Natural Language Generation

Claire Gardent

CNRS/LORIA
Campus Scientifique,
BP 239,
F-54 506 Vandœuvre-lès-Nancy, France

2005/2006

Outline

What is NLG?

Natural language generation is the process of deliberately constructing a natural language text in order to meet specified communicative goals. [McDonald 1992]

Non linguistic information $\Rightarrow$ NL Text

NLG Input and Output

Input:

- Some underlying non-linguistic representation of information
- Linguistic knowledge (grammar, lexicon, etc.)
- Domain knowledge
- Communicative goal
- User profile (optional)

Output:

- documents, reports, explanations, help messages, etc.

NL Generation vs. Understanding

- NL Generation : Information ⇒ Text
- NL Understanding : Text ⇒ Information
- But NLG is not simply the reverse of NLU.

NL Generation vs. Understanding

- Major issues differ:
  - NLU: how to handle grammatically incorrect or ill-formed input?
  - NLG: how to generate text that is easy to understand?
- Main difficulties differ:
  - NLU: hypothesis management
    Which interpretation is the most plausible one?
  - NLG: Choice
    Which of the different means to achieve is the most appropriate one?
- The input to NLG is often very different from the NLU output

FOG

- Input: Graphical/numerical weather depiction
- Function: Produces textual weather reports in English and French
- Input: Graphical/numerical weather depiction
- User: Environment Canada (Canadian Weather Service)
- Developer: CoGenTex

FOG output

Frobisher Bay.
Winds southwest 15 diminishing to light late this evening. Winds light friday. Showers ending late this evening. Fog.
Outlook for saturday: light winds.

East Breevort and East Davis.
Gale warning continued.
Winds south 30 to Gales 35 diminishing to south winds 15 early friday morning. [...]

PlanDoc

- Function: Produces a report describing the simulation options that an engineer has explored
- Input: A simulation log file for a proposed telephone network modification
- User: Southwestern Bell
- Developer: Bellcore and Columbia University
- Status: Fielded, in operational use since 1996

PlanDocinput

RUNID fiberall FIBER 6/19/93 act yes
FA 1301 2 1995
FA 1201 2 1995
FA 1401 2 1995
FA 1501 2 1995
ANF co 1103 2 1995 48
ANF 1201 1301 2 1995 24
ANF 1401 1501 2 1995 24
END. 856.0 670.2

PlanDoc output

This saved fiber refinement includes all DLC changes in Run-ID ALLDLC. RUN-ID FIBERALL demanded that PLAN **activate fiber for CSAs 1201, 1301, 1401 and 1501** in 1995 Q2. It requested the placement of a 48-fiber cable from the CO to section 1103 and the placement of 24-fiber cables from section 1201 to section 1301 and from section 1401 to section 1501 in the second quarter of 1995. For this refinement, the resulting 20 year route PWE was $856.00K, **a $64.11K savings over the BASE plan** and the resulting 5 year IFC was $670.20K, a $60.55K savings over the BASE plan.

STOP

- Function: Produces a personalised smoking-cessation leaflet
- Input: Questionnaire about smoking attitudes, beliefs, history
- User: NHS (British Health Service)
- Developer: University of Aberdeen
- Status: Undergoing clinical evaluation to determine its effectiveness

STOP output

Dear Ms Cameron
Thank you for taking the trouble to return the smoking questionnaire that we sent you. It appears from your answers that **although you're not planning to stop smoking in the near future, you would like to stop if it was easy. You think it would be difficult to stop because smoking helps you cope with stress, it is something to do when you are bored, and smoking stops you putting on weight. However, you have reasons to be confident of success if you did try to stop**, and there are ways of coping with the difficulties.

What is NLG used for?

- Automated production of routine documents : *weather forecasts, simulation reports, letters, ...*
- Presentation of information to people in an understandable fashion: medical records, expert system reasoning, ...
- Teaching : information for students in CAL systems

Choices in NLG

NLG is a choice problem. In particular, an NLG system will need to make the following decisions

- What to say
- what not to say
- how to organise the content to be verbalised into a coherent discourse
- which tone and degree of formality to adopt
- how to break the content to be verbalised into sentences
- which syntactic construction to use
- how to describe entities
- which words to use

Example

Ce cours est donné par Claire Gardent. C'est un cours sur les applications du TAL.

This text embodies the following decisions/choices:

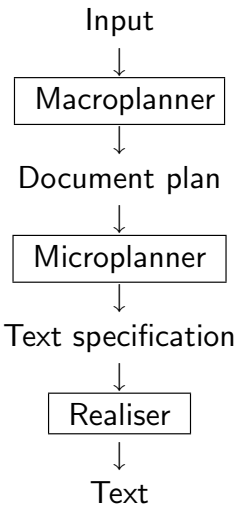
- content: of all things known about the course, it states only the lecturer's name and the topic
- discourse structure: 2 simple sentences rather than one complex one.
- syntax: passive rather than active in the first sentence
- word choice: *donné* rather than *enseigné*
- entity description: uses a pronoun rather than a full Noun Phrase to talk about *ce cours* in the second sentence

The typical NLG modules

Choices are made by various modules:

- Content determination: deciding what to say
- Document structuring: structuring the content into a document plan
- Lexicalisation: choosing words
- Referring expressions: deciding how to refer to entities (pronoun, definite descriptions, etc.)
- Aggregation: mapping the document plan into a discourse plan
- Realisation: mapping abstract sentence representations into sentences

Consensus Architecture



Standard NLG Modules and Interfaces

Macroplanning: Input → Document Plan

- Content determination
- Document structuration

Microplanning: Document Plan → Text specification

- Aggregation
- Referring expressions
- Lexicalisation

Realisation: Text specification → Text

Content Determination

- selects a subset of the input information and maps it into a set of messages (basic informational units to be verbalised)
- selection is determined by e.g.,
  - communicative goal
    describing, defining, explaining
  - end user
    novice, expert
  - constraints
    space restrictions
  - target text
    how much of the input information should be verbalised

Content determination Example

Input:

```
((ENROUTE E1)          ((SHIP K1)              ((DATE T1)
 (ACTOR E1 K1)          (NAME K1 KNOX)          (DAY T1 24)
 (DESTINATION E1 S1)    (READINESS K1 C1))      (MONTH T1 4))
 (NEXT-ACTION E1 A1)   ((PORT S1)              ((DATE T2)
 (LOCATION E1 P1))      (NAME S1 SASEBO))       (DAY T2 25)
((ARRIVE A1)           ((READINESS-STATUS C1)   (MONTH T2 4))
 (ACTOR A1 K1)          (NAME C1 C4))          ((DATE T3)
 (TIME A1 T1)          ((POSITION P1)           (DAY T3 28)
 (NEXT-ACTION A1 L1))   (HEADING P1 H1)         (MONTH T3 4))
((LOAD L1)              (LATITUDE P1 79)
 (ACTOR L1 K1)          (LONGITUDE P1 18))
 (STARTTIME L1 T2)     ((HEADING H1)
 (ENDTIME L1 T3))       (COURSE H1 195))
```

Constructed messages:

(readiness knox c4), (enroute c4 sasebo), (position knox 79n18e), (heading knox ssw)

Final text:

Knox, which is C4, is en route to Sasebo. It is at 79N 18E heading SSW.
It will arrive on 4/24, and will load for four days.

Content determination Process

Content determination is application dependent and is usually carried out using standard AI techniques (e.g., production systems, planning).

Broadly, content determination consists in:

- Specifying a set of message structures i.e., a set of basic informational units to be verbalised
- Specifying a set of message construction rules mapping the input data into messages

Message structures

A message structure

- specifies the basic informational units handled by document structuring
- can be more or less fine grained (one input fact/one message vs. several input facts/one message)
- can be more or less abstract i.e., different from a NL verbalisatioin
- is defined based on the domain ontology (which objects are being talked about? which relations? which properties?)

In practice, message structures are defined by looking at the target texts and by identifying the set of (recurrent) basic informational units which are conveyed by the texts.

Message structure example

```
class TemperatureSpellMsg extends Message {
    period: Month; //Month was defined in Section 3.4
    spell: Interval;
    temperature: one of {extremelyhot, veryhot, hot, verywarm, warm,
        mild, cool, cold, verycold, freezing};
}

class Interval {
    begin: Date;
    end: Date;
}

class Date {
    day: integer;
    month: integer;
    year: integer;
}
```

Figure 4.8 The definition of TemperatureSpellMsg.

Message example

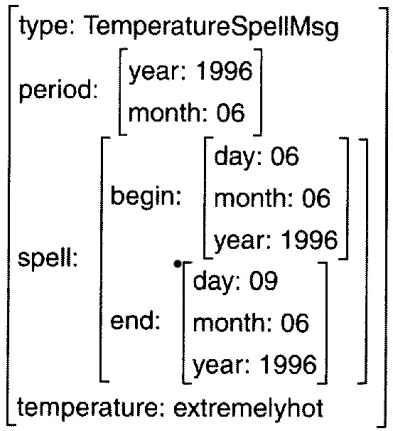


Figure 4.9 A TemperatureSpellMsg message.

