CS 533
Natural Language Processing
Lecture 8 – March 31, 2003

Matthew Stone

THE VILLAGE

Department of Computer Science
Center for Cognitive Science
Rutgers University

Natural Language Generation (NLG)

Outline

- The practical NLG pipeline
- Descriptions in NLG
- Generating referring expressions (GRE)
- GRE and the grammar
- Broader context of NLG

In practice, NLG systems work the way we can build them.

They solve a specific, carefully-delineated task.

- They can verbalize only specific knowledge.
- They can verbalize it only in specific, often quite stereotyped ways.

In practice, NLG systems work the way we can build them.

That means start with available input and the desired output, and putting together something that maps from one to the other.

- Any linguistics is a bonus.
- Any formal analysis of computation is a bonus.

Input can come from …

- Existing database (e.g., tables)
  Format facilitates update, etc.
- An interface that allows a user to specify it (e.g., by selecting from menus)
- Language interpretation

For Example

Input: Rail schedule database.

Current train status.

User query

When is the next train to Glasgow?

Output:

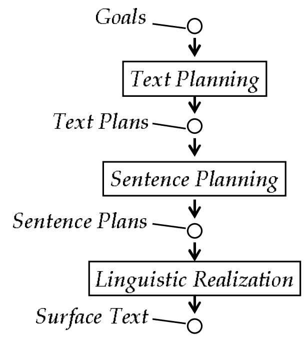
There are 20 trains each day from Aberdeen to Glasgow. The next train is the Caledonian express; it leaves Aberdeen at 10am. It is due to arrive in Glasgow at 1pm, but arrival may be slightly delayed.

To get from input to output means selecting and organizing information

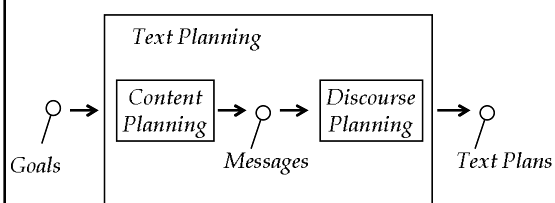
The selection and organization typically happens in a *cascade* of processes that use special ***data structures*** or ***representations***

- Each makes explicit a degree of selection and organization that the system is committed to.
- Indirectly, each indicates the degree of selection and organization the system has still to create.

The NLG Pipeline



Overview of Processes and Representations, 1



Message

A ***message*** represents a piece of information that the text should convey, in domain terms.

Example Messages

message-id: msg01
relation: IDENTITY
arguments: arg1: NEXT-TRAIN
arg2: CALEDONIAN-EXPRESS

The next train is the Caledonian Express

Example Messages

message-id: msg02
relation: DEPARTURE
arguments: entity: CALEDONIAN-EXPRESS
location: ABERDEEN
time: 1000

The Caledonian Express leaves Aberdeen at 10am.

A close variant

- *Q: When is the next train to New Brunswick?*
- *A: It's the 7:38 Trenton express.*

I know something about the domain in this case – and can highlight how nonlinguistic the domain representation will be.

Variant message

message-id: msg03
relation: NEXT-SERVICE
arguments:
 station-stop: STATION-144
 train: TRAIN-3821

The next train to New Brunwick is the Trenton Local.

Closer to home

message-id: msg04
relation: DEPARTURE
arguments:
 origin: STATION-000
 train: TRAIN-3821
 time: 0738

It leaves Penn Station at 7:38.

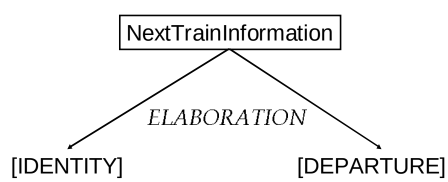
How I got domain knowledge

NY Penn Station really is NJT Station 000,
New Brunswick really is Station 144
(you have to key this into ticket machines!)
This really is train #3821
(it's listed with this number on the schedule!)

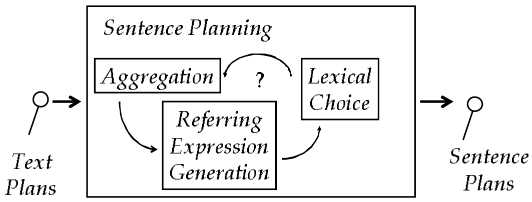
Text Plan

A ***text plan*** represents the ***argument*** that the text should convey; it is a hierarchical structure of interrelated messages.

