3. Surface realisation

Goal:

to convert text specifications into actual text

Purpose:

to hide the peculiarities of English (or whatever the target language is) from the rest of the NLG system

Tasks:

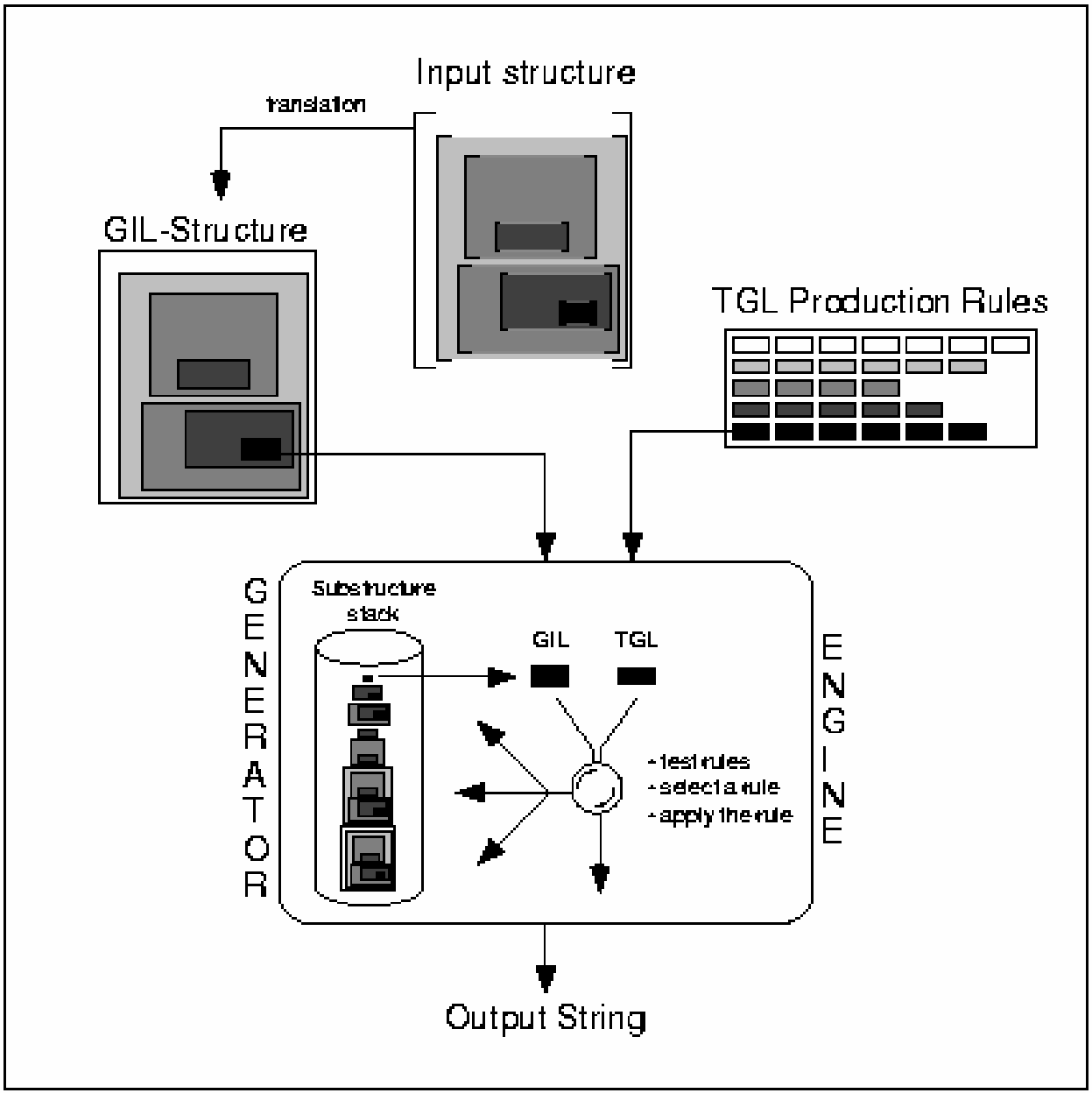
- *Structure realisation*
  - Choose markup to convey document structure
- *Linguistic realisation*
  - Insert function words
  - Choose correct inflection of content words
  - Order words within a sentence
  - Apply orthographic rules