WeatherReporter messages

WeatherReporter texts contain two main types of messages:

- Routine messages (always present in reports):
  MonthlyRainfallMsg, MonthlyTemperatureMsg, TotalRainSoFarMsg, MonthlyRainyDaysMsg, TemperatureSpellMsg
- Significant event messages (exceptionnally included in reports):
  RainSpellMsg, TemperatureSpellMsg, RainEventMsg, TemperatureEventMsg

Message construction rules

Message construction rules

- specifies the mapping between input data and message structures
- are defined by looking at the target texts and at the related input data
- involve selecting and/or summarising, abstracting the input data

Selecting Data

When specifying the mapping between input data and message structures, message construction rules select the data to be verbalised based on:

- communicative goal
  description, explanation
- context
  other selected information
- audience
  novice, expert
- constraints
  space restrictions

Data Selection and Communicative Goal

$$\dfrac{\varphi_1 \vee \varphi_2 \qquad \begin{array}{c}[\varphi_1]\\ \vdots\\ \psi\end{array} \qquad \begin{array}{c}[\varphi_2]\\ \vdots\\ \psi\end{array}}{\psi}\ \textit{Case}$$

Communicative goal: Explain

```
(Case-Analysis :Goal ψ :Cases (φ1, φ2))
(Case :Number 1 :Hyp φ1)
(Derive :Reasons (φ1) :Conclusion ψ)
(Case :Number 2 :Hyp φ1)
(Derive :Reasons (φ2) :Conclusion ψ)
(End-Case-Analysis)
```


Data Selection and Communicative Goal

$$\dfrac{\varphi_1 \vee \varphi_2 \qquad \begin{array}{c}[\varphi_1]\\ \vdots\\ \psi\end{array} \qquad \begin{array}{c}[\varphi_2]\\ \vdots\\ \psi\end{array}}{\psi}\ \textit{Case}$$

Communicative goal: Present

```
(Derive :Conclusion ψ :Method Case-Analysis(φ1, φ2))
```


Data Selection and Context

- dependent on context
  - earlier: `(Derive :Conclusion` $a > 0$`)` (*)
  - now:
$$\dfrac{a > 0}{a \neq 0} R$$
  - if (*) immediately before leave $a > 0$ *implicit*:

    Therefore, we have $a \neq 0$.
  - if (*) longer ago make $a > 0$ *explicit*:

    Since $a > 0$, *we have* $a \neq 0$

⇒ *implicit* vs. *explicit* referring expression

Data Selection and User

$$\dfrac{\varphi_1 \quad \cdots \quad \varphi_n}{\psi} R \qquad \dfrac{\varphi'_1 \quad \cdots \quad \varphi'_n}{\psi} R'$$

- user knows R', but not R

```
(Derive :Reasons (φ'1, ..., φ'n) :Conclusion ψ :Method R')
```

Summarizing Data

- when data is too fine-grained
- generalization or abstraction yields the interesting/important information
- e.g., 38 facts describing wind speed and direction between 6am and midnight
  $\Leftrightarrow$ *Winds southwest 15 to 20 knots diminishing to light late this evening*

Abstracting data in *PROVERB*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall S_1.\forall S_2.S_1 \subset S_2 \Leftrightarrow (\forall x.x \in S_1 \Rightarrow x \in S_2)}{\forall S_2.F \subset S_2 \Leftrightarrow (\forall x.x \in F \Rightarrow x \in S_2)}\forall E}{F \subset G \Leftrightarrow (\forall x.x \in F \Rightarrow x \in G)}\forall E}{F \subset G \Rightarrow (\forall x.x \in F \Rightarrow x \in G)} \Leftrightarrow E \qquad F \subset G}{\dfrac{\forall x.x \in F \Rightarrow x \in G}{a \in F \Rightarrow a \in G}\forall E \qquad a \in F}\Rightarrow E}{a \in G}\Rightarrow E$$

$\Rightarrow$

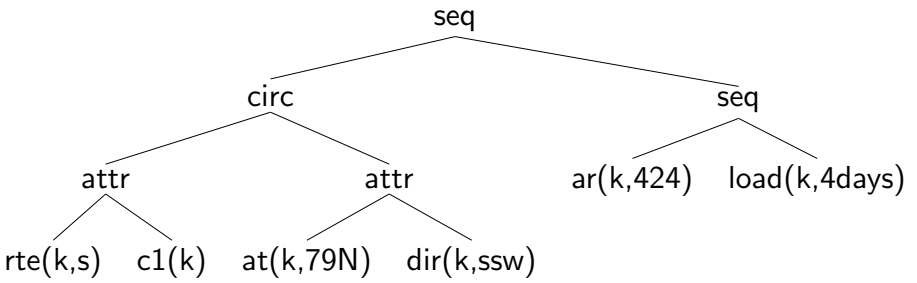
$$\dfrac{F \subset G \quad a \in F}{a \in G}\mathrm{Def}\subset$$

Document structuring

- imposes order and structure over the information to be conveyed
- usually imposes a tree structure on selected set of messages produced by content determination
- non terminal nodes in that tree structure are labelled with structural (*paragraph*, *phrases*) and rhetorical relations

Document plan example

Document plan:



Final text:

Knox, which is C4, is en route to Sasebo. It is at 79N 18E heading SSW.
It will arrive on 4/24, and will load for four days.

Rhetorical Structure Theory

Mann & Thompson, 1988

- RST characterises text structure in terms of relations holding between adjacent spans of text.
- Many functional relations are asymmetric, with one part (the **satellite**, S) providing support for the other (the **nucleus**, N).
- Relations are recursive: the text span serving as **N** or **S** of one relation may be decomposed by another relation into an **N** & **S** of its own.
- Thus RST takes both the local and overall structure of a text to be hierarchical.
- A text is coherent if all parts fit into a single overarching relation

Example

1. P. M. has been with KUSC longer than any other staff member.
2. While attending Occidental College,
3. where he majored in philosophy,
4. he volunteered to work at the station as a classical music announcer.
5. That was in 1970.

Example

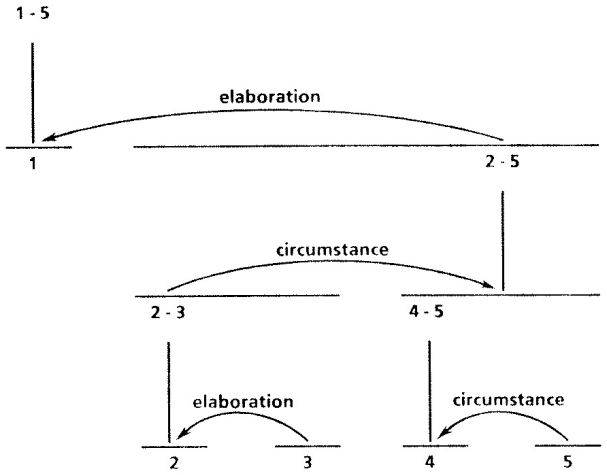


Figure 1-2: RST diagram for "Meet the Announcers" text

RST Relations

Formally, an RST relation :

- holds between two portions of text, the *nucleus* (N, major material) and the *satellite* (S, ancillary material which supports N)
  - S incomprehensible without N
  - S can be replaced by S' to better support N
- is defined by:
  - constraints on the nucleus
  - constraints on the satellite
  - constraints on the combination of the nucleus and the satellite
  - the effect (on the reader)

Rhetorical Relation Example

relation name: Motivation

constraints on N: presents an action on which Hearer is the actor, unrealised with respect to the context of **N**.

constraints on S: –

constraints on N+S: comprehending **S** increases Hearer's desire to perform action presented in **N**.

effect: Hearer's desire to perform the action presented in **N** is increased.

locus of effect: **N**

Types of Rhetorical Relations

Rhetorical Relations come in two flavors:

- **presentational**, whose intended effect is to increase some inclination in the hearer – e.g.
  - **motivation** - the desire to act
  - **evidence** - belief in the info given in **N**
  - **justify** - acceptance of the info given in **N**
  - also **enablement**, **antithesis**, **concession**
- **subject-matter**, whose intended effect is that H recognise that the relation holds in the domain – **e.g.** circumstance, solutionhood, elaboration, background, volitional cause, non-volitional cause, volitional result, non-volitional result, purpose, sequence, contrast, joint

Other RhetRels have been proposed as well.

RST in NLG [Hovy, 1988]

Claim:

Given an initial goal and a set of relevant proposition, RST can be used to organise them into a coherent text.

Operationalising RST Relations for NLG

- Treat the **effect** as the **goal** the plan operator can be used to achieve.
- Treat constraints on **N** and **S** as **preconditions** that are adopted as **sub-goals**.
- Introduce **growth points** as (optional) **plan steps**.

This turns an RST relation into a schema.

Operationalising the SEQUENCE relation

relation name: sequence

goal: ((BMB S H (SEQ-OF ?PART ?NEXT)))

N subgoals: ((BMB S H (TOPIC ?PART)))

S subgoals: ((BMB S H (TOPIC ?NEXT)))

N+S subgoals: ((NEXT-ACTION ?PART ?NEXT))

N growth points: ((BMB S H (CIRCUM-OF ?PART ?CIR))
(BMB S H (ATTRIB-OF ?PART ?VAL))
(BMB S H (PURP-OF ?PART ?PURP)))

S growth points: ((BMB S H (ATTRIB-OF ?NEXT ?VAL))
(BMB S H (DETAILS-OF ?NEXT ?D))
(BMB S H (SEQ-OF ?NEXT ?FOLL)))

order: (N S)

relation-phrases: ("" "then" "next")

(BMB S H P) ⇔ S believes that S & H mutually believed that P.

Structuring Algorithm

Given a set of input units and an agenda consisting of a single goal,

- find an operator that matches a goal on the agenda
- remove main topic from inputs
- add N and S growth points to the agenda

until the agenda or the set of inputs is empty.

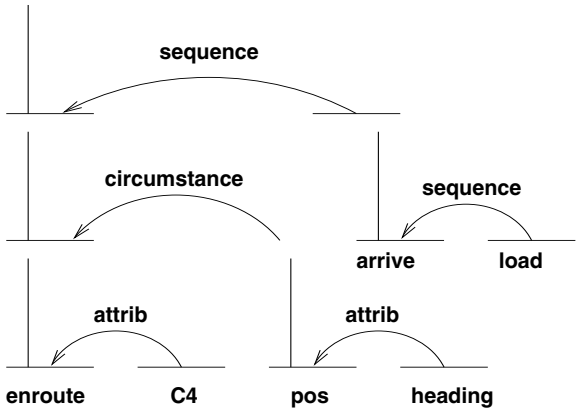
Example derivation

Initial goal: (BMB S H (SEQ-OF E1 ?nxt))
(i.e. S believes that S & H mutually believed that event E1 is followed by some ?nxt event.)