Example Text Plan



Overview of Processes and Representations, 2



Sentence Plans

A ***sentence plan*** makes explicit the lexical elements and relations that have to be realized in a sentence of the output text.

Example Sentence Plan

```
(S1/be
  :subject (NEXT-SERVICE/it)
  :object (TRAIN-3821/express
           :modifier Trenton
           :modifier 7:38
           :status definite))
```

It's the 7:38 Trenton express.

We know what's happened

Aggregation: we have constructed a single sentence that realizes *two* messages.

Once we have the first message:

It's the Trenton express.

We just add *7:38* to realize the second message:

It's the 7:38 Trenton express.

We know what's happened

Referring expression generation: we have figured out to realize the *next-service* as *it,* and figured out to identify the train by its *destination* and *frequency of stops.*

We know what's happened

Lexical (and grammatical) choice:

to use the verb *be* with *it* as the subject and a reference to the train second;

to say *express* rather than *express train.*

to say *Trenton* rather than *Northeast Corridor.*

But there’s no consensus method for how to do it.

- Reiter (1994, survey of 5 NLG systems): Most practical systems follow a pipeline, even though this makes some things difficult to do. *Example:* Avoidance of ambiguity
- Cahill et al. (1999, survey of 18 NLG systems): Tasks like *Aggregation* and *GRE* can happen almost anywhere in the system, e.g.,
  - as early as Content Planning
  - as late as Sentence Realization

But there’s no consensus method for how to do it.

And we’ll see that *formal* and *computational* questions raise important difficulties for

what representations you can have

what processes and algorithms you can use

how you bring knowledge of language into the loop

Overview of Processes and Representations, 3

Sentence Plans → *Linguistic Realization* → *Surface Text*

This is easier to think about

We all know what a surface text looks like!

And we all know you ***have*** to have a grammar (of some kind or other) to get one!

Our Question This Week

What are the possible ways of using *knowledge (of the world and of language)* in formulating an utterance?

Knowledge in utterances

Knowledge of the *world*

Utterance says something ***useful*** and ***reliable***.

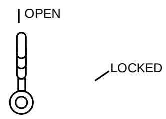
Knowledge of *language*

Utterance is ***natural*** and ***concise***,

in other words, it fits ***hearer*** and ***context***.

A Concrete Example

Our partner is working with equipment that looks like:



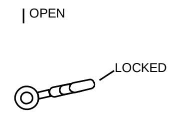
The instruction that we'd like to give them is:

Turn handle to locked position.

Knowledge in this utterance

Knowledge of the *world*

Utterance says something ***useful*** and ***reliable***.



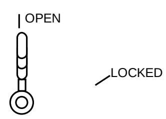
*This **is** what has to happen next.*

Knowledge in this utterance

Knowledge of *language*

Utterance is ***natural*** and ***concise***.

Consider the alternatives…



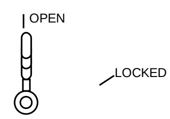
Move the thing around.

Knowledge in this utterance

Knowledge of *language*

Utterance is ***natural*** and ***concise***.

Consider the alternatives…



You ought to readjust the fuel-line access panel handle by pulling clockwise 48 degrees until the latch catches.

Our Question This Week

What are the possible ways of using *knowledge (of the world and of language)* in formulating an utterance?

This is a *formal* question; the answers will depend on the *logics* behind grammatical information and real-world inference.

The NLG problem depends on the input to the system

If the input looked like this:

INPUT: *turn′(handle′, locked′)*

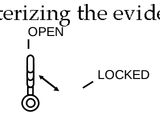
Deriving the output would be easy:

OUTPUT: *Turn handle to locked position.*

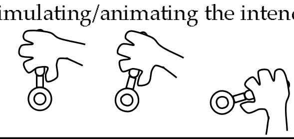
Real conceptual input is richer and organized differently

Must support correct, useful domain reasoning

- e.g., characterizing the evident function of equipment

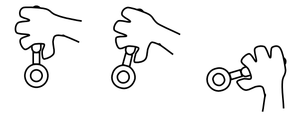


- e.g., simulating/animating the intended action



Difference in Content

Input: New info complete, separate from old



Output: New info cut down, mixed with old

Turn handle to locked position.

