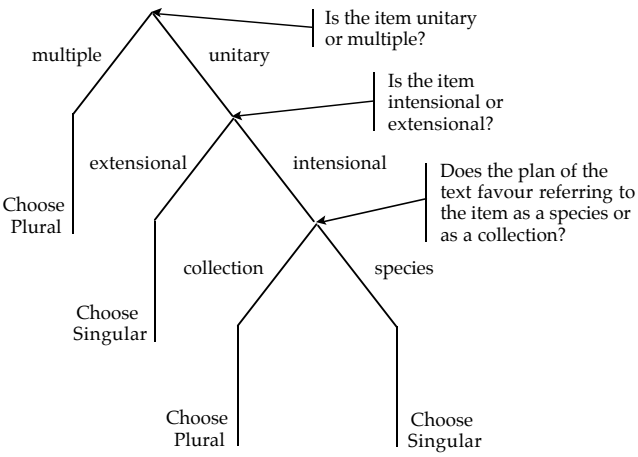
The Strata



Choosers and Inquiries



Choosers



The Nigel Systemic Grammar

NIGEL contains 200–300 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 Nigel Systemic Grammar

Three data structures updated as the grammar is traversed:

- the Selection Expression
- The Function Association Table
- Realization Statements

Realization Operators

Realization rules specify minimal aspects of structural organisation:

- Structure building: Insert, Conflate, Expand.
- Order-constraining: Partition, Order, OrderAtFront, OrderAtEnd
- Associating features with functions: Preselect, Classify, OutClassify, Lexify

Realization Operators

Insert SUBJECT: an element functioning as SUBJECT will be present

Conflate SUBJECT ACTOR: the constituent functioning as SUBJECT = the constituent functioning as ACTOR

Expand MOOD SUBJECT: SUBJECT is a functional constituent of MOOD

Realization Operators

Order FINITE SUBJECT: FINITE must immediately precede SUBJECT

Partition FINITE SUBJECT: FINITE must precede SUBJECT

OrderAtFront SUBJECT: SUBJECT must be the first constituent in the structure being built

OrderAtEnd SUBJECT: SUBJECT must be the last constituent in the structure being built

Realization Operators

Lexify AGENTMARKER by: AGENTMARKER must be realized by the lexical element *by*

Classify PROCESS StateVerb: PROCESS must be realized by the lexical category StateVerb

OutClassify PROCESS StateVerb: PROCESS must not be realized as a StateVerb

Preselect LOCATIVE PrepositionalPhrase: specifies that the lower level constituent LOCATIVE must bear the feature PrepositionalPhrase

Realization Operators

Operator	Abbreviation
Insert F	+F
Conflate F G	F/G
Expand F G	F(G)
Classify F L	F!L
Preselect F P	F:P
Lexify F L	F = L
Order F G	F↑G
OrderAtEnd F	F↑
OrderAtFront F	↑F

Nigel At Work

The sentence to be generated:

> In Greenwich, in South East London, there is a small brick gazebo. **This gazebo was built by Sir Christopher Wren**. It is a rather undistinguished structure, which might have been a task set for homework when he was at school.

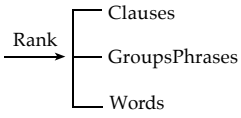
The Function Association Table

ONUS is the function corresponding to hub that is the environment's name for the plan to generate the sentence:

Hubs	Grammatical Functions
WREN-GAZEBO	ONUS

The FAT provides a correspondence between the environment's symbols and the grammar's symbols.

The Rank System



The resulting Selection Expression:

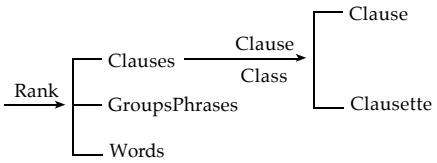
[Clauses]

Hubs	Grammatical Functions
WREN-GAZEBO	ONUS
WREN-GAZEBO-STATEMENT	SPEECH-ACT

If the object has an illocutionary force it will be realized by a clause.