Input:

```
((ENROUTE E1)            ((SHIP K1)               ((DATE T1)
 (ACTOR E1 K1)            (NAME K1 KNOX)           (DAY T1 24)
 (DESTINATION E1 S1)      (READINESS K1 C1))       (MONTH T1 4))
 (NEXT-ACTION E1 A1)     ((PORT S1)               ((DATE T2)
 (LOCATION E1 P1))        (NAME S1 SASEBO))        (DAY T2 25)
((ARRIVE A1)             ((READINESS-STATUS C1)    (MONTH T2 4))
 (ACTOR A1 K1)            (NAME C1 C4))           ((DATE T3)
 (TIME A1 T1)            ((POSITION P1)            (DAY T3 28)
 (NEXT-ACTION A1 L1))     (HEADING P1 H1)          (MONTH T3 4))
((LOAD L1)                (LATITUDE P1 79)
 (ACTOR L1 K1)            (LONGITUDE P1 18))
 (STARTTIME L1 T2)       ((HEADING H1)
 (ENDTIME L1 T3))         (COURSE H1 195))
```

Structuring Algorithm



Possible text: Knox, which is C4, is en route to Sasebo. Knox, which is at 18N 79E, heads SSW. It arrives on 4/24. It loads for 4 days.

More generally

- Schemas are patterns that specifies how a document is constructed from (possibly recursive) discourse constituents
- these patterns may have optional constituents (with conditional tests) and activation conditions which specifies given a context and a set of possible patterns, the pattern to be activated in that context
- a set of schema is aka of text grammar
- well suited for stereotypical portions of a discourse
- Works well only when text structure is predictable and can be captured by predefined patterns
- fails in applications where text structures varies a great deal

Bottom-Up Approach

Let POOL = *messages produced by content determination mechanism*
while (size(POOL)≥1) **do**
  find all pairs of elements in POOL which
    can be linked by a discourse relation
  assign each such pair a desirability score,
    using a heuristic preference function
  find the pair E_i and E_j with the highest preference score
  combine E_i and E_j into a new DocumentPlan E_k,
    using an appropriate discourse relation;
  remove E_i and E_j from POOL and replace them with E_k;
end while

Figure 4.17 A bottom-up discourse structuring algorithm.

Problems

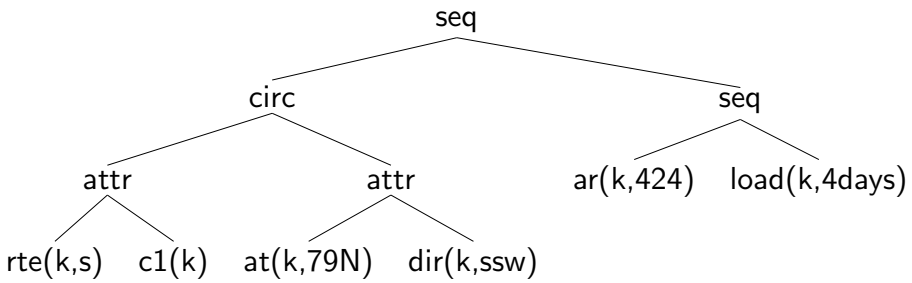
- bottom-up slow due to extensive search
- difficult to create comprehensive set of operators
- conflict resolution needed when several operators applicable
- how introduce paragraphs and sections?

⇒ Hybrid approaches are often used which combine top-down information with bottom-up processing.

Microplanning

- Document Plan ⇒ Linguistic Specification
- Three main subtasks:
  - **Lexicalization:** choice of words/syntactic constructs and mark up annotations
  - **Aggregation:** deciding how much information is communicated by each sentence
  - **Referring expressions:** determining what phrases should be used to identify entities to the user

Example input document plan



Example output linguistic specification

Output linguistic specification (for the leftmost leaf only)

$$\begin{bmatrix} \text{head} & \text{en\_route} \\ \text{features} & \begin{bmatrix} \text{tense} & \text{present} \end{bmatrix} \\ \text{subject} & \begin{bmatrix} \text{head} & \text{Knox} \\ \text{definite} & \text{true} \end{bmatrix} \\ \text{object} & \begin{bmatrix} \text{head} & \text{sasebo} \\ \text{definite} & \text{true} \end{bmatrix} \end{bmatrix}$$

Example

Generated text:

> *The month was slightly warmer than average, with the average number of rain days. Heavy rain fell on the 27th and the 28th.*

Microplanning has ensured the following:

Aggregation. 2 messages in one sentence

Lexicalisation. Average temperature month 2 degrees higher than usual $\Rightarrow$ *slightly warmer than average*

Referring expressions. May 1996 $\Rightarrow$ *the month*

Lexicalization

Lexicalisation maps a domain model based message into linguistic constructs namely, words and syntactic constructions.

Simple lexicalisation using templates

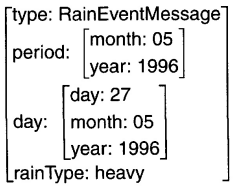


Figure 5.13 A simple message.

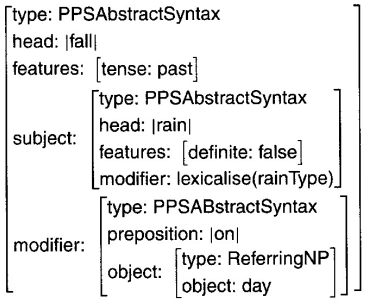


Figure 5.12 A simple template for RainEventMsgs.

The instantiated template

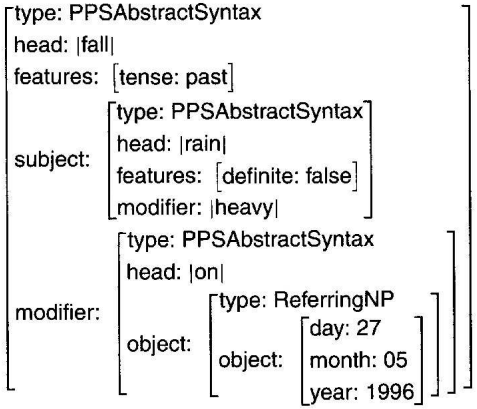


Figure 5.14 The proto-phrase specification produced by applying the template to a message.

A more complex lexicalisation case

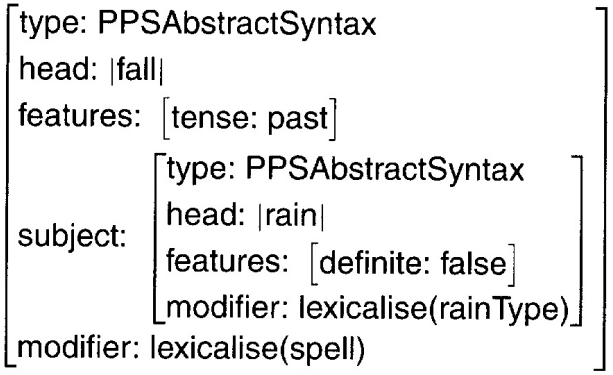


Figure 5.15 A simple template for RainSpellMessages.

A more complex lexicalisation case

A time spell will be lexicalised differently depending on its length:

- from the 9th to the 20th
- on the 9th
- on the 9th and 10th

The spell lexicalisation routine

if spell.begin = spell.end **then**
    return a proto-phrase specification for *on the ith*
else if spell.begin is the day before spell.end **then**
    return a proto-phrase specification for *on the ith and jth*
else
    return a proto-phrase specification for *from the ith to the jth*

Figure 5.16 An algorithm for lexicalising spells.

The linguistic specification for "on the ith"

```
[type: PPSAbstractSyntax
 head: |on|
 object: [type: PPSAbstractSyntax
          head: spell.begin.day
          features: [definite: true
                     inflection: ordinal]]]
```

Figure 5.17 A template proto-phrase specification for *on the ith*.

The linguistic specification for "on the ith and jth"

```
[type: PPSAbstractSyntax
 head: |on|
 object: [type: PPSAbstractSyntax
          head: |and|
          conj1: [type: PPSAbstractSyntax
                  head: spell.begin.day
                  features: [definite: true
                             inflection: ordinal]]
          conj2: [type: PPSAbstractSyntax
                  head: spell.end.day
                  features: [definite: ellided
                             inflection: ordinal]]]]
```

Figure 5.18 A template proto-phrase specification for *on the ith and jth*.

Allowing for syntactic variations

The lexicalisation templates/routine should also allow for variation in the syntactic categories used e.g., NP vs PP:

- (The 12th, 13th and 14th)$_{NP}$ was the wettest three day period seen so far this year
- The wettest three day period seen so far this year was (from the 12th to the 14th)$_{PP}$.

Factors influencing lexicalisation

- Parameter value (e.g., spell length)
- Syntactic context (e.g., subject vs predicative attribute)
- Communicative goal e.g., use enumeration rather than interval to emphasise length of interval
  - *It rained on the 20th, 21st, 22nd, 23rd, 24th, 25th, and 26th.*
- User knowledge e.g., use "Thanksgiving week" only if denotation known to user (difference between US, UK and Canada)
  - *It rained during Thanksgiving week.*

Factors influencing lexicalisation

- Repetition vs Conciseness: repetition helps reducing misunderstanding; conciseness might be necessary in case of space restrictions
  - *It rained for seven days from the 20th to the 26th.*
  - *It rained for seven days from the 20th.*
  - *It rained from the 20th to the 26th.*
- Consistency with previous text
  - *It snowed for five days from the 10th, and it rained for seven days from the 20th to the 26th.*
  - *It snowed for five days from the 10th, and it rained for seven days from the 20th.*

Other mechanisms used for lexicalisation

- Decision trees
- Systemic networks
- Description logic

Decision Tree

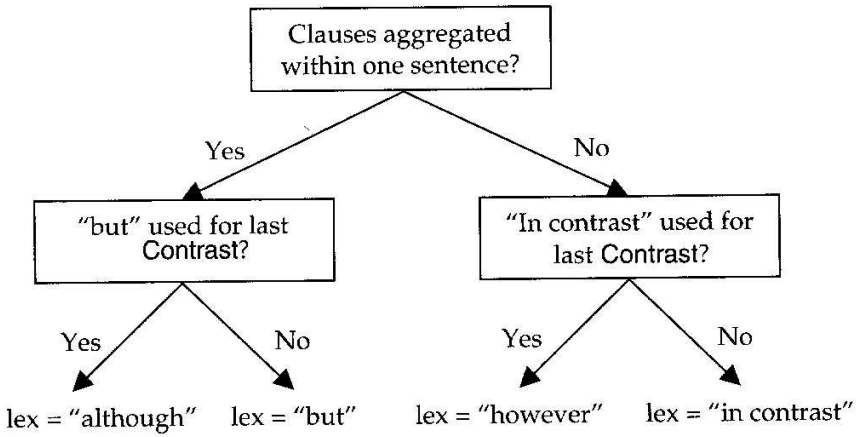


Figure 5.19 A decision tree for realising the Contrast relation.