Difference in Organization

Input: *deictic* representation for objects

- through atomic symbols that index a flat database
  - handle(**object388**). number(**object388**, "16A46164-1").
  - composedOf(**object388**,steel). color(**object388**,black).
  - goal(**object388**,activity116).
  - partOf(**object388**,object486).

Output: *descriptions* for objects

- through complex, hierarchical structures
  - NP – DET – **the**
  - - N′ – N – **handle**

Why we have to *invent* ways of describing things

1. The referent has a familiar name, but it's not unique, e.g., *'John Smith'*
2. The referent has no familiar name: trains, furniture, trees, atomic particles, …

(In such cases, databases use **database keys,** e.g., *'Smith73527', 'TRAIN-3821'*)

3. Similar: *sets* of objects

Formal problem

NLG means applying input domain knowledge that looks quite different from output language!

Formal problem

How can we characterize these different sources of information

in a common framework

as part of a coherent model of language use

For example: how can we represent linguistic distinctions that make choices in NLG good or bad?

Our Question This Week

What are the possible ways of using ***knowledge (of the world and of language)*** in formulating an utterance?

This is not just a ***mathematical*** question –

This is a ***computational*** question; possible ways of using knowledge will be ***algorithms***.

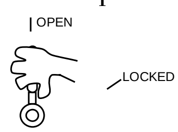
No Simple Strategy to Resolve Differences

Lots of variability in natural instructions

- Lift assembly at hinge.
- Disconnect cable from receptacle.
- Rotate assembly downward.
- Slide sleeve onto tube.
- Push in on poppet.

Strategy Must Decide When to Say More

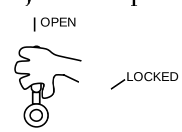
Using utterance interpretation as a whole



Turn handle by hand-grip
from current open position for handle
48 degrees clockwise
to locked position for handle.

In particular, hearer matches shared initial state

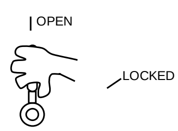
So describe objects and places succinctly



Turn handle by hand-grip
from current open position for handle
48 degrees clockwise
to locked position for handle.

In particular, hearer applies knowledge of the domain

So omit inevitable features of action



Turn handle by hand-grip
from current open position for handle
48 degrees clockwise
to locked position for handle.

Computational problem

Because this process is so complex, it takes special effort to specify the process, to make it effective, and ensure that it works appropriately.

For example: How much search is necessary? How can we control search when search is required? What will such a system be able to say?

Descriptions in NLG Overview

An NLG system is given as ***input*** some ***domain representations*** that need to be presented to the user.

The system has to formulate an ***output*** utterance that will get this information across ***linguistically***.

Descriptions in NLG: Overview

A ***description*** is a linguistic expression whose interpretation accesses a domain representation drawn from context.

The flexibility of description.

The ***semantics*** of a description is a linguistic representation.

But its ***interpretation*** involves a ***resolution*** that can link meaning up with domain representations ***arbitrarily.***

Example

Description: ***the mug***.

Semantics: $mug(x)$

Presents an object $x = m211$ to the user

Interpretation: $mug(m211)$

Because you use ***inference*** and ***substitution*** to compute the value $m211$ for x, the only formal constraint required in the model is that the value of x is a term in your domain representation language.

Example

Description:

slide the sleeve onto the elbow.

Semantics:

$slide(a,s,p) \wedge sleeve(s) \wedge onto(p,e) \wedge elbow(e)$

Presents an ***action*** a to the user.

This is where flexibility becomes important

Description:

slide the sleeve onto the elbow.

Semantics:

$slide(a,s,p) \wedge sleeve(s) \wedge onto(p,e) \wedge elbow(e)$

Interpretation can access whatever ***independent representation of a*** the system has:

$a = step5$

$a = displace(s13,vector(-1,0,0))$

$a = slide(s13, path(at(j13),on(e13)))$

Description in NLG: Overview

The key task of the generator is now to ***construct*** a semantics with the ***right interpretation***.

Example

Input: Present an object $x = m211$ to the user

Output description: ***the mug***.

Constructed semantics: $mug(x)$

Derived interpretation: $mug(m211)$

Declarative programming

Allows us to use a grammar in NLG to construct syntax and semantics for an output sentence simultaneously.