3b. Linguistic realisation - techniques

Use handcrafted templates

- Ex: "shallow" generation with *TG/2* (DFKI) (Busemann, 1996,98)
  - Canned text, templates and context free rules
  - All expressed as production rules whose actions are determined by conditions on input structure (written in TGL)
  - Input structures specified in the *Generation Interface Language* (GIL)
  - Three-step processing cycle as in AI production systems on the available TGL rules:
    - identify all applicable rules,
    - select an applicable rule (e.g. according to preferences),
    - fire that rule
  - Output can easily include formatting elements

TG/2 Overview



A GIL Input Structure

```
[(COOP wertueberschreitung)
           (TIME [(PRED dofc)
                  (NAME [(DAY 31)
                         (MONTH 12)
                         (YEAR 1996)])])
           (POLLUTANT so2)
           (SITE "Völklingen-City")
           (THRESHOLD-VALUE [(AMOUNT 1000)
                             (UNIT mkg-m3)])
           (DURATION [(DAY 30)])
           (SOURCE [(LAW-NAME vdi-richtlinie-2310)
                    (THRESHOLD-TYPE mikwert)])
           (EXCEEDS [(STATUS yes)
                     (TIMES 4)])]
```

A TGL Rule

```
(defproduction threshold-exceeding "WU01"
  (:PRECOND (:CAT DECL
             :TEST ((coop-eq 'threshold-exceeding)
                    (threshold-value-p)))
   :ACTIONS (:TEMPLATE
               (:OPTRULE PPTime (get-param 'time)
               (:OPTRULE SITEV (get-param site)
               (:RULE THTYPE (self)
               (:OPTRULE POLL (get-param 'pollutant)
               (:OPTRULE DUR (get-param 'duration)
               "(" (:RULE VAL (get-param 'threshold-value))
                   (:OPTRULE LAW (get-param 'law-name)) ")"
               (:RULE EXCEEDS (get-param 'exceeds)) "."
             :CONSTRAINTS (:GENDER (THTYPE EXCEEDS) :EQ))))
```

TG/2 Output

On 31-12-1996 at the measurement station at Völklingen-City, the MIK value (MIK-Wert) for sulphur dioxide over a period of 30 days (1000 µg/m³ according to directive VDI 2310 (VDI-Richtlinie 2310)) was exceeded four times.