The Grammar

Do we need a major or a minor clause?



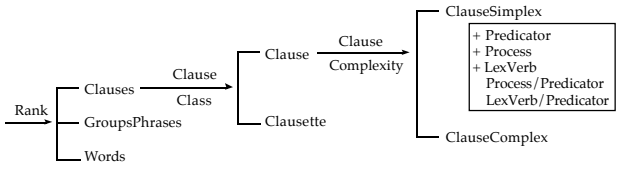
The resulting Selection Expression:

[Clauses, Clause]

Hubs	Grammatical Functions
WREN-GAZEBO	ONUS
WREN-GAZEBO-STATEMENT	SPEECH-ACT
NOW	SPEAKING-TIME

The Grammar

How complex is the plan: one process or many?



The resulting Selection Expression:

Clauses, Clause, ClauseSimplex

The Function Association Table

The same chooser then gets some other information for the FAT:

Hubs	Grammatical Functions
WREN-GAZEBO	ONUS
WREN-GAZEBO-STATEMENT	SPEECH-ACT
NOW	SPEAKING-TIME
GAZEBO-BUILDING	PROCESS
HISTORIC-TIME	EVENT-TIME

The chooser asks what words are denotationally appropriate for the PROCESS in question:

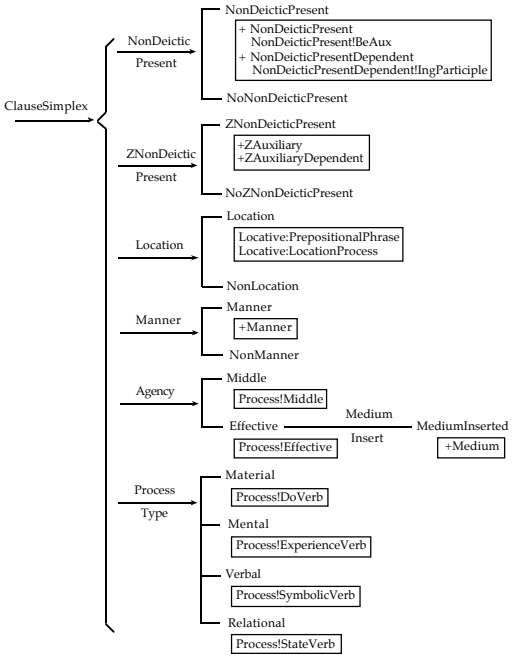
The initial TERMSET for PROCESS:

build	create	construct
builds	creates	constructs
built	created	constructed
built	created	constructed
building	creating	constructing

Realization Statements

+Predicator
+Process
+LexVerb
Process/Predicator
LexVerb/Predicator

The Grammar



Agentivity

+Agent
+Actor
Actor:NominalGroup
Actor/Agent
+AgentMarker
AgentMarker = by
AgentMarker ↑ Agent
+Passive
Passive!BeAux
+PassParticiple
PassParticiple!EnParticiple

The Final Function Association Table

Hubs	Grammatical Functions
WREN-GAZEBO	ONUS
WREN-GAZEBO-STATEMENT	SPEECH-ACT
NOW	SPEAKING-TIME
GAZEBO-BUILDING	PROCESS
HISTORIC-TIME	EVENT-TIME
HISTORIC-TIME	RELEVANT-TIME
SIR-CHRISTOPHER	CAUSER
MEDIOCRITY	MANNER
THE-GAZEBO	GOAL
THE-GAZEBO	MEDIUM
SIR-CHRISTOPHER	ACTOR
SIR-CHRISTOPHER	AGENT
GR-REGION	PARAGRAPH-THEME
WREN-GR-REGION-PATH	AGENT-THEMATIC-PATH
GAZEBO-GR-REGION-PATH	MEDIUM-THEMATIC-PATH
THE-GAZEBO	SUBJECT
THE-GAZEBO	THEME
GAZEBO-BUILDING-POLARITY	POLARITY