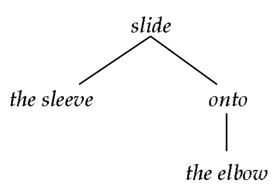
Good design:

- generation happens in one place
- easy extension of routines that build semantics
- impossible for semantics to crash realization

Example

Syntactic derivation structure for

slide the sleeve onto the elbow



Example

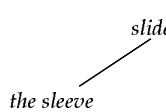
Step-by-step guide for building meaning

slide

$slide(a,s,p)$

Example

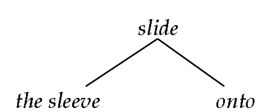
Step-by-step guide for building meaning



$slide(a,s,p) \wedge sleeve(s)$

Example

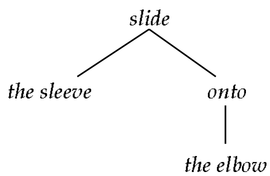
Step-by-step guide for building meaning



$slide(a,s,p) \wedge sleeve(s) \wedge onto(p,e)$

Example

Step-by-step guide for building meaning



$slide(a,s,p) \wedge sleeve(s) \wedge onto(p,e) \wedge elbow(e)$

GRE as an NLG task

Find the best NP description to present some domain object to the user

GRE is microcosm of NLG: e.g., determines

- which properties to express (*Content Determination*)
- which syntactic configuration to use (*Syntactic Realization*)
- which words to choose (*Lexical Choice*)

What is the best description?

One that fulfills the Gricean maxims.

(Quality:) list properties truthfully

(Quantity:) list sufficient properties to allow hearer to identify referent – but not more

(Relevance:) use properties that are of interest in themselves

(Manner:) be brief

(Dale & Reiter 1995)

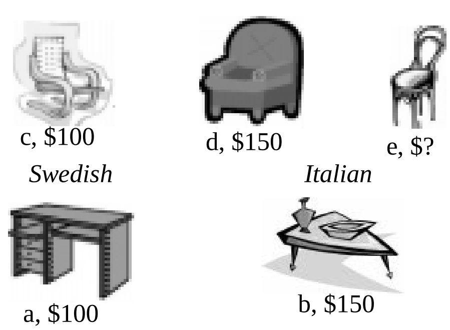
Why obey the maxims?

Violation of a maxim leads to *implicatures*.

For example,

- **[Quantity]** *'the pitbull'* (when there is only one dog).
- **[Manner]** *'Get the cordless drill that's in the toolbox'* (Appelt).

Example Situation



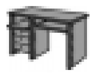
Formalized in a KB

- Type: furniture (**abcde**), desk (**ab**), chair (**cde**)
- Origin: Sweden (**ac**), Italy (**bde**)
- colors: dark (**ade**), light (**bc**), grey (**a**)
- Price: 100 (**ac**), 150 (**bd**) , 250 ({})
- Contains: wood ({}), metal ({**abcde**}), cotton(**d**)

Assumption: all this is shared knowledge.

Violations of …

- **Manner**:
  *‘The $100 grey Swedish desk which is made of metal’
  (Description of **a**)



- **Relevance**:
  *‘The **cotton** chair is a fire hazard?*

  *?Then why not buy the **Swedish** chair?*
  (Descriptions of **d** and **c** respectively)

Problem: characterizing the maxims correctly

Missing implicatures:

- [Manner] ‘the red chair’ (when there is only one red object in the domain).
- [Manner/Quantity] ‘I broke my arm’ (when I have two).

(empirical work shows much redundancy)

- [Quality] ‘the man with the martini’ (Donellan)

etc.

A computational apprach

Make choices heuristically

by a procedure that generally yields descriptions that are interpreted as intended

Incremental approach

Properties are considered in a fixed order:

P = $P_1, P_2, P_3, \ldots, P_n$

called preference order.

A property is included if it is ‘useful’:

true of target; false of some distractors

Stop when done;

so preferred properties have a greater chance of being included.