3b. Linguistic Realisation - techniques

Utilize *grammars* tuned for generation

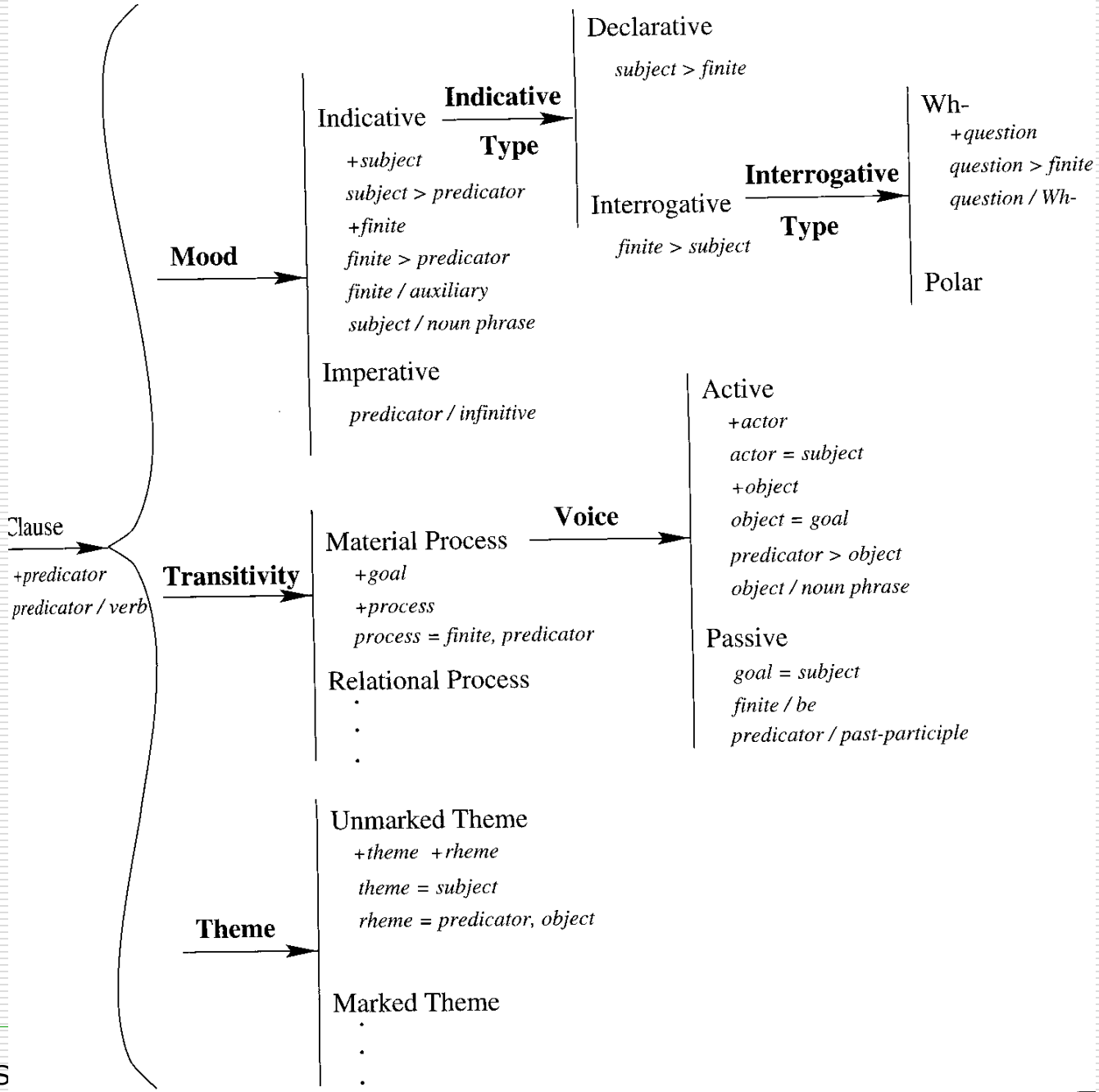
- Provides a set of choices for realisation, made based on input text spec
- Grammar can *only* be used for NLG
- Important approaches
  - Systemic grammar
  - Functional unification grammar
  - Tree-adjoining grammar

Systemic grammar

- Represent sentence as collection of *functions*
- Three kinds of groups of functions
  1. Interpersonal (*mood*): establish and maintain interaction between speaker and listener (e.g. commanding vs telling vs asking)
  2. Ideational (*transitivity*): propositional content
  3. Textual (*theme*): make the expression fit into discourse (theme/rheme, reference)

- Grammar = *system network*, a directed, acyclic graph that maps *functions* to *grammatical form*, e.g. indicative, declaractive, imperative
- Realization statements then map grammatical form onto *syntactic forms*