Final Set of Realization Statements

+Predicator
+LexVerb
LexVerb/Predicator
+Medium
+Goal
Goal:NominalGroup
+Agent
Actor:NominalGroup
+AgentMarker
AgentMarker↑Agent
Passive!BeAux
PassParticiple!EnParticiple
Medium/Subject
+Mood
Finite!PastForm
+Subject
+Topical
Topical↑
Subject:Nominative
Subject:Singular
Finite!ThirdPerson
+Process
Process/Predicator
Process!Effective
Process!DoVerb
Goal/Medium
Process!Creation
+Actor
Actor/Agent
AgentMarker = by
+Passive
+PassParticiple
LexVerb/PassParticiple
Subject↑Finite
+Finite
Finite/Passive
Mood:Subject
Theme:Topical
↑Theme
Topical/Subject
Finite!Singular

The Final Selection Expression

Clauses, Clause, ClauseSimplex;
NoNonDeicticPresent, NoZNonDeicticTense;
Effective, MediumInserted, NonLocation, Non-Manner, Material;
GoalInsertedConflated, Creative;
Receptive, Agentive;
LexVerbPassPart;
IndependentClause, Indicative;
Declarative, Untagged;
UnmarkedDeclarativeTheme;
Temporal, Past;
Positive, UnmarkedPositive;
NonConsciousSubject, SingularSubject;
LexicalVerbTermResolution

Default Ordering Functions

(Topical Subject Goal Medium)
(Finite Passive)
(PassParticiple Predicator LexVerb Process)
AgentMarker
(Actor Agent)

Result:

Topical + Subject + Goal + Medium was built by Actor + Agent

The Final Utterance

Theme	(Topical ^) ^Theme			
Mood	Subject ^ [Singular, Nominative] Mood	Finite [Singular, PastForm, ThirdPerson]	Predicator	
Transitivity	Goal [NominalGroup] Medium		Process [Effective, DoVerb, Creation] LexVerb	AgentMarker ^ Actor by [NG] Agent
Verbal Voice		Passive [BeAux]	PassParticiple [EnParticiple]	
	This gazebo	was	built	by Sir Christopher Wren

Inquiries

Not just semantic—also pragmatic: for example

- Is it preferable to mention the causative relation between the CAUSER and the PROCESS?
- What is the most salient aspect of the knowledge represented by PROCESS?
- What symbol represents the most salient chain of relationships in the reader's attention and knowledge between MEDIUM and THEME?

Systemic Grammar

What systemic grammars do:

- the networks of a systemic grammar are organized by major syntactic categories
- grammaticality is ensured by forcing the generation process to stay within predefined paths in the systemic network
- the paths define dependencies between abstract text characteristics

The important point:

- individual choices are not simultaneously commitments to the actual utterance, and need not be made in surface order

Systemic Grammar in Other NLG Systems

- Davey's PROTEUS [1972, 1978]: first significant implementation
- Patten's SLANG [1985, 1988]: using AI problem-solving techniques for efficient navigation of systemic networks
- Fawcett's COMMUNAL [1988, 1990]: more explicitly oriented to semantics
- McCoy and Yang [1991]: systemic grammar used to choose TAG structures

Systemic Grammar as a Formalism

- Patten and Ritchie [1986]: a rigorous formal model of systemic grammar
- Mellish [1988]: implementing systemic classification using unification
- Brew [1991]: the computational complexity of systemic classification
- Bateman, Emele and Momma [1991], Carpenter [1991]: implementation of systemic grammar in typed feature structure formalisms

KPML

- Implementation of SFG derived from the Penman/Nigel system
- Available from John Bateman `bateman@darmstadt.gmd.de`

Overview

- The Nature of the Input
- Unification-based Approaches to Realisation
- Data-driven Approaches to Realisation
- Systemic Functional Grammar
- Towards a Synthesis

FUG and SFG

- Many shared assumptions about language and grammar
- Grammatical descriptions organised around feature choices
- Functions of constituents represented explicitly

