

Is English really a formal language?

Robin Cooper

Department of Linguistics

Göteborg University

Contents

1	Natural languages and formal languages	3
2	The Grammatical Framework	8
3	Innovation and the Generative Lexicon	13
4	Records and record types	23
5	Dynamic generalized quantifiers	28
6	Treating lexical innovation	36
7	Conclusion: Towards a theory of linguistic innovation	46

1. Natural languages and formal languages

Natural and formal languages in 20th century linguistics

- languages as sets of strings and early transformational grammar
- interpreted languages as sets of string-meaning pairs
- Montague in ‘Universal Grammar’:

There is in my opinion no important theoretical difference between natural languages and the artificial languages of logicians; indeed I consider it possible to comprehend the syntax and semantics of both kinds of languages within a single natural and mathematically precise theory.

Natural languages as formal languages – the advantages

- productive theoretical abstraction allowing application of logical techniques to natural language
- a basis for much computational processing of language

Natural languages as formal language – the problems

grammaticality

- degrees of grammaticality
- context-dependent grammaticality
- speakers adapt the language to new situations and domains, changing grammaticality judgements

meaning

- words and phrases do not have a fixed range of interpretations
- speakers adapt meaning to the subject matter
- speakers negotiate meaning in dialogue
 - same proper name for different individuals
 - abstract or theoretical concepts like democracy or meaning

Natural languages as collections of resources

- a collection of resources (a “toolbox”) which can be used to construct (formal) languages
- address the problems
- maintain the insights and precision gained from the formal language view
- speakers of natural languages are constantly in the process of creating new language to meet the needs of novel situations in which they find themselves
- A corpus of natural language data (even a single dialogue) is not required to be consistent either in terms of grammaticality or in terms of meaning since it represents output based on a collection of related grammars rather than a single grammar.

2. The Grammatical Framework

Abstract syntax

example of MP3 player (from TALK project):

- Categories

```
cat Action ; Playlist ; Object ; Song ; Artist ;
```

- Functions

```
fun
```

```
  add_object_to_playlist : Playlist -> Object -> Action ;  
  create_playlist : Action ;  
  play : Object -> Action ;
```

Concrete syntax

characterizes how the semantic objects are *linearized*.

```
lin
```

```
add_object_to_playlist x y = "add" ++ x ++ "to" ++ y ;  
create_playlist = "create" ++ "a" ++ "playlist" ;  
play x = "play" ++ x ;
```

or for French:

```
lin
```

```
add_object_to_playlist x y = "ajouter" ++ x ++ "à" ++ y ;  
create_playlist = "créer" ++ "une" ++ "liste" ++  
                  "de" ++ "lecture" ;  
play x = "jouer" ++ x ;
```

Resource grammar libraries

Functions in the resource library are based on general syntactic structures

```
mkImp : V3 -> NP -> NP -> Imp
```

These functions can be used in concrete syntax:

```
add_object_to_playlist x y = mkImp add_V3 x y
```

The constant `add_V3` defined in each language, e.g. for German:

```
add_V3 = mkV3 (prefixV "hinzu" (regV "fügen"))  
          accPrep zu_Prep
```

Practical advantages of resource grammars

- division of labour between linguists and domain experts
- linguistic resources are domain independent
- resource grammars as linguistic ontologies
- formalizes the unifying structure of English or German which is then put into use in different “language games”

3. Innovation and the Generative Lexicon

Dot types and coercion in the Generative Lexicon

- dot types are compositions of two types which nevertheless allow the two individual types to be recovered
- used by Pustejovsky in *The Generative Lexicon*
- type theoretical approach proposed by Asher and Pustejovsky (2005)
- relation to type coercion

Lunch as food or an event

- (1) a. The lunch was delicious
- b. The lunch took forever

delicious – normally a predicate of food but
not events

- (2) a. The blancmange was delicious
b. ⁱIFK Göteborg's last game was delicious

take forever – a predicate of events, not of food

- (3) a. ⁱThe blancmange took forever
b. IFK's last game took forever

Is *lunch* polysemous?

lunch₁ : *FoodPred*

lunch₂ : *EventPred*

Copredication

- (4) a. The lunch was delicious but took forever
b. 'The bank specializes in IPO's and is being quickly eroded by the river

The dot-type analysis

- (5) a. A lunch was delicious
b. $\exists x : Food \exists v : Food \cdot Event$ [lunch(v) \wedge
O-Elab(x, v) \wedge delicious(x)]

Alternative:

- A lunch is both food *and* an event in virtue of having parts which are (purely) food and (purely) an event.
- Complex objects inherit properties from their parts (or at least we talk as if they do).