Lexicalisation using Description Logic

$i : Human, i : Female, (i, d_1) : hasChild, d_1 : Human, d_1 : Female, (d_1, d_2) : hasChild, d_2 : Human$

- What is the best way to describe i?
  - *a female person whose child is a woman who herself has a child*
  - *a grandmother*
- Retrieve set of concepts of which i is an instance
- Use the most specific one and select the corresponding word

Example

Knowledge Base

$GrandMother \Leftrightarrow Woman \sqcap \exists hasChild.Parent$
$Parent \Leftrightarrow (Father \sqcup Mother)$
$Mother \Leftrightarrow Woman \sqcap \exists hasChild.Human$
$Father \Leftrightarrow Man \sqcap \exists hasChild.Human$
$Man \Leftrightarrow Human \sqcap Male$
$Woman \Leftrightarrow Human \sqcap Female$
$Male \Leftrightarrow \neg Female$

Input Data

$i : Human, i : Female, (i, d_1) : hasChild, d_1 : Human, d_1 : Female, (d_1, d_2) : hasChild, d_2 : Human$

Set of concepts of which i is an instance

$Human, Female, Woman, Mother, Parent, GrandMother$

Aggregation

- take a set of simple phrase specifications and combine them to achieve more complex sentence structures
- usually involves restructuring of information (sentence ordering, paragraph formation) to avoid redundancies

Example

Text without aggregation

> *The month was slightly warmer than average. The month had almost exactly the average rainfall. Heavy rain fell on the 27th. Heavy rain fell on the 28th.*

Text after aggregation

> *The month was slightly warmer than average with the average number of rain days. Heavy rain fell on the 27th and on the 28th.*

Types of aggregation

- simple conjunction
- conjunction via shared participants
- conjunction via shared structure
- syntactic embedding

Simple conjunction

Combines two messages into one sentence using sentence coordination.

There was a mild spell from the 5th to the 9th. It was cold at night from the 10th to the 15th.

⇒

There was a mild spell from the 5th to the 9th and it was cold at night from the 10th to the 15th.

Conjunction via Shared Participants

When two entities share argument position and have the same content

The month was colder than average. The month was relatively dry.
⇒ *The month was colder than average and relatively dry.*

January was colder than average. February was colder than average.
⇒ *January and february were colder than average.*

John gave a book to Mary. John gave a book to Fred.
⇒ *John gave books to Mary and Fred.*

Conjunction via Shared Structure

When two messages share non constituent like information.

The month was colder than average. The month was drier than average.
⇒ *The month was colder and drier than average.*

Syntactic Embedding

When a message is realised as a subordinate clause or as a modifier rather than as a main clause.

Example.
The patient's last name is Jones. The patient is female. The patient has hypertension. The patient has diabetes. The patient is 80 years old. The patient is undergoing CABG. Dr. Smith is the patient's doctor.
$\Rightarrow$ *Ms. Jones is an 80-year-old, hypertensive, diabetic, female patient of Dr. Smith undergoing CABG.*

Counterexample.
John is lazy. John is a pianist.
$\not\Rightarrow$ *John is a lazy pianist.*

Referring Expressions

Entities are usually referred to by means of Noun Phrases (NP).
NPs have different forms:

- Indefinite NPs
  - *a cat, some cat*
- Definite NPs
  - pronouns: *he, she, it*
  - Proper names: *The Caledonian Express, Glasgow*
  - Definite descriptions: *The train, the train to Paris*

Referring Expressions

Need to decide how to refer to entities

> *Joe and Jane bought a red and a green apple and a pear in the grocery store. On their way home, Joe ate the pear and the green apple, while Jane had the red one. Joe liked the pear better than the apple. He said, that it was sweeter.*

Usage of Noun Phrases

Roughly,

- indefinite noun phrases are used for initial reference, i.e., when first introduced
- definite noun phrases are used for subsequent reference, i.e., when already introduced

But this is an approximation:

- *Can you tell me where the railway station is?*
- *The station where I boarded this train was deserted.*

Initial Reference

- relatively unexplored
- domain and task dependent
- possible strategies:
  - use full proper name along with relevant properties
    - *Tony Blair, the British Prime Minister, met ...*
  - introduce objects by mentioning their location
    - *You should use the bicycle in the back of the room.*

Subsequent Reference

Trade-off: Avoid ambiguity and Be concise.
Ambiguity:

Take a seat in first or second coach.
? It *is nonsmoking.*

Conciseness:

Take **This train** *vs.*
Take **the train on platform 12 with the red locomotive and the green cars**.

The Role of the Discourse Model

- *potential distractors*: entities that might be mistaken as the ones being referred to
- sources of potential distractors:
  - the immediate physical context
    - must be explicitly modeled
  - the preceding discourse
    - represented in the discourse model: a list of entities mentioned in previous text (sometimes a tree)
    - attentional spaces

Generating pronouns (I)

- refer to recently mentioned entities
- distinguished by
  - gender: *he*, *she*, *it*
  - person: *we*, *you*, *they*
  - number: *it*, *they*
  - case: *he*, *him*, *his*

Constraints on intrasentential Pronouns

- syntactic constraints:
  - ⋆ *The apple had a worm in the apple.*
  - *The apple had a worm in it.*
  - ⋆ *It had a worm in the apple.*
  - *It had a worm in it.*
  - ⋆ *John saw him in the mirror.*
  - *John saw himself in the mirror.*

Intersentential pronouns

- *Rule*:
  IF the intended referent was last mentioned in the previous sentence
  THEN use a pronoun
- Example:
  - *The train is leaving at 5pm.*
    It arrives in Edinburgh at 7pm.
  - *John said the train is leaving at 5pm.*
    He thinks it arrives in Edinburgh at 7pm.
  - *Take a seat in first or second coach.*
    ~~*It*~~ *is nonsmoking.*

Pronominalization Rule (II)

- Rule 2:
  Dale 92: "Generate a pronoun if it is **not ambiguous** i.e., if there is no competing antecedent (i.e. an entity matching in number and gender) in the previous or the current sentence.
- but: too restrictive
  - *Sue invited Mary over for dinner.*
    She cooked her a most amazing meal.
  - *The councillors refused the women a permit because they feared revolution.*
    The councillors refused the women a permit because they advocated revolution.
- In these examples, world KL is involved. Examples involving WKL cannot be handled, hence NLG systems will potentially omit to generate a pronoun when it would be possible.

Looking at real data

Test results: Of 437 ref.expr., 104 were found ambiguous but only 51 of those were realised as pronouns. ([McCoy & Strube 99])
⇒ **The rule is both too restrictive and too permissive:**

- In some cases, even though there are several competing antecedent in the previous sentence, a pronoun can be used.
- In other cases, there is no competing antecedent in the previous sentence but a pronoun is not used.

Other factors influencing pronominalisation

[McCoy & Strube 99] Other factors also play a role namely:

- Sentence boundaries.
- Temporal structure

A newspaper example

1a. Questioned about the criminal activities of the football club,
1b. **Mrs. Mandela** maintained
1c. that **she** had never had any control over them.
2a. This despite testimony from a half dozen former members
2b. that they even had to get permission to go in and out of **her** yard.
3a. **Mrs. Mandela** also said
3b. **she** had disbanded the club
3c. after **her** husband asker **her** to, despite evidence to the contrary.
4.Mrs. Mandela faced questions from lawyers ...
$\Rightarrow$ Although Mrs. Mandela is the focus of every sentence, not all anaphoric references to her are pronouns.

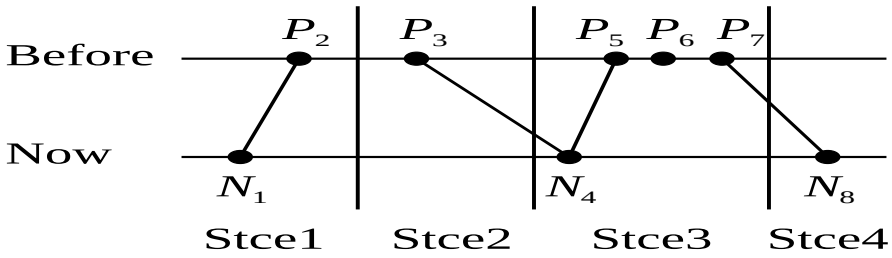
Temporal structure and pronoun use

- Changes in time reliably signal **discourse segment boundaries** (in newspaper articles).
- When a reference to a discourse entity occurs in a **different time plane** than the last reference to that entity, a definite description must be used.
- When a reference to a discourse entity occurs in the **same time plane** than the last reference to that entity, a pronoun must be used.

Long Distance coreference: When a discourse entity has not been referred to for several sentences, a definite description is almost always used regardless of time changes.

Short Distance coreference: When a discourse entity is referred to several times within the same sentence, a pronoun is almost always used regardless of time changes.

Example analysis



	Last mention of MM is in:
P_2	the same sentence (hence PRO)
P_3	the same time plane (hence PRO)
N_4	diff. time plane, diff. stce (hence DD)
P_5, P_6, P_7	the same sentence (hence PRO)
N_8	diff. time plane, diff. stce (hence DD)

Final algorithm

1. If the coreference is a **long–distance one**, use a **definite description**
2. else, if the coreference is **unambiguous and intra–sentential**, use a **pronoun**
3. else, if there is a **time change**, use a **definite description**
4. else, if there is a **competing antecedent**, use the **ambiguity rule**
5. else, use a **pronoun**

Results

Algorithm run on three texts from the NY Times.