Simple systemic grammar



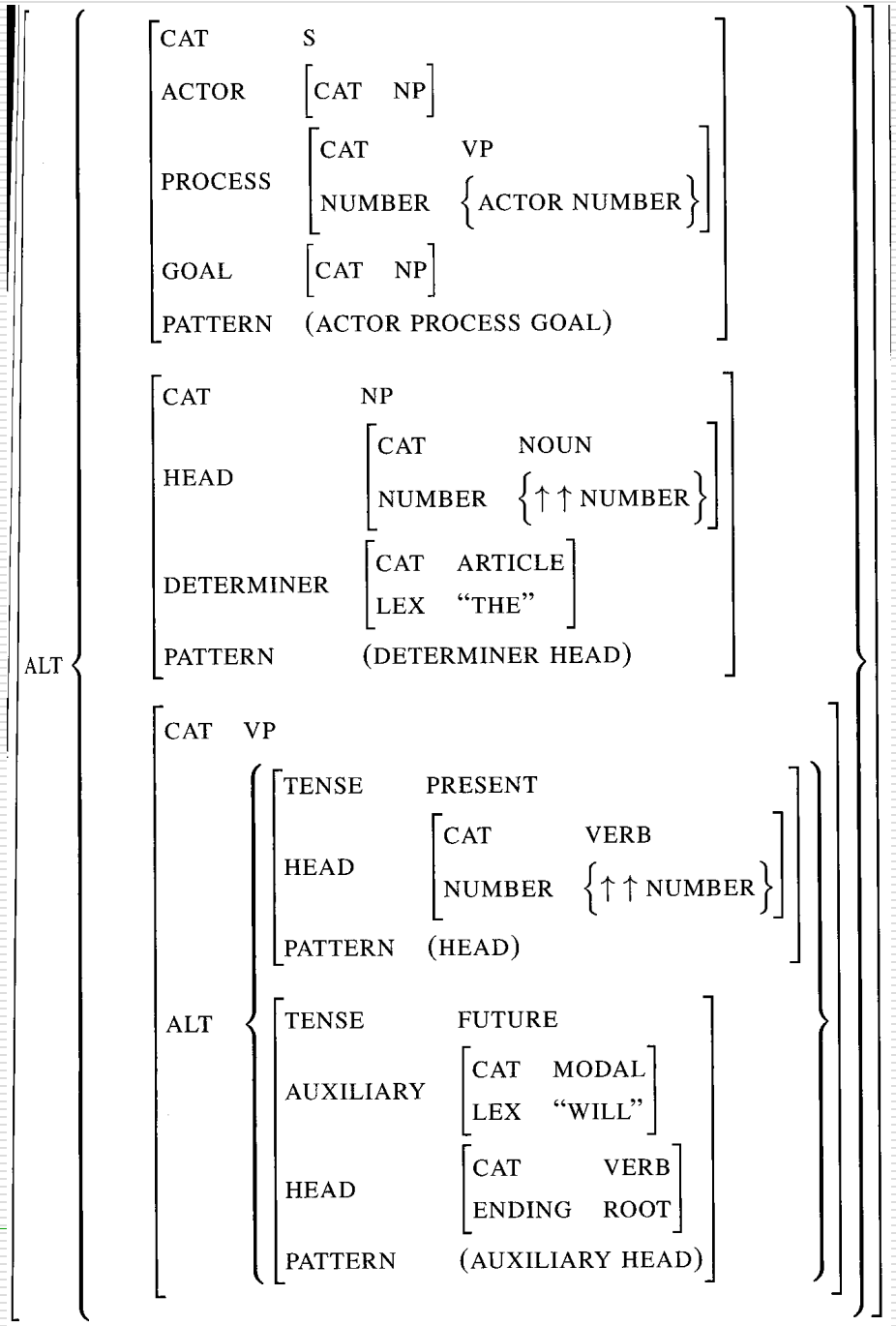
From Jurafsky & Martin (2000)

Functional Unification Grammar (FUG)

- Use *feature structure* with lists of possible alternations and unify it with input spec
- Input structure = *functional description* (FD), feature structure just like the grammar
  - Requires more information than systemic grammar (e.g. tense), more decisions by discourse planning
- Unification produces full feature structure that can be linearized to form sentence output

Simple FUF grammar

FUF = *functional unification formalism*, an implementation of FUG (Elhadad, 1993)