Consequence: We can use *meet* (conjunction) instead of *dot* and *O-elab*.

Predications of objects in virtue of predications of parts

- (6) a. The house is locked
 - b. The door/lock is locked

- (7) a. The choir/singers sang the Hallelujah Chorus. Mildred had a tickle in her throat and didn't sing a note.
 - b. The (players of the) Gothenburg Symphony Orchestra played Mahler's second and a Mozart symphony. There were many more people playing for the Mahler.

Genericity and inexact predicates

Boston drivers are bad (Greg Carlson)

- (8) The GSO played Elgar's Intro and Allegro for Strings
- (9) The GSO played a solo violin partita by Bach

4. Records and record types

Records and record types

If $a_1 : T_1, a_2 : T_2(a_1), \dots, a_n : T_n(a_1, a_2, \dots, a_{n-1})$,

the record:

$$\left[\begin{array}{l} l_1 = a_1 \\ l_2 = a_2 \\ \dots \\ l_n = a_n \\ \dots \end{array} \right]$$

is of type:

$$\left[\begin{array}{l} l_1 : T_1 \\ l_2 : T_2(l_1) \\ \dots \\ l_n : T_n(l_1, l_2, \dots, l_{n-1}) \end{array} \right]$$

a man owns a donkey

Record type:

$$\left[\begin{array}{l} x : Ind \\ c_1 : \text{man}(x) \\ y : Ind \\ c_2 : \text{donkey}(y) \\ c_3 : \text{own}(x,y) \end{array} \right]$$

Record:

$$\left[\begin{array}{l} x = a \\ c_1 = p_1 \\ y = b \\ c_2 = p_2 \\ c_3 = p_3 \end{array} \right]$$

where

a, b are of type *Ind*, individuals

p_1 is a proof of $\text{man}(a)$

p_2 is a proof of $\text{donkey}(b)$

p_3 is a proof of $\text{own}(a, b)$

[⇐ contents](#)

Records are recursive

$$r = \left[\begin{array}{l} f = \left[\begin{array}{l} f = \left[\begin{array}{l} ff = a \\ gg = b \end{array} \right] \\ g = c \end{array} \right] \\ g = \left[\begin{array}{l} h = \left[\begin{array}{l} g = a \\ h = d \end{array} \right] \end{array} \right] \end{array} \right]$$

is of type

$$R = \left[\begin{array}{l} f : \left[\begin{array}{l} f : \left[\begin{array}{l} ff : T_1 \\ gg : T_2 \end{array} \right] \\ g : T_3 \end{array} \right] \\ g : \left[\begin{array}{l} h : \left[\begin{array}{l} g : T_1 \\ h : T_4 \end{array} \right] \end{array} \right] \end{array} \right]$$

given that $a : T_1, b : T_2, c : T_3$ and $d : T_4$.

⇐ contents

Functions (λ -calculus)

donkey

$\lambda r: [x:Ind] ([c:donkey(r.x)])$

:

$([x:Ind])RecType$

cf. Montague's $\langle e, t \rangle$

5. Dynamic generalized quantifiers

Generalized quantifiers

the donkey runs

(10) [c : $\text{the}(\lambda r:[x:Ind]([c_1:\text{donkey}(r.x)]))$, $\lambda r:[x:Ind]([c_2:\text{run}(r.x)])$]

Polymorphic generalized quantifiers

Used in the treatment of dynamic generalized quantifiers.

If q is a quantifier predicate then

1. $\text{arity}(q) = \langle (X \sqsubseteq [x:Ind]) RecType, (X \sqsubseteq [x:Ind]) RecType \rangle$
2. q is associated with a relation between sets q^* (the relation between sets from classical generalized quantifier theory) such that

$$p : q(f_1 : (T_1) RecType, f_2 : (T_2) RecType)$$

iff

$$p = \langle \{a | \exists r : T_1 [f_1(r) \text{ is non-empty} \wedge a = r.x]\}, \\ \{a | \exists r : T_2 [f_2(r) \text{ is non-empty} \wedge a = r.x]\} \rangle$$

and q^* holds between p_1 and p_2

Dynamic generalized quantifiers

$$(11) \quad q(f_1 : (T)RecType, f_2 : (\mathcal{F}(f_1))RecType)$$

where: $\mathcal{F}(f)$ is the *fixed-point type* of f , i.e. $a : \mathcal{F}(f) \rightarrow a : f(a)$

$$(12) \quad \left[c : \text{the}(\lambda r : [x:Ind]([c_1:\text{donkey}(r.x)])), \lambda r : \left[\begin{array}{l} x : Ind \\ c_1 : \text{donkey}(x) \end{array} \right]([c_2:\text{run}(r.x)]) \right]$$