- Correct in 370 cases (84.7%)
- Incorrect in 67 cases (15.3%)

Incorrect cases and Future Work

Over–generation of definite descriptions

- Due to the rule that requires a definite description to be used in case of time change.
- Consistent with [Vonk et al. 92] which shows that where a segment boundary is marked by other means (e.g. temporal subordinate clause), pronouns are often used.
- Future work: investigate other markers for segment change.

Over-generation of pronouns

- Due to the intra-sentential rule.
- Suggests that a more sophisticated time analysis is called for.

Generating definite descriptions

The referring expression

- must be adequate, i.e., provide enough information to pick out the intended referent
- must be efficient, i.e., not provide more information than necessary to pick out the intended referent
- should be sensitive to the needs and abilities of the audience

Distinguishing Descriptions (I)

To generate an appropriate referring expression, the microplanner attempts to select a set of properties for the object being described (the target object) such that this set uniquely identifies this object. We call this a *distinguishing description*.

- A *distinguishing description* is an accurate description of the intended referent but not of any object in the contrast set, i.e., the other entities in focus

The problem

Given some intended entity e, how to generate a description D_e which uniquely identifies e?

E	the set of salient entities
$C_e = E - \{e\}$	the contrast–set/set of distractors
$P_e = \{p \mid p(e)\}$	the properties true of e
$D_e \subseteq P_e$	a distinguishing description for e

Distinguishing descriptions

D_e is a **distinguishing description** for e if:

- $\forall p \in D_e \; p(e)$
  All properties in D_e are true of the intended entity e.
- $\forall e_i \in C_e \exists p_i \in D_e \; \neg p_i(e_i)$
  For all other entities in E, there is a property in D_e that does not hold of that entity.

Example

Context

e_1 : ⟨ dog, small, black ⟩
e_2 : ⟨ dog, large, white ⟩
e_3 : ⟨ cat, small, black ⟩

Goal

Generate a distinguishing description for e_1.

Values

E	$\{e_1, e_2, e_3\}$
$C_{e_1} = E - \{e_1\}$	$\{e_2, e_3\}$
$P_{e_1} = \{p \mid p(e_1)\}$	{ dog, small, black }
$D_{e_1} \subseteq P_{e_1}$	$\langle dog, black \rangle$

A set cover problem

The problem of finding a distinguishing description for e can be formulated as a set cover problem.

- RulesOut(p) $= \{e_i \in E \mid \neg p(e_i)\}$
  the set of entities ruled out by p (the entities for which p does not hold).
- D_e is a distinguishing description for e if

$$\cup_{\forall p \in D_e} RulesOut(p) = C_e$$

- D_e specifies a set of *RulesOut* sets that together cover all of C_e.

This is useful because it means that *algorithms* and *complexity results* for set cover problems can be used to generate definite descriptions. In particular, it is known that:

- Finding the minimal size set (i.e. the shortest definite description) cover is NP–Hard [Garey & Johnson 79].
- The **greedy heuristic** algorithm of [Johnson 74] permits finding a *close to minimal* set cover. It is polynomial.

[Dale & Reiter 95]:

- explore how these results can be put to use for generating definite descriptions and
- relate them to cognitive considerations (how plausible are various algorithms as cognitive models of definite description generation?)

Full Brevity

[Dale 89, Dale 92] propose an algorithm that generates the **shortest possible description** through breadth–first search:

- First check whether any one-component description is successful,
- then check whether any two-component description is successful etc.

NP–Hard (because look for the *minimal* set cover).

Greedy Heuristic

Generates the **close to shortest possible description**.
To generate the distinguishing description D_e, do:

1. Initialise C to $C - \{e\}$ and D_e to $\emptyset$.
2. Check success:
   If $\mid C \mid = 0$ **return** D_e
   elseif $P_e = \emptyset$ **then** fail
   else goto step 3.
3. Choose property $p_i \in P_e$ which picks out the smallest set $C_i = C \cap \{x \mid p_i(x)\}$ of distractors.
4. Update D_e to $D_e \cup \{p_i\}$, C to C_i and P_e to $P_e - \{p_i\}$.
   goto step 1.

Example

Context
e_1 : ⟨ large, red, plastic ⟩
e_2 : ⟨small, red, plastic⟩
e_3 : ⟨small, red, paper ⟩
e_4 : ⟨medium, red, paper ⟩
e_5 : ⟨large, green, paper ⟩
e_6 : ⟨large, blue, paper ⟩
e_7 : ⟨large, blue, plastic ⟩

Example

To generate D_{e1}:

- Select plastic $\Rightarrow C = \{e_2, e_7\}$
- Select large (or red) $\Rightarrow C = \{e_7\}$
- Select red (or large)$\Rightarrow C = \emptyset$
- D_{e_1} : ⟨large, red, plastic ⟩
- *the large, red, plastic cup* instead of *the large, red cup*

Incremental algorithm

- in a given domain, we can identify a conventionally useful set of properties to be used in building referring expressions (e.g., location and color for physical objects)
- Iterates through list of properties and adds a property to D_e if it rules out any distractors that have not already been ruled out.
- Terminates when the set of distractors is empty.
- Faster than other algorithms because does not attempt to look for optimal properties (no comparison of distractor sets needed).
- Does not always generate shortest possible description.

Incremental algorithm

To generate the distinguishing description D_e, given a list of attributes A do:

1. Initialise C to $C - \{e\}$ and D_e to $\emptyset$.
2. Check success:
   If $\mid C \mid = 0$ **return** D_e
   elseif $P_e = \emptyset$ **then** fail
   else goto step 3.
3. Choose first property $p_i \in A$ such that it eliminates at least one distractor ($C_i = C \cap \{x \mid p_i(x)\} \neq C$).
4. Update D_e to $D_e \cup \{p_i\}$, C to C_i and P_e to $P_e - \{p_i\}$.
   goto step 1.

Example

attributes: ⟨colour,label,size,shape⟩
Context
b_1 : ⟨ button, red, power, square ⟩
b_2 : ⟨ button, red, reset, round ⟩
b_3 : ⟨ button, black, load, square ⟩
s_4 : ⟨ switch, black, lock, large ⟩

Example

To generate D_{b1}:

- Select button $\Rightarrow C = \{b_2, b_3\}$
- Select red $\Rightarrow C = \{b_2\}$
- Select power $\Rightarrow C = \emptyset$
- D_{e_1} : ⟨red, power, button ⟩
- *the red power button*

Shortcomings

No relational properties: The Dale & Reiter's algorithms only work for flat properties i.e. properties which do not include reference to other entities.
cf. [Dale & Haddock 91, Horacek 96]

No linguistic interface: the distinguishing description is a list of properties not a definite description. As a result, the corresponding definite descriptions might be very akward e.g.
The bottle which is on a table on which there is a cup besides which there is the bottle. (too many relational attributes)
The large, red, speedy, comfortable car (not enough relational attributes, too many attributes)
cf. [Horacek 97, Stone and Webber 98]

Relational properties and nested descriptions

[Dale & Haddock 91]

- The bowl on the table $\rightsquigarrow$ *bowl(b_2) on(b_2,t_1) table(t_1)*
- A recursive process: To describe b_2, describe t_1.
- Algorithm uses the following data–structures:
  - A goal stack e.g. [*describe(e,v)*, ..]
  - A constraint network
    $N = \langle$ Desc, Context–sets $\rangle$
    e.g. $\langle\{$ bowl(x) $\}, (x : \{b1, b2\})\rangle$
- Idea: represent entity e by a variable v associated with a set variable S initially constrained to be a subset of E; Apply the constraints coming from the distinguishing description until the value of S is the singleton set $\{e\}$.

Algorithm

To *describe(e, v)*, do:

1. Initialise Network N to $\langle \emptyset, v : E \rangle$, E is the set of discourse entities.
   Stack $\leftarrow$ [describe(e,v)].
2. If success (the set value of each variable in the constraint network is a dingleton), stop; else go to step 3.
3. Choose property $p_i \in P_e$ such that p_i picks out the network N with the smallest context set for v
4. $p_i \leftarrow [e/v]p_i$
   Extend description with p_i.
   For every constant e' in p_i **do** $p' \leftarrow [e'/v']p_i$
   Push *describe(e',v')* onto stack.
   $N \leftarrow N + p'$
   goto step 2.

Example

Context
$rab(r_1), in(r_1, h_1)$
$rab(r_2), in(r_1, b_1)$
$hat(h_1)$
$box(b_1)$
$hat(h_2)$

Example

The rabbit in the hat
To *describe(r_1,x)*
$P_{r1} = \{$ rab $(r_1), in(r_1, h_1)\}$
$N \leftarrow \langle \emptyset, x : \{r_1, r_2, h_1, h_2\} \rangle$
Choose $rab(r_1)$
$p \leftarrow$ rab(x)
$N \leftarrow \langle \{$ rab(x) $\}, x : \{r_1, r_2\} \rangle$
Choose $in(r_1, h_1)$
$p \leftarrow$ in(x,y)
Push *describe(h_1,y)* onto goal stack.
$N \leftarrow \langle \{$ rab(x), in(x,y) $\}, (x : \{r_1, r_2\}, y : \{h_1, b_1\}) \rangle$
To *describe(h_1,y)*
Choose hat(h_1)
$p \leftarrow$ hat(y)
$N \leftarrow \langle \{$ rab(x) in(x,y) hat(y) $\}, (x : \{r_1\}, y : \{h_1\}) \rangle$

Infinite descriptions

$$P_1 = \{rab(r_1)hat(h_1)in(r_1, h_1)\}$$

$\rightsquigarrow$ *the rabbit in the hat*
$\rightsquigarrow$ *the rabbit in the hat containing the rabbit in the hat containing*
...
Possible solutions

- If a two-place predicate is used in describing an entity, the second object of this predicate is itself only described in terms of *unary* properties (restricted recursion) [Novak 88].
- Do not describe an entity in terms of entities already mentioned in the description built so far [Davey 79].

Problem
The$_i$ man who ate the cake that poisoned him$_i$

- Do not use the same attribute more than once.