Formal setup

r = individual to be described
P = list of properties, in preference order
P is a property
L= properties in generated description

L := Φ
C := Domain
For all $P \in$ P do :
  If $r \in [[P]]$ & $C \not\subset [[P]]$ then do
    L := $L \cup \{P\}$
    C := $C \cap [[P]]$
    If $C = \{r\}$ then Return L
Return Failure

P = < furniture (**abcde**), desk (**ab**), chair (**cde**),
Swedish (**ac**), Italian (**bde**),
dark (**ade**), light (**bc**), grey (**a**),
100$ ({**ac**}), 150$(**bd**) , 250$ ({}),
wooden ({}), metal (**abcde**), cotton ({**d**}) >

Domain = {**a,b,c,d,e**} . Now describe*:*


a = <...>


d = <...>


e = <...>

P = < furniture (**abcde**), desk (**ab**), chair (**cde**),
Swedish (**ac**), Italian (**bde**),
dark (**ade**), light (**bc**), grey (**a**),
100$ (**ac**),200$ (**bd**),250$ ({}),
wooden ({}), metal (**abcde**), cotton (**d**) >

Domain = {**a,b,c,d,e**} . Now describe*:*


a = <desk {**ab**}, Swedish {**ac**}>


d = <chair,Italian,dark,200> *(Nonminimal)*


e = <chair,Italian,dark, ...> *(Impossible)*

Incremental Algorithm

It's a *hillclimbing* algorithm: ever better approximations of a successful description.

'Incremental' means *no backtracking.*

Not always the *minimal* number of properties.

Incremental Algorithm

Logical completeness: A unique description is found in finite time *if there exists one*. (Given reasonable assumptions, see van Deemter 2002)

Computational complexity: Assume that testing for usefulness takes *constant time*. Then worst-case time complexity is $O(n_p)$ where n_p is the number of properties in **P**.

Using more domain knowledge (D&R 1995)

Attribute + Value model:

Properties grouped together:
origin: Sweden, Italy, ...
color: dark, grey, ...

Pick attributes in order.

Optimize within properties based on that attribute.

Incremental Algorithm, using Attributes and Values

- **r** = individual to be described
- **A** = list of Attributes, in preference order
- *Def*: $V_{i,j}$ = Value **i** of Attribute **j**
- **L**= properties in generated description

$L := \Phi$

$C :=$ Domain

For all $A_i \in A$ do :

$V_{i,j} = \boxed{\text{Find BestValue}(r, A_i)}$

If $r \in [[V_{i,j}]]$ & $C \not\subset [[V_{i,j}]]$ then do

$L := L \cup \{V_{i,j}\}$

$C := C \cap [[V_{i,j}]]$

If $C = \{r\}$ then Return L

Return Failure

- FindBestValue(**r**,**A**):

 - Find Values of **A** that are true of **r**, while removing some distractors (If these don't exist, go to next Attribute)

 - Within this set, select the Value that *removes the largest number of distractors*

 - If there's a tie, select *the most general one*

 - If there's still a tie, select an arbitrary one

Example: D = {a,b,c,d,f,g}

- Type: furniture (**abcd**), desk (**ab**), chair (**cd**)
- Origin: Europe (**bdfg**), USA (**ac**), Italy (**bd**)

Describe **a**: {desk, American}

(*furniture* removes fewer distractors than *desk*)

Describe **b**: {desk, European}

(*European* is more general than *Italian*)

N.B. This disregards relevance, etc.

Complexity of the algorithm

n_d = nr. of distractors

n_l = nr. of properties *in the description*

n_v = nr. of Values (for all Attributes)

Alternative assessment: $O(n_v)$

(*Worst-case* running time)

According to D&R: $O(n_d n_l)$

(*Typical* running time)

Minor complication: Head nouns

Another way in which human descriptions are nonminimal

- A description needs a *Noun*, but not all properties are expressed as *Nouns*

- Example: Suppose color was the most-preferred Attribute, and **target** = **a**

- colors: dark (**ade**), light (**bc**), grey (**a**)
- Type: furniture (**abcde**), desk (**ab**), chair (**cde**)
- Origin: Sweden (**ac**), Italy (**bde**)
- Price: 100 (**ac**), 150 (**bd**) , 250 ({})
- Contains: wood ({}), metal ({**abcde**}), cotton(**d**)

target = a

Describe **a**: {grey}

'*The grey*' ? (Not in English)

D&R's repair:

- Assume that Values of the Attribute <u>Type</u> can be expressed in a *Noun*.
- After the core algorithm:
  - check whether <u>Type</u> is represented.
  - if not, then add the best Value of the <u>Type</u> Attribute to the description

GRE and surface realization

Arguably, GRE ***uses a grammar***.

– Parameters such as the ***preference order*** on properties reflect knowledge of how to communicate effectively.