Conservativity: The donkey runs \leftrightarrow The donkey is a donkey which runs

Additional constraints on arguments

take forever

$$(13) \quad \lambda r: \left[\begin{array}{l} x : Ind \\ c_1: \text{event}(x) \end{array} \right] ([c_2: \text{take_forever}(r.x)])$$

be delicious

$$(14) \quad \lambda r: \left[\begin{array}{l} x : Ind \\ c_1: \text{food}(x) \end{array} \right] ([c_2: \text{be_delicious}(r.x)])$$

Additional constraints on nouns

(15) a. $\lambda r: \left[\begin{array}{l} x : Ind \\ c_1 : \text{food}(x) \end{array} \right] ([c_2 : \text{blancmange}(r.x)])$

b. $\lambda r: \left[\begin{array}{l} x : Ind \\ c_1 : \text{event}(x) \end{array} \right] ([c_2 : \text{game}(r.x)])$

c. $\lambda r: \left[\begin{array}{l} x : Ind \\ c_1 : \text{food}(x) \\ c_2 : \text{event}(x) \end{array} \right] ([c_3 : \text{lunch}(r.x)])$

Dynamic quantification

the game took forever

$$(16) \quad \text{the}(\lambda r: \left[\begin{array}{l} x : \text{Ind} \\ c_1 : \text{event}(x) \end{array} \right] ([c_2 : \text{game}(r.x)]), \\ \lambda r: \left[\begin{array}{l} x : \text{Ind} \\ c_1 : \text{event}(x) \\ c_2 : \text{game}(x) \end{array} \right] ([c_3 : \text{took_forever}(r.x)]))$$

Dynamic quantification and *lunch*

- (17) a. $\text{the}(\lambda r: \begin{bmatrix} x : \text{Ind} \\ c_1 : \text{food}(x) \\ c_2 : \text{event}(x) \end{bmatrix} ([c_3 : \text{lunch}(r.x)]))$,
- $\lambda r: \begin{bmatrix} x : \text{Ind} \\ c_1 : \text{food}(x) \\ c_2 : \text{event}(x) \\ c_3 : \text{lunch}(x) \end{bmatrix} ([c_4 : \text{took_forever}(r.x)])$
- b. $\text{the}(\lambda r: \begin{bmatrix} x : \text{Ind} \\ c_1 : \text{food}(x) \\ c_2 : \text{event}(x) \end{bmatrix} ([c_3 : \text{lunch}(r.x)]))$,
- $\lambda r: \begin{bmatrix} x : \text{Ind} \\ c_1 : \text{food}(x) \\ c_2 : \text{event}(x) \\ c_3 : \text{lunch}(x) \end{bmatrix} ([c_4 : \text{be_delicious}(r.x)])$

6. Treating lexical innovation

The game was delicious

$$(18) \quad \text{the}(\lambda r: \left[\begin{array}{l} x : Ind \\ c_1 : \text{event}(x) \end{array} \right] ([c_2 : \text{game}(r.x)])), \\ \lambda r: \left[\begin{array}{l} x : Ind \\ c_1 : \text{event}(x) \\ c_2 : \text{game}(x) \end{array} \right] ([c_3 : \text{be_delicious}(r.x)]))$$

Lexical resource for *be delicious*:

$$\lambda r: \left[\begin{array}{l} x : Ind \\ c_1 : \text{food}(x) \end{array} \right] ([c_2 : \text{be_delicious}(r.x)])$$

$$\left[\begin{array}{l} x : Ind \\ c_1 : \text{event}(x) \\ c_2 : \text{game}(x) \end{array} \right] \not\sqsubseteq \left[\begin{array}{l} x : Ind \\ c_1 : \text{food}(x) \end{array} \right]$$

The use of *be delicious* is *innovative* with respect to this resource.