Realizer

- synonyms: *surface generator*, *surface realizer*, *linguistic realizer*
- takes linguistic specification
  - Tree where leaves are labelled with phrase specifications and internal nodes with information about discourse relations, section/paragraph grouping and title/subtitle info.
- outputs the surface text i.e., a sequence of words, punctuation symbols and markup annotations

Linguistic vs. Structure Realization

A text specification contains both phrase specification and information about document structure. Each is used differently:

- *Linguistic Realization*: maps phrase specifications into surface-form sentences or sentence fragments i.e, words and punctuation symbols
- *Structure Realization*: maps document structure information into appropriate annotations for the document presentation system i.e., adds annotation markup symbols and possibly punctuation ones.

Example text specification

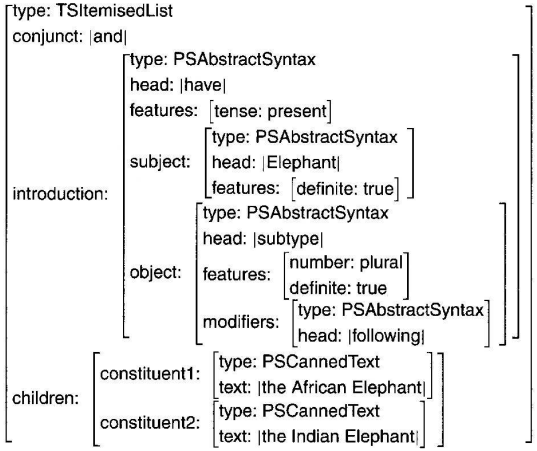


Figure 6.1 A simple PEBA text specification.

Output text

The Elephant has the following subtypes:

- the African Elephant; and
- the Indian Elephant.

Figure 6.3 The PEBA text as displayed by the presentation system.

Structure realisation

- *Structure realisation* reduces to producing a logical structure for the text
- This logical structure is mapped to the text physical structure by a document presentation system (latex, HTML-based web browser, etc.)

Linguistic Realization

- mapping from linguistic specification to surface text
- is more or less complex depending on input linguistic specification

Varieties of phrase specifications

NLG systems differ in the type of phrase specifications they assume for linguistic realisation. The most common types are:

- Orthographic strings : ready to go!
- Canned text : orthographic processing
- Abstract syntactic structures : morphosyntactic and syntactic processing need to be done; only one syntactic realisation possible
- Lexicalised case frames : morphosyntactic and syntactic processing need to be done; several syntactic realisations are possible
- Meaning specifications : lexicalisation, morphosyntactic and syntactic processing need to be done
- Skeletal propositions : referring expressions, lexicalisation, morphosyntactic and syntactic processing need to be done

Skeletal propositions

The courier delivered the green bicycle to Mary.

- Skeletal proposition: $deliver(c1, p1, m)$
- basic content is determined but
- additional content need to be computed to determine referring expressions
- lexicalisation still needs to take place
- Example : SPUD (Matthew Stone et al.)

Meaning Specifications

deliver(c1,p1,m) $\wedge$ *isa(c1,courier)* $\wedge$ *isa(p1, bicycle)* $\wedge$ *has_color(p1, green)* $\wedge$ *has_name(m, Mary)*

- All content is determined (the realiser must produce a text that realises all and only the meaning described by the meaning specification)
- lexicalisation still needs to take place

Lexicalized Case Frames

Semantic predicates have been mapped into lexemes. Several syntactic realisations still possible.

$$\begin{bmatrix} \text{head: } |\text{deliver}| \\ \text{participants: } \begin{bmatrix} \text{arg1: } [\text{head: } |\text{courier}|] \\ \text{arg2: } \begin{bmatrix} \text{head: } |\text{bicycle}| \\ \text{mod: } |\text{green}| \end{bmatrix} \\ \text{arg3: } [\text{head: } |\text{Mary}|] \end{bmatrix} \end{bmatrix}$$

Figure 6.8 A lexicalised case frame.

Abstract Syntactic Structures

Only one realisation possible.

$$\begin{bmatrix} \text{head: } |\text{deliver}| \\ \text{subject: } [\text{head: } |\text{courier}|] \\ \text{object: } \begin{bmatrix} \text{head: } |\text{bicycle}| \\ \text{modifier: } [\text{head: } |\text{green}|] \end{bmatrix} \\ \text{indirectobject: } [\text{head: } |\text{Mary}|] \end{bmatrix}$$

Figure 6.9 An abstract syntactic structure.

Abstract Syntactic Structures

- Basic grammatical properties have been specified
- Content words need to be added
- Inflections must be determined
- The linear order of the constituents must be fixed

Canned Text

[text: |the courier delivered the green bicycle to Mary|]

Figure 6.10 A canned text structure.

- Surface form of the words and phrases have been determined
- Still needs to be processed orthographically (Upper case at beginning of sentence; sentence final punctuation needed; bold or italic to indicate emphasis, etc.)

Orthographic Strings

- final surface form of the text including punctuation and capitalization is specified
- No need for linguistic realisation

[body: |*The courier delivered the green bicycle to Mary.*|]

Figure 6.11 An orthographic string structure.

Realizers

- KPML –based on systemic grammar (Halliday)
- SURGE –based on systemic grammar
- REALPRO –based on Meaning-Text Theory (Melcuk)
- TAG-GEN – based on Tree Adjoining Grammar
- GENI – based on Tree Adjoining Grammar

GenI

- Based on a Tree Adjoining Grammar
- The same grammar is used both for analysis and for generation
- The input is either a lexicalised case frame (several possible outputs) or a an abstract syntactic structure (only one output possible)

Proposal

- a generative approach: the grammar covers (generates) all paraphrases
- optimised to reduce the combinatorics early on in the surface realisation process (by drastically reducing the search space before search starts)
- and selective: supports the selection of those paraphrases that conform to the given contextual restrictions

Overview

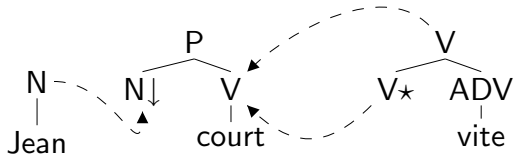
- The grammar (a Feature Based Tree Adjoining Grammar)
- The basic surface realisation algorithm
- Optimisations
- Selection
- Implementation and experimentation
- Related approaches
- Future and related work

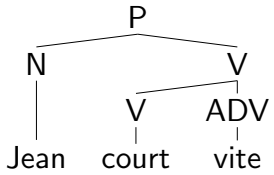
The grammar – the syntactic dimension

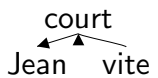
Lexicalised Feature Based Tree Adjoining Grammar (FTAG)

- Set of trees (initial and auxiliary)
- Each tree is anchored with a word
- The nodes of the trees are labelled with two feature structures (TOP and BOTTOM)
- Two combining operations : substitution and adjonction

Example





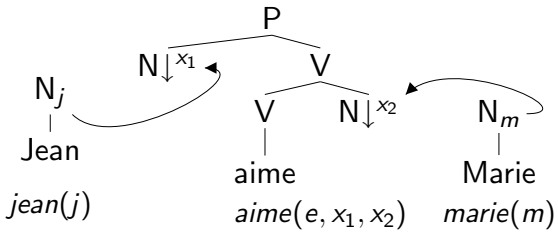


The grammar – adding a semantic dimension

Unification based semantic construction

- The trees are associated with semantic representations in which the semantic parameters are unification variables
- The (appropriate) tree nodes are labelled with semantic parameters
- The semantic of a derived tree is the union of the semantic representations of the trees entering in its derivation modulo unification

Example



$\Rightarrow jean(j), marie(m), aime(e, j, m)$

Linguistic coverage

- Basic subcategorisation frames
- Redistributions : active, passive, middle, reflexive, impersonal, impersonal passive
- Argument realisation : cliticisation, extraction, ommissions, word order variation

Surface realisation algorithm

- Tabular and bottom-up
- + Optimisations
- + Parameterisation for paraphrase selection

Basic algorithm (simplified)

1. Input: the grammar (G), a semantic representation (Sem)
2. Declaration : Chart, Agenda, AgendaA $\leftarrow 0$
3. Initialisation of the agenda : all trees in G whose semantic subsumes part of Sem are added to the agenda
4. Processing the agenda (substitutions) : for each tree I in Agenda which can combine by substitution with a tree J in Chart, add IJ to Agenda ; the trees with no empty substitution node but with a foot node are moved to AgendaA
5. Reinitialisation : Agenda $\leftarrow$ Chart, Chart $\leftarrow$ AgendaA
6. Processing of agenda (Adjunctions)
7. Output: all the strings which are the yield of a syntactically complete tree whose semantic is Sem

GenI: a TAG based surface realiser

- Tabular and bottom-up
- + Optimisations
- + Parameterisation for paraphrase selection

Algorithm outline

Three main steps:

Lexical selection step: Select trees whose semantics subsumes the input semantics.

Realisation step: Perform substitutions or adjunctions between selected.
Substitutions are applied first, then adjunctions

Success lookup step: Return the trees which are syntactically complete and whose semantics matches the input semantics.