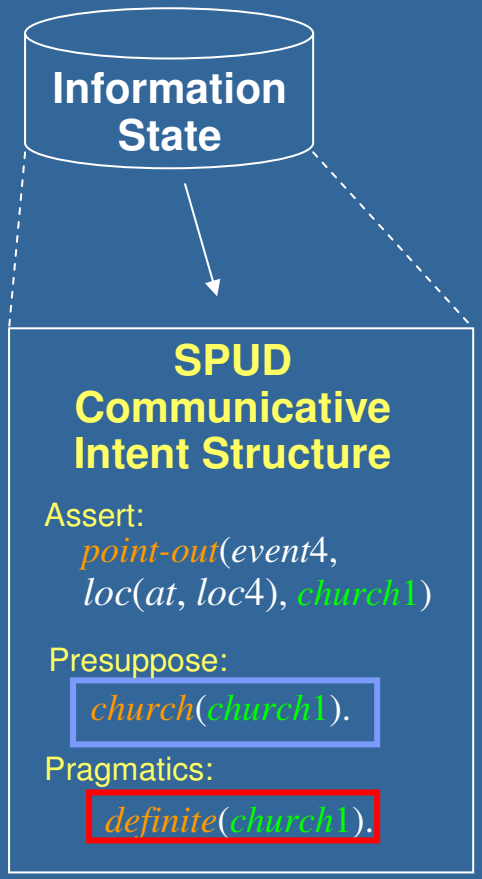


From Jurafsky & Martin (2000)

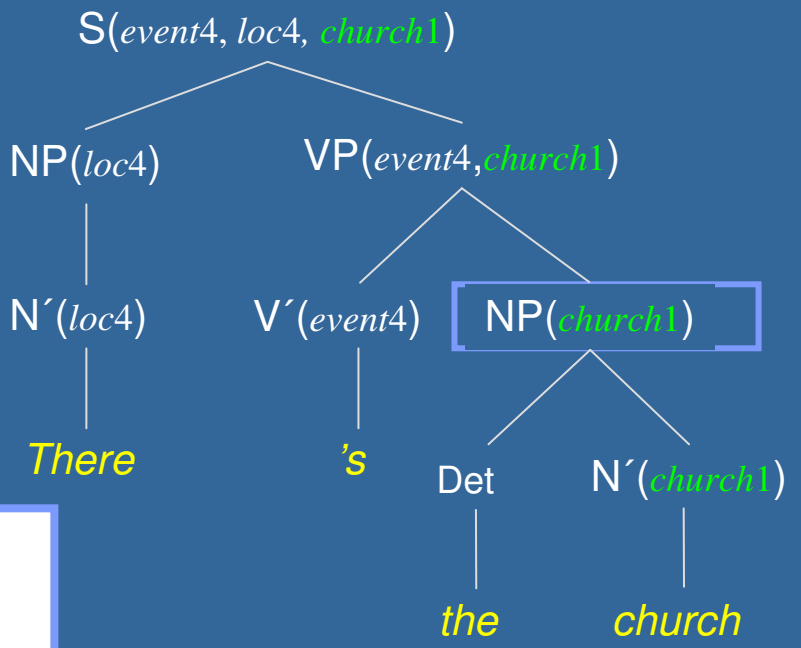
Examples

- Bateman's KPML, uses FUG http://www.fb10.uni-bremen.de/anglistik/langpro/NLG-table/details/KPML.htm
- Stone's SPUD: communicative-intent–based sentence planning http://www.cs.rutgers.edu/~mdstone/nlg.html
  - Given: desired update to conversation
  - Idea: construct utterance with communicative intent that achieves it and can be recognized
  - Use: Search and Constraint Satisfaction to link content of utterances to context
    - Words and grammatical structures are chosen from a lexicon & LTAG grammar to realize communicative goals, constrained by the system's knowledge about context

Snapshot in a SPUD Construction



Derivation Complete!



Lexical Entry

Name: church
Parameters: *X*
Target: NP
Semantics: *church*(*X*)
Pragmatics: none
Tree List: theNPindefiniteTree, theNPdefiniteTree, …