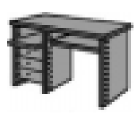
– Decisions about ***usefulness*** or ***completeness*** of a referring expression reflect beliefs about ***utterance interpretation***.

Maybe this is a good idea for NLG generally.

GRE and surface realization

But we've thought GRE outputs semantics:

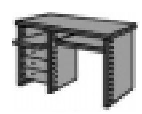
$$\begin{pmatrix} \text{referent: furniture886} \\ \text{type: desk} \\ \text{status: definite} \\ \text{color: brown} \\ \text{origin: sweden} \end{pmatrix}$$



GRE and surface realization

We also need to link this up with surface form:

the brown Swedish desk



Note: ***not***

?the Swedish brown desk

Observations

It's ***hard*** to do realization on its own

mapping from semantics to surface structure.

It's ***easy*** to combine GRE and realization

because GRE ***is*** grammatical reasoning!

if you have a good representation for syntax.

Why it's hard to do realization

A pathological grammar of adjective order:

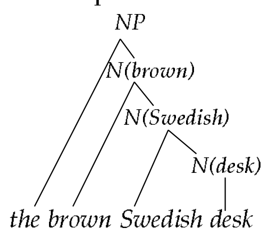
NP → the N(w).

N(w) → w N(w') if w is an adjective and $w\mathbf{R}w'$.

N(w) → w if w is a noun.

Syntax with this grammar

Derivation of example:



*Requires: brown **R** Swedish, Swedish **R** desk*

Realization, formally

You start with *k* properties.

Each property can be realized lexically.

- assume: one noun, many adjectives
- (not that it's easy to enforce this)

Realization solution:

- NP which realizes ***each property exactly once.***

Quick formal analysis

View problem graph-theoretically:

- *k* words, corresponding to vertices in a graph
- ***R*** is a graph on the *k* words
- Surface structure is a ***Hamiltonian path*** (which visits each vertex exactly once) through ***R***.

This is a famous NP complete problem

- So surface realization itself is intractable!

Moral of the example

Semantics underdetermines syntactic relations.

- Here, semantics underdetermines syntactic relations of adjectives to one another and to the head.

Searching for the correspondence is hard.

- See also Brew 92, Koller and Striegnitz 02.

Observations

It's ***hard*** to do realization on its own

- mapping from semantics to surface structure.

It's ***easy*** to combine GRE and realization

- because GRE ***is*** grammatical reasoning!
- if you have a good representation for syntax.

Syntactic processing for GRE

Lexicalization

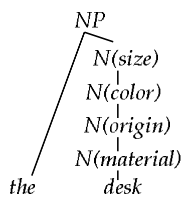
Steps of grammatical derivation correspond to meaningful choices in NLG.

E.g., steps of grammar are synched with steps of adding a property to a description.

Syntactic processing for GRE

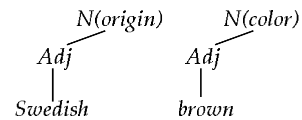
Key ideas: ***lexicalization,*** plus

- ***Flat*** dependency structure (adjs modify noun)
- ***Hierarchical*** representation of word-order



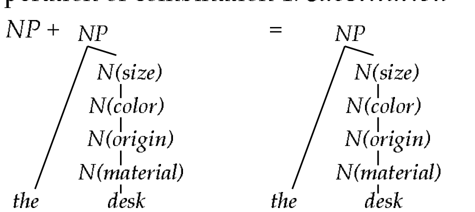
Syntactic processing for GRE

Other syntactic lexical entries



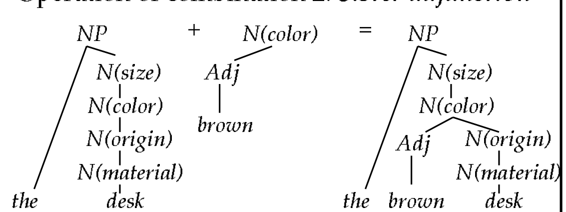
Describing syntactic combination

Operation of combination 1: ***Substitution***



Describing syntactic combination

Operation of combination 2: ***Sister adjunction***



Abstracting syntax

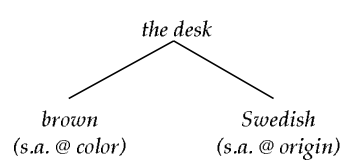
Tree rewriting:

- Each lexical item is associated with a structure.
- You have a starting structure.
- You have ways of combining two structures together.