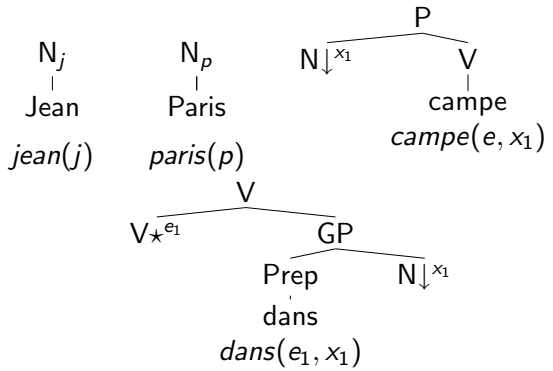
Algorithm (simplified)

1. Input: the grammar (G), a semantic representation (Sem)
2. Declaration : Chart, Agenda, AgendaA $\leftarrow 0$
3. Initialisation of the agenda : all trees in G whose semantic subsumes part of Sem are added to the agenda
4. Processing the agenda (substitutions) : for each tree I in Agenda which can combine by substitution with a tree J in Chart, add IJ to Agenda ; the trees with no empty substitution node but with a foot node are moved to AgendaA
5. Reinitialisation : Agenda $\leftarrow$ Chart, Chart $\leftarrow$ AgendaA
6. Processing of agenda (Adjunctions)
7. Output: all the strings which are the yield of a syntactically complete tree whose semantic is Sem

Example

$$Sem = \{campe(s,j), jean(j), dans(s,l), paris(l)\}$$

Lexical lookup phase



Substitutions

Substitutions

Agenda	Chart	Combination	AgendaA
Jean,campe,dans,Paris			
campe,dans,Paris	Jean		
dans,Paris,JeanCampe	Jean, campe	↓(campe,Jean)	
Paris,JeanCampe	Jean,campe,dans		
JeanCampe, dansParis	Jean,campe,dans,Paris	↓(dans,Paris)	
dansParis	Jean,campe,dans, Paris,JeanCampe		
	Jean,campe,dans, Paris,JeanCampe		dansParis

Example (Ct'ed)

Adjunctions

Agenda	Charte	Combination	AgendaA
Jean,Paris,JeanCampe	dansParis		
Paris,JeanCampe	dansParis,Jean		
JeanCampe	dansParis,Jean,Paris		
	dansParis,Jean,Paris JeanCampe	⋆(JeanCampe, ,dansParis)	
JeanCampeDansParis	dansParis,Jean,Paris, JeanCampe		
	dansParis,Jean,Paris, JeanCampe		
	JeanCampeDansParis		

Optimisations

- Substitutions ⟨ Adjunctions
- Elimination of redundant structures
- Polarity based filtering

Multiple modifiers

```
fierce(x),little(x),cat(x),black(x)
```

For n modifiers, $n!$ intermediate structures :

fierce cat, fierce black cat, little cat,little black cat, fierce little cat, black cat

multiplied by the context :
fierce cat, fierce black cat, little cat,little black cat, fierce little cat, black cat
the *fierce cat, the fierce black cat, the little cat, the little black cat, the fierce little cat, the black cat*
the *fierce cat* **runs**, *the fierce black cat runs, the little cat runs, the little black cat runs, the fierce little cat runs, the black cat runs*

Substitutions ⟨ Adjunctions

Adjunction restricted to syntactically complete trees
The $n!$ intermediate structures are not multiplied out by the context :
the cat runs
the fierce cat runs, the fierce black cat runs, the little cat runs, the little black cat runs, the fierce little cat runs, the black cat runs

Elimination of redundant structures

The same syntactic structure can be constructed in different ways :

- Distinct relative ordering of substitutions within a tree
- Distinct relative ordering of adjunctions within a tree
  ⇒ Only one operation order allowed (left to right)
- Distinct relative ordering of multiple adjunctions to a given node
  ⇒ No adjunction on foot node

Polarity based filtering

- The search space created by the lexical lookup phase is exponential in the number of literals present in the input semantics
- Nb of possible combinations : $\prod_{1 \leq i \leq n} a_i$ avec :
  a_i, the degree of lexical ambiguity of the i-th literal and
  n, the number of literals in the input semantics.
- Polarity based filtering filters out all combinations of lexical items which cannot result in a grammatical structure

Example (Ct'ed)

- The grammar trees are associated with polarities reflecting their syntactic resources and requirements
- All combination of trees covering the input semantics but whose polarity is not zero is necessarily syntactically invalid and is therefore filtered out.
- A finite state automata is built which represent the possible choices (transitions) and the cumulative polarity (states)
- The paths leading to a state with polarity other than zero are deleted (automata minimisation)

Ambiguity

Many combinations are syntactically incompatible. The goal is to detect these combinations and filter them out.
Semantic Representation : `tableau(t)`, `cout(t,g)`, `grand(g)`

Lexical look up :

`tableau(t)`	`cout(t,g)`	`grand(g)`
$\tau_{tableau}$	τ_{cout}	$\tau_{est\ eleve}$
$\tau_{peinture}$	τ_{coute}	τ_{cher}

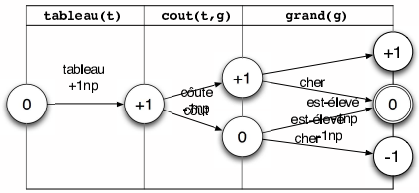
Le tableau coûte cher

Le coût du tableau est élevé

* $\tau_{peinture}$, τ_{coute}, $\tau_{est\ eleve}$

Polarity filtering (Perrier 2003)

We associate each tree in the grammar with a set of polarities representing syntactic resources and requirements.
We construct an automaton to represent the possible combinations of lexical items and their polarities.



We then perform automaton minimisation to remove the incompatible combinations.

Paraphrase selection

- The generator can be parameterised by one (or more) restrictor(s)
- Restrictor ::= `-Synt:SemIdex`
- The grammar trees are (automatically) associated with polarities of the form `+Synt:SemIdex`
- Polarity based filtering eliminate all tree combinations which fail to satisfy the property expressed by the restrictor

Exemple

```
regarde(e,j,m), jean(j), marie(m)
-cleft:j                          C'est Jean qui regarde Marie
-declarative:e                    Jean regarde Marie
-interrogative:e                  Jean regarde-t'il Marie?
```


Implementation and Experimentation

- Implemented in Haskell (Carlos Areces, Eric Kow)
- Graphic interface
- Debugging and testing facilities (batch processing, step-wise visualisation of the different data structures)

Implementation and Experimentation

Test cases of Carroll et al. 1999 and Koller and Striegnitz 2002.

(1) The manager in that office interviews a new consultant from Germany.
Le directeur de ce bureau auditionne un nouveau consultant d'Allemagne.

(2) The manager organizes an unusual additional weekly departmental conference.
Le directeur organise un nouveau seminaire d'equipe hebdomadaire special.

Polarity filtering and Paraphrase selection results

Grammar of 2751 trees.

	Example 1	Example 1
Possible combinations	1 377	1 003 833
Combinations explored	9	9
Sentences (w/o selection)	6	36
Sentences (with selection)	2	12

Polarity filtering and Paraphrase selection results

Chart size is reduced by 77% and 87%

	Example 1		Example 2	
Optimisations	Chart sz	Time	Chart sz	Time
none	522	0.9 s	362	2.1 s
pol	125	0.2 s	46	0.7 s
pol + factor	77	0.3 s	30	1.1 s
pol + select	24	0.1 s	10	0.3 s
Carroll	n/a	1.8 s	n/a	4.3 s
Koller	n/a	1.4 s	n/a	0.8 s

Polarity filtering and Tabulation results

Decreases the chart size by 83%.

(3) The fact that the manager organizes a conference annoys the consultant.
Que le directeur organise un seminaire ennuie le consultant

Optimisations	Chart size	CPU time
pol	1258	0.61 s
pol + factor	219	0.47 s

Related approaches

Improving the efficiency of surface realisation:

- HPSG based approach (Carroll et al. 99; Carroll and Oepen 06)
- greedy strategy (White 04)
- Constraint based approach of (Koller and Striegnitz 2002)

Inference in NLG

- Much meaning in discourse is left implicit.
- Generating discourse involves deciding what can be effectively conveyed by implicit means.
- This requires models of the hearer's knowledge and reasoning based on this model.

Implicit information in discourse

Example from (McDonald, 1995)
Text:

> Fairchild Corp. *(Chantilly VA) - Donald E. Miller was named senior VP and general counsel, succeeding Dominic A. Petito, who resigned in November, at this aerospace business.*

Implicit information (= missing verb argument):

- Donald E. Miller succeeds Dominic A. Petito as senior VP and general counsel of Fairchild Corp.
- Dominic A. Petito resigned from his position of senior VP and general counsel at Fairchild Corp. in November.

Example 2

Haddock 1991

> *Remove the rabbit from the hat.*

Implicit information (=verb presupposition):

- the rabbit is in a hat
- the hat contains a rabbit

Example 3

Bierner and Webber 1991

> *John and the other ardvaaks*

Implicit information (=presupposition of "other"):

- John is an ardvaak

More generally

- Different kinds of information can be conveyed implicitly (properties, proposition, relations)
- Implicit information is conveyed by various means (lexical semantics, missing arguments, presuppositions)
- NLG systems must be able to decide when and how to effectively convey information by implicit means

Which linguistic means for providing implicit information?

- Missing arguments (of verbs, adjectives, nouns)
- Definite descriptions
  John's house is very nice. The door (of John's house) is bright blue.
- Certain adjectives
  John and the other aardvarks
- Modifiers - how, when
  Fold the square in half (along the diagonal/in two rectangles)
- etc.

When can information be conveyed implicitly?

Information can be conveyed implicitly when:

- this information is entailed by the context
- a linguistic means is available which supports the non verbalisation of this information

Therefore, S must be able to predict what the context will be like when utterance reaches H.

Testing for entailed information

- Information can be left implicit when it is entailed by the context
- The context in NLG includes the preceding text
- Text ordering hence, preceding text, is fixed by microplanning
- Hence entailments can only be checked at the surface realisation stage
- The NLG microplanners SPUD and INDIGEN integrate realisation and reasoning against the context to generate "natural sounding text" i.e., text leaving out information that is entailed by the context.