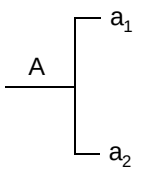
Representing SFG in FUG

- Each SFG unit—eg, clause or group—corresponds to a functional description
- Each FUG attribute corresponds to the description of an SFG feature or function
- Most—but not all—of SFG's elements can be represented in FUG without additional machinery

SFG in FUG

- a systemic choice is represented as a disjunction
- each disjunct contains an attribute with one of the feature choices of that system as its value
- delicacy is represented by embedding

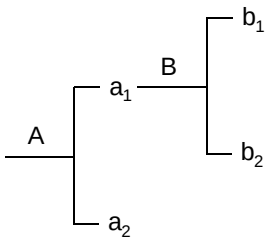
Representing SFG in FUG



$\Longleftrightarrow$

$$\left\{ \left[A: a_1 \right] \left[A: a_2 \right] \right\}$$

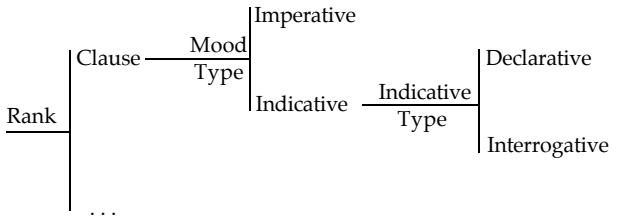
Representing SFG in FUG



$\Longleftrightarrow$

$$\left\{ \left[A: a_1 \left\{ \left[B: b_1 \right] \left[B: b_2 \right] \right\} \right] \left[A: a_2 \right] \right\}$$

MoodType and IndicativeType Systems



$$\left[\begin{array}{l}\text{Rank} = \text{Clause} \\ \left\{\left[\begin{array}{l}\text{MoodType} := \text{Imperative} \\ \text{MoodType} := \text{Indicative} \\ \quad \left\{\begin{array}{l}\left[\text{IndicativeType} := \text{Declarative}\right] \\ \left[\text{IndicativeType} := \text{Interrogative}\right]\end{array}\right\}\end{array}\right]\right\}\end{array}\right]$$

Realisation Operators in FUG

Insertion: a pattern containing the name of the inserted function; no constraint placed on order with respect to other constituents

Preselection: specification of an FD for the given function; constrains one attribute of the function so that it must have the value of the preselected feature

Classify: since there's no formal distinction between lexical features and grammatical features in FUG, preselection and classification are represented in the same way

Order: a pattern that contains the names of the functions in the required order

Conflation: encoded by unification

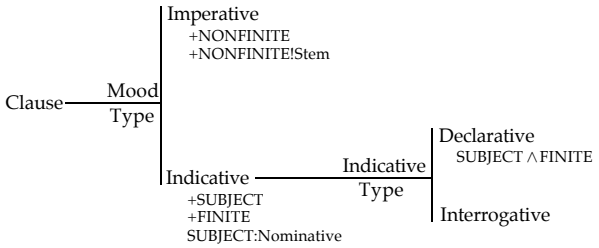
Realisation Operators in FUG

SFG realization	FUG description
Insert +SUBJECT	pattern = (... SUBJECT ...)
Preselect SUBJECT : Nominative	SUBJECT = [Case = Nominative]
Classify FINITE ! Singular	FINITE = [Number = Singular]
Lexify FINITE = has	FINITE = [Lex = has]
Conflate SUBJECT / AGENT	SUBJECT = ⟨AGENT⟩

Order in FUG

Realization Operator	FUG Pattern
Order SUBJECT ∧ FINITE	(... SUBJECT FINITE ...)
Partition FINITE | POLARITY	(... FINITE ... POLARITY ...)
OrderAtFront $^THEME	(THEME ...)
OrderAtEnd TOPICAL^$	(... TOPICAL)