Abstracting syntax

Derivation tree

records elements and how they are combined



An extended incremental algorithm

- r = individual to be described
- P = lexicon of entries, in preference order
  - *P* is an individual entry
  - *sem(P)* is a property or set of entries from the context
  - *syn(P)* is a syntactic element
- L = surface syntax of description

Extended incremental algorithm

L := NP$\downarrow$
C := Domain
For each $P \in$ P do:
  If $r \in sem(P)$ & $C \not\subset sem(P)$
  Then do
    $\boldsymbol{L := \textbf{add}(syn(P), L)}$
    $C := C \cap sem(P)$
    If $C = \{r\}$ then return L
Return failure

Observations

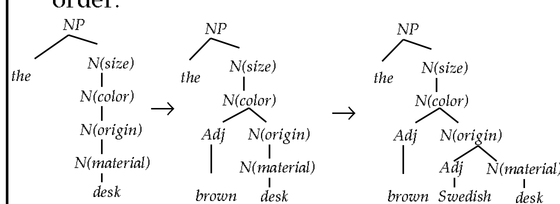
Why use tree-rewriting - not,e.g. CFG derivation?

NP $\rightarrow$ the N(w).
N(w) $\rightarrow w$ N(w') if w is an adjective and $w\boldsymbol{R}w'$.
N(w) $\rightarrow w$ if w is a noun.

CFG derivation forces you to select properties in the surface word-order.

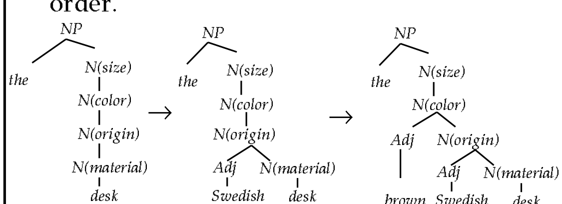
Observations

Tree-rewriting frees word-order from choice-order.



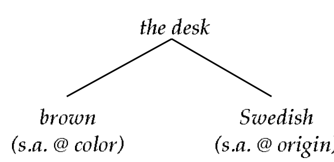
Observations

Tree-rewriting frees word-order from choice-order.



This is reflected in derivation tree

Derivation tree
records elements and how they are combined



Formal results

Logical completeness.

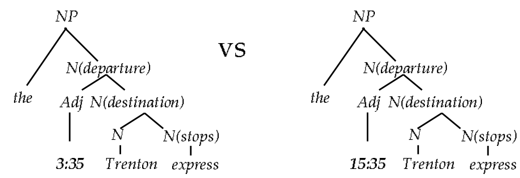
- ***If*** there's a ***flat*** derivation tree for an NP that identifies referent ***r,***
- ***Then*** the ***incremental algorithm*** finds it.

But

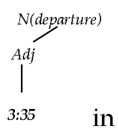
- Sensible combinations of ***properties*** may ***not*** yield surface *NPs*.
- ***Hierarchical*** derivation trees may require ***lookahead*** in usefulness check.

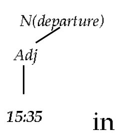
Now, though, we're choosing specific lexical entries

maybe these lexical items express the same property…



What motivates these choices?

- Use 

in 12-hour time context
- Use 

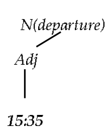
in 24-hour time context

Need to extend grammar again

- P = lexicon of entries, in preference order
  - *P* is an individual entry
  - *sem(P)* is a property or set of entries from the context
  - *syn(P)* is a syntactic element
  - *prags(P)* is a test which the context must satisfy for the entry to be appropriate

Need to extend grammar again

For example:

syn: 

sem: departure(x, 1535)

prags: twentyfourhourtime

Extended incremental algorithm

L := NP↓
C := Domain
For each $P \in$ P do:
  If $r \in sem(P)$ & $C \not\subset sem(P)$ **& *prags*(*P*) is true**
  Then do
    L := add($syn(P)$, L)
    C := $C \cap sem(P)$
    If $C = \{r\}$ then return L
Return failure

Discussion: What does this entry do?

syn: NP — ***it***

sem: thing(x)

prags: in-focus(x)

Suggestion: find best value

Given:

- A set of entries that combine syntactically with *L* in the same way
- Related by semantic generality and pragmatic specificity.
- Current distractors

Take entries that remove the most distractors

Of those, take the most semantically general

Of those, take the most pragmatically specific

Extended incremental algorithm

L := NP↓ C := Domain

Repeat

    Choices := { *P* : add(*syn*(*P*), *L*) at next node & $r \in$ *sem*(*P*) & *prags*(*P*) is true }

    P := find best value(*Choices*)

    L := add(*syn*(*P*), *L*)

    $C := C \cap sem(P)$

    If $C = \{r\}$ then return *L*

Return failure

What is generation anyway?