Spud: the example of implicit definite descriptions

We now see how SPUD allows the generation of :

> *Remove the rabbit from the hat.*

rather than:

> *Remove the rabbit that is in the hat from the hat that contain the rabbit.*

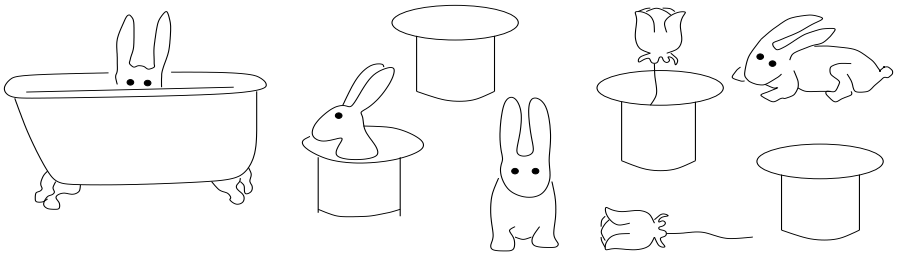
SPUD (Sentence Planning Using Descriptions)

[Stone & Doran 97], [Stone & Webber 98]
SPUD adopts an integrated approach to sentence planning, at the same time identifying ≥ 1 properties that contribute to identifying an entity and choosing a lexico-syntactic structure in which to realise them.
SPUD consists of:

- A generation algorithm
- A discourse model which records discourse entities, their cognitive status and their respective salience.
- A TAG–grammar specifying the syntactic, semantic and pragmatic contributions of & constraints on each lexical element

Conveying properties implicitly

(Stone & Webber, 1998)



(rabbit r_1)	(rabbit r_2)	(rabbit r_3)
(rabbit r_4)	(white r_1)	(white r_2)
(white r_3)	(white r_4)	(fluffy r_1)
(fluffy r_2)	(scruffy r_3)	(sleek r_4)
(hat h_1)	(hat h_2)	(hat h_3)
(hat h_4)	(tulip f_1)	(bathtub b_1)
(in f_1 h_3)	(tulip f_2)	(in r_1 h_1)
(in r_2 b_1)	(crouching r_3)	(alert r_4)

Given the Communicative Goals:

- communicate: causedEvent(e_0, $\langle$hearer$\rangle$)
- communicate: Result away(res(e_0),r_1,h_1)
- identify: e_0
- identify: $\langle$hearer$\rangle$
- identify: r_1
- identify: h_1

[Dale & Haddock] show that r_1 and h_1 can be specified as:

- r_1: "the rabbit in the hat"
- h_1: "the hat containing the rabbit"

These communicative goals could thus be satisfied by saying:

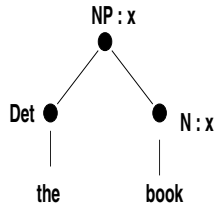
> *"Remove the rabbit in the hat from the hat containing the rabbit".*

Why is this silly?

Claim: Predicates impose implicit constraints on their arguments that can complement explicit constraints from NP descriptors.
$\Rightarrow$ "Remove the rabbit from the hat."
But not: "Smile at the rabbit."

A sample grammar entry

Lexical entry for *the book*.
SYNTAX:



SEMANTICS: *Presupposition: book(x)*
Assertion: —
PRAGMATICS: *uniquely–identifiable(x)*

uniquely–identifiable(x) requires that:

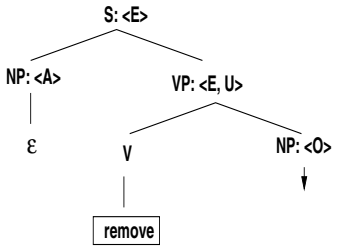
1. the entity instantiating x be uniquely identifiable in the current discourse model;
2. a description be generated which uniquely identifies x with respect to equally salient entities.

The generation algorithm

- Start with one–node tree (e.g. S, NP) and a set of communicative goals.
- While the current tree is incomplete, or a referring form is ambiguous, or any goals remain unsatisfied:
  - Identify applicable lexical options i.e. trees which can combine with the current syntactic tree and either contribute some required information or distinguish some entity from its distractors.
  - Rank the results of combining these trees with the current tree by: number of unfilled substitution sites, discriminatory power, goal satisfaction, etc.
  - Make the highest ranking new tree the current tree.

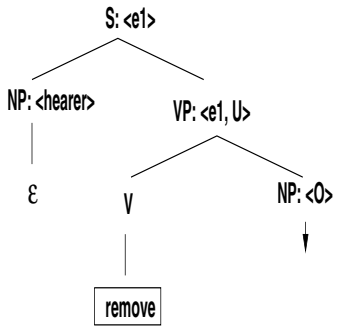
Rabbits & Hats: Cycle 1

Imperative /remove/ tree:
SYNTAX:

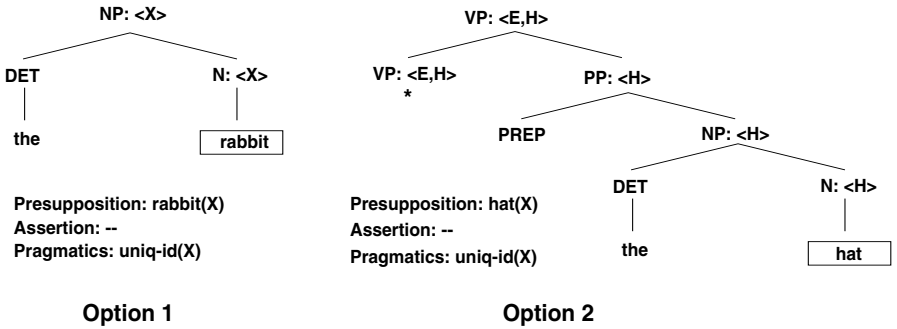


SEM: *Presupposition*: loc(start(E),O,U)
Assertion: CausedEvent(E,A) ∧
Result away(res(E),O,U)
PRAGMATICS: –

Check presuppositions: Are presuppositions of 'remove' entailed by the context? correspond to shared information?
Check assertions: Do assertions of 'remove' correctly describe e_0 from speaker's perspective?
Syntax: Is syntactic tree complete?
Communicative Goals: Are the *communicate* goals satisfied? Are the *distinguish* goals satisfied?
Updated tree:



Rabbits & Hats: Cycle 2



For each option:

Check presuppositions Are presuppositions entailed by the context?

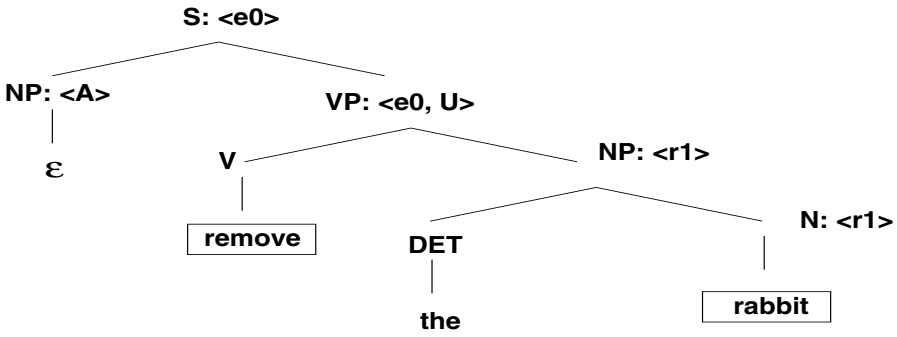
Check assertions Do assertions correctly describe e_0 from speaker's perspective?

Syntax Is syntactic tree complete?

Communicative Goals Are the *communicate* goals satisfied? Are the *distinguish* goals satisfied?

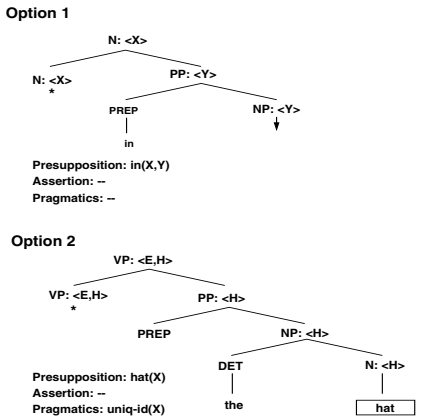
Updated tree

Option 1 beats Option 2 by completing the tree.



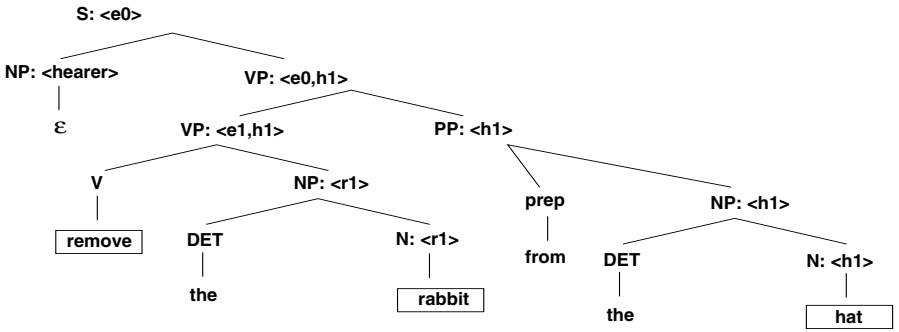
But the communicative goals are not yet all satisfied.

Rabbits & Hats: Cycle 3



1. Are presuppositions entailed by the context?
2. Do assertions correctly describe e_0 from speaker's perspective?
3. Is syntactic tree complete?
4. Are the *communicate* goals satisfied? Are the *distinguish* goals satisfied?

Option 2 Selected



Recap

- Natural language text conveys much information implicitly
- NLG systems must reason against the context to detect when information is already entailed by the common knowledge shared by the user
- Generating natural sounding text implies interleaving surface realisation with reasoning

Standard NLG Architecture

- Macroplanning
  - Content determination
  - Document structuration
- Microplanning
  - Aggregation
  - Referring expressions
  - Lexicalisation
- Realisation

Where to find more information about NLG?

Conferences

- *International Conference on Natural Language Generation* (INLG)
- *European Workshop on Natural Language Generation* (EWNLG)
- *Annual Meeting of the Association for Computational Linguistics* (ACL)
- *International Conference on Computational Linguistics* (COLING)
- *International Joint Conference on Artificial Intelligence* (IJCAI)
- *German Conference on Artificial Intelligence* (KI)

More on NLG

Journals

- *Computational Linguistics*
- *Artificial Intelligence*
- *Natural Language Engineering*

Websites

- ACL Special Interest Group on Generation (SIGGEN)
  `http://www.siggen.org/`
- Website with NLG systems:
  `http://www.fb10.uni-bremen.de/anglistik/langpro/NLG-table`

Acknowledgments

Many thanks to Armin Fiedler, Eric Kow, and Yannick Parmentier for allowing me to recycle some of their slides.