MoodType and IndicativeType Realisations

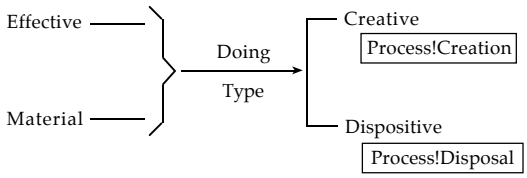


$$\left[\begin{array}{l}\text{Rank} = \text{Clause} \\ \left\{\begin{array}{l}\left[\begin{array}{l}\text{MoodType} := \text{Imperative} \\ \text{pattern} = (\ldots\ \text{NONFINITIVE}\ \ldots) \\ \text{NONFINITIVE} = \left[\text{Form} = \text{Stem}\right]\end{array}\right] \\ \left[\begin{array}{l}\text{MoodType} := \text{Indicative} \\ \text{pattern} = (\ldots\ \text{SUBJECT}\ \ldots) \\ \text{pattern} = (\ldots\ \text{FINITE}\ \ldots) \\ \text{SUBJECT} = \left[\text{Case} = \text{Nominative}\right] \\ \left\{\begin{array}{l}\left[\begin{array}{l}\text{IndicativeType} := \text{Declarative} \\ \text{pattern} = (\ldots\ \text{SUBJECT FINITE}\ \ldots)\end{array}\right] \\ \left[\text{IndicativeType} := \text{Interrogative}\right]\end{array}\right\}\end{array}\right]\end{array}\right\}\end{array}\right]$$

Differences Between SFG and FUG

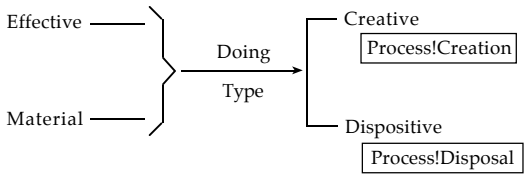
- purpose: formalism *vs* theory
- notational differences:
  - system names are used as attribute labels
  - order operators are expressed as pattern configurations
  - encoding of features and functions

Complex Input Conditions



$$\left[\begin{array}{l}\text{Rank} = \text{Clause} \\ \left\{\begin{array}{l}\left[\begin{array}{l}\text{Agency} = \text{Effective} \\ \text{ProcessType} = \text{Material} \\ \left\{\begin{array}{l}\left[\begin{array}{l}\text{DoingType} := \text{Creative} \\ \text{PROCESS} = \left[\text{Type} = \text{Creative}\right]\end{array}\right] \\ \left[\begin{array}{l}\text{DoingType} := \text{Dispositive} \\ \text{PROCESS} = \left[\text{Type} = \text{Dispositive}\right]\end{array}\right]\end{array}\right\}\end{array}\right] \\ \left[\begin{array}{l}\left\{\begin{array}{l}\left[\text{Agency} = \text{NOT Effective}\right] \\ \left[\text{ProcessType} = \text{NOT Material}\right]\end{array}\right\} \\ \text{DoingType} := \text{none}\end{array}\right]\end{array}\right\}\end{array}\right]$$

Complex Input Conditions



$$\left[\begin{array}{l}\text{Rank} = \text{Clause} \\ \left[\begin{array}{l}\text{Agency} = \text{Effective} \\ \text{ProcessType} = \text{Material}\end{array}\right] \\ \longrightarrow \left\{\begin{array}{l}\left[\begin{array}{l}\text{DoingType} := \text{Creative} \\ \text{PROCESS} = \left[\text{Type} = \text{Creative}\right]\end{array}\right] \\ \left[\begin{array}{l}\text{DoingType} := \text{Dispositive} \\ \text{PROCESS} = \left[\text{Type} = \text{Dispositive}\right]\end{array}\right]\end{array}\right\}\end{array}\right]$$

Conclusions

- Parsing grammars are best indexed by surface structures
- Realisation grammars are best indexed by underlying function or data types
- Different approaches to linguistic realisation are closer than they might at first seem

What's Coming Next ...

1. An Overview of NLG
2. Linguistic Realization
3. Text Planning
4. Generating Referring Expressions