Generation is ***intentional (or rational) action***

that's why Grice's maxims apply, for example.

You have a ***goal***

You build a ***plan to achieve it***

(& achieve it economically in a recognizable way)

You carry out the plan

In GRE...

The ***goal*** is for hearer to know the identity of ***r***

(in general g)

The ***plan*** will be to utter some NP ***U***

such that the ***interpretation of U identifies { r }***

(in general $c \cap u \subseteq c \cap g$)

Carrying out the plan means realizing this utterance.

In other words

GRE amounts to ***a process of deliberation.***

Adding a property to L incrementally is like ***committing to an action.***

These commitments are called ***intentions.***

Incrementality is characteristic of intentions – though in general intentions are open to ***revision.***

Note: this connects with ***belief-desire-intention models of bounded rationality.***

GRE as (BDI) rational agency

L := NP ↓ // Initial plan
C := Domain // Interpretation
while (P := FindBest(***P***, *C*, *L*)) { // Deliberation
  L := add(*syn*(*P*), *L*) // Adopt new intention
  C := C ∩ *sem*(*P*) // Update interpretation
  if *C* = { ***r*** } return *L* // Goal satisfied
}
fail

NLG as (BDI) rational agency

L := X ↓
C := Initial Interpretation
while (P := FindBest(***P***, *C*, *L*)) {
  L := AddSyntax(*syn*(*P*), *L*)
  C := AddInterpretation(*sem*(*P*), *C*)
  if GoalSatisfied(*C*) return *L*
}
fail

Example

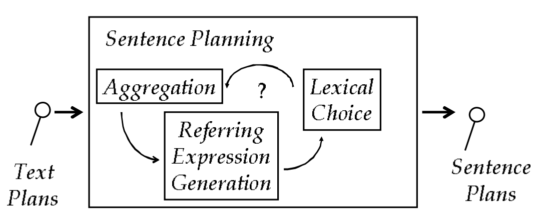
Description:
slide the sleeve onto the elbow.
Semantics:
$slide(a,s,p) \wedge sleeve(s) \wedge onto(p,e) \wedge elbow(e)$
Contribution:
$do(a)$
Pragmatics:
$imp(a) \wedge the(s) \wedge the(e)$

Example

Interpret
$slide(a,s,p) \wedge sleeve(s) \wedge onto(p,e) \wedge elbow(e)$
by proving it in the context
finding possible values for *a*, *s*, *p*, and *e*

Interpretation is successful if there's only one value each for *a*, *s*, *p*, and *e*

Overview of Processes and Representations, 2



Solving generation tasks with declarative descriptive NLG

Lexical choice:
- Key challenge: good lexical choice achieves multiple goals (Elhadad et al 1997)

Example
(Elhadad et al 1997)

Desired output:
- AI <u>requires</u> six assignments

Multiple goals:
- Class *c* (AI) involves stuff *a* (assignments)
- The stuff *a* represents a significant demand

Match contributions of *require*

Lexical choice

Accurate lexical choice depends on
- declarative conceptual and linguistic specifications for lexical items
- assessment of contribution of items to interpretation

=> declarative descriptive NLG.

Solving generation tasks with declarative descriptive NLG

"Aggregation"
- Organize complex sentences that present multiple domain representations to user.
- E.g., Dalianis 1996

Example

Two things to present:

do(*step5*), *purpose*(*step5, goal6*)

Extend

slide the sleeve onto the elbow

To

slide the sleeve onto the elbow
to uncover the fuel-line sealing-ring.

Corresponds to use of lexicogrammatical resource

Syntax: *VP: a*

Sinf:b

Contribution:

purpose(*a*,*b*)

Aggregation

Aggregation naturally builds from
- declarative conceptual and linguistic specifications for lexical items
- assessment of contribution of items to interpretation

=> declarative descriptive NLG.