⇐ contents

Coercion and dynamic quantification

- coercion in *the game was delicious* results from the general treatment of dynamic quantification
- but what would an appropriate proof object be?

Proof objects for non-mathematical types

- Martin-Löf – Proof objects for mathematical propositions (as types)
- Ranta – events as proof objects for non-mathematical propositions (as types), e.g. a proof of *Amundsen flew over the North Pole* is a (Davidsonian) event of flying by Amundsen over the North Pole
- Ranta’s “problem” – natural language predicates like *man* and *tree* do not correspond to sets in the constructive mathematical sense
- the “problem” seems greater with predicates like *delicious* which is both vague and subjective

Ranta's three strategies

- work with *types* rather than sets
- develop techniques of *approximation*
- study delimited *models* of language use

It seems to me that all three of these should be developed in an adequate type-theoretical approach to natural language.

In particular, ...

Inexact types

- It seems important that we recognize that human beings reason with *inexact* types for which they do not have (or indeed there may not even exist) clearly defined procedures for determining whether objects belong to those types or not.
- Such types may be refined by successive approximations as a result of being exposed to natural language utterances in a variety of situations.
- We have the ability to “tie down” certain expressions in a given restricted domain, “for the purposes of discussion” or at least to know what certain complex expressions would mean under the assumption that exact interpretations were assigned to their constituent parts.

Inexact types in innovative uses

- $be_delicious(g)$ for some game g
- the type itself may be well-defined as an object
- but not the objects (proof-objects) which belong to the type
- we may accept as being sufficient information for present purposes
- or we may clarify: *what do you mean “delicious”?*

Importance of inexact types for innovation

- inexactitude plays an important role in allowing innovation
- *?the game was divisible by three*

Resource based coercions

The blancmange took forever

(19) *exploiting dynamic quantification*

$$\begin{aligned} & ?\text{the}(\lambda r: \left[\begin{array}{l} x : \text{Ind} \\ c_1 : \text{food}(x) \end{array} \right] ([c_2 : \text{blancmange}(r.x)])), \\ & \lambda r: \left[\begin{array}{l} x : \text{Ind} \\ c_1 : \text{food}(x) \\ c_2 : \text{blancmange}(x) \end{array} \right] ([c_3 : \text{took_forever}(r.x)]) \end{aligned}$$

(20) *exploiting qualia in the resource lexicon*

$$\begin{aligned} & \text{the}(\lambda r: \left[\begin{array}{l} x : \text{Ind} \\ c_1 : \text{event}(x) \end{array} \right] ([c_2 : \text{preparation_of_blancmange}(r.x)]), \\ & \lambda r: \left[\begin{array}{l} x : \text{Ind} \\ c_1 : \text{event}(x) \\ c_2 : \text{preparation_of_blancmange}(x) \end{array} \right] ([c_3 : \text{took_forever}(r.x)]) \end{aligned}$$

Preferred readings and degrees of innovation

Exploiting resources yields a preferred reading with a robust intuition that it is “less innovative” than non-resource based coercions.

7. Conclusion: Towards a theory of linguistic innovation

Natural languages as resources

- Natural languages are not formal languages
- ...but they are toolboxes which allow us to create formal languages

From GF to linguistic theory

- GF is a grammar engineering application, not a linguistic theory
- ...but it suggests a view of natural languages as collections of resources
- some components that would need to be added to obtain a theory of linguistic innovation
 - Agent-related resources
 - Lexical semantics
 - Speaker coordination

Agent-related resources

- theory of agents with individual linguistic resources
- differs from standard linguistic theories in that the resources are used to construct domain specific grammars/languages

Lexical semantics

- GF provides no semantics in resource grammars
- eschews any general semantic theory
- semantics represented by abstract syntax associated with a given application
- natural languages do not have a fixed interpretation, but provide tools for creating languages *ad hoc* with interpretations appropriate to the purposes at hand
- but speakers are constrained by meanings associated with words in their previous experience (*open* can't suddenly be used to mean *close*).

Speaker coordination

- speakers adjust their language so that it is similar to that of interlocutor
- particularly important in the case of innovation
- learning GF grammars on the basis of input?
- adjusting the resources on the basis of (a greater amount of) input
- *i.e.* dynamic grammar construction on the basis of resources *and* dynamic resources which change as agent is exposed to new linguistic situations