

Agreement and Types in Natural Language

Informatics 2A: Lecture 22

John Longley

7 November 2013

Type-related phenomena in NL

- We've met the concept of **types** in programming languages, along with the idea of **typing constraints** on programs.
- Types also play a variety of roles in NL: e.g.
 - for disambiguation (via **selectional restrictions**),
 - for NL semantics (as in upcoming lecture).
- Furthermore, some phenomena that would be typically handled via types in a PL context (notably **agreement**) are often handled in other ways in NL.

We'll briefly survey this material in this lecture.

Agreement phenomena

In PLs, typing rules enforce **type agreement** between different (often separated) constituents of a program:

```
int i=0; ...; if (i>2) ...
```

There are somewhat similar phenomena in NL: constituents of a sentence (often separated) may be constrained to agree on an attribute such as person, number, gender.

- You, I imagine, **are** unable to attend.
- The **hills are** looking lovely today, **aren't they?**
- **He** came very close to injuring **himself**.

Agreement in various languages

These examples illustrate that in English:

- Verbs agree in **person** and **number** with their subjects;
- Tag questions agree in **person**, **number**, **tense** and **mode** with their main statement, and have the opposite **polarity**.
- Reflexive pronouns follow suit in **person**, **number** and *gender*.

French has much more by way of agreement phenomena:

- Adjectives agree with their head noun in gender and number.

Le petit chien, La petite souris, Les petites mouches

- Participles of *être* verbs agree with their subject:

Il est arrivé, Elles sont arrivées

- Participles of other verbs agree with preceding direct objects:

Il a vu la femme, Il l'a vue

How can we capture these kinds of constraints in a grammar?

Agreement rules: why bother?

Modelling agreement is obviously important if we want to **generate** grammatically correct NL text.

But even for **understanding** input text, agreement can be useful for resolving ambiguity.

E.g. the following sentence is ambiguous . . .

The boy who eats flies ducks.

. . . whilst the following are less so:

The boys who eat fly ducks.

The boys who eat flies duck.

Node-splitting via attributes

One solution is to refine our grammar by splitting certain non-terminals according to various *attributes*. Examples of attributes and their associated values are:

- **Number**: singular, plural
- **Person**: 1st, 2nd, 3rd
- **Gender**: masculine, feminine, neuter
- **Case**: nominative, accusative, dative, ...
- **Tense**: present, past, future, ...

In principle these are language-specific, though certain common patterns recur in many languages.

We can then split phrase categories as the language demands, e.g.

- Split NP on person, number, case (e.g. NP[3,sg,subj]),
- Split VP on person, number, tense (e.g. VP[3,sg,fut]).

Parameterized CFG productions

We can often use such attributes to enforce agreement constraints. This works because of the **head phrase structure** typical of NLS. E.g. we may write parameterized rules such as:

$$\begin{array}{l} S \rightarrow NP[p,n,subj] VP[p,n] \\ NP[3,n,c] \rightarrow Det[n] Nom[n] \end{array}$$

Each of these really abbreviates a finite number of rules obtained by specializing the attribute variables. (Still a CFG!) When specializing, each variable must take the same value everywhere, e.g.

$$\begin{array}{l} S \rightarrow NP[3,sg,subj] VP[3,sg] \\ S \rightarrow NP[1,pl,subj] VP[1,pl] \\ NP[3,pl,obj] \rightarrow Det[pl] Nom[pl] \end{array}$$

Parsing algorithms can be adapted to work with this machinery: don't have to 'build' all the specialized rules individually.

Example: subject-verb agreement in English

S	→	NP[p,n,subj] VP[p,n]
NP[p,n,c]	→	Pro[p,n,c]
Pro[1,sg,subj]	→	<i>I</i> , etc.
Pro[1,sg,obj]	→	<i>me</i> , etc.
NP[3,n,c]	→	Det[n] Nom[n] RelOpt[n]
Nom[n]	→	N[n] Adj Nom[n]
N[sg]	→	<i>person</i> , etc.
N[pl]	→	<i>people</i> , etc.
RelOpt[n]	→	ϵ <i>who</i> VP[3,n]
VP[p,n]	→	VV[p,n] NP[p',n',obj]
VV[p,n]	→	V[p,n] BE[p,n] VG
V[3,sg]	→	<i>teaches</i> , etc.
BE[p,n]	→	<i>is</i> , etc.
VG	→	<i>teaching</i> , etc.

(Other rules omitted.)

Types in Natural Language Semantics

Types are also very useful if we wish to describe the **semantics** (i.e., meaning) of natural languages. For example, we can use types employed in **logic** to model the meanings of various phrase types.

Basic Types

- 1 ***e*** — the type of real-world *entities* such as Inf2a, Stuart, John.
- 2 ***t*** — the type of *facts with truth value* like 'Inf2a is amusing'.

From these two basic types, we may construct more complex types via the **function type** constructor.

From basic to complex formal types

Where PL people write $\sigma \rightarrow \tau$, NL people often write $\langle \sigma, \tau \rangle$. E.g.:

- $\langle e, t \rangle$: **unary predicates** – functions from entities to facts.
- $\langle e, \langle e, t \rangle \rangle$: **binary predicates** – functions from entities to unary predicates.
- $\langle \langle e, t \rangle, t \rangle$: **type-raised entities** – functions from unary predicates to truth values.

- Inf2a, Stuart : e
- enjoys : $\langle e, \langle e, t \rangle \rangle$
- enjoys Inf2a, is amusing : $\langle e, t \rangle$
- Inf2a is amusing, Stuart enjoys Inf2a : t
- every student : $\langle \langle e, t \rangle, t \rangle$

This simple system of types will be enough to be going on with (see Lecture 24). But for more precise semantic modelling, a much richer type system is desirable.

Different types of entities in NL

We can distinguish those entities we can **count** and those we can't:

- A student kept **a chicken** in her room.
 - A student kept **two chickens** in her room.
 - I ate **rice** and drank **milk**.
 - *I ate **two rices** and drank **two milks**.
-
- **individuals** (things we can count): one student, two students, one chicken, many chickens, one room, many rooms
 - **mass** (things we can't count): rice, milk

Switching types: Coercion

Sometimes changing syntactic form changes type interpretation:

- I drank **beer** last night.
- I had **a beer** last night. (\Rightarrow one glass of beer)
- I tasted **two beers** last night. (\Rightarrow two kinds of beer)
- I ate **chicken** for lunch. (\Rightarrow some meat, not one chicken!)
- There were **chickens** all over the floor. (\Rightarrow many chickens)
- There was **chicken** all over the floor. (\Rightarrow bits of chicken)

This is one illustration of **type coercion** in Natural Language.

Subtypes in NL

```
hamburger <: sandwich <: food item <: food  
<: substance <: matter <: physical entity <: entity
```

- To deal with meanings in NL, more fine-grained classifications (of varying levels of specificity) are often useful.
- There are also many other more abstract types of entities to which a NL expression may refer: e.g., locations, points in time, time spans, events, beliefs, desires, possibilities, ...
- This leads to a vast system of subtypes capturing information about real-world concepts and their relationships. (Cf. the **WordNet** database.)

Selectional restrictions

We can often characterize verbs and other predicates in terms of their **selectional restrictions** — constraints on the type of entities or expression can serve as their arguments.

- I want to eat somewhere close to Appleton Tower.
- I want to eat some Thai food.

How do we know that *Thai food* is the **object** of the eating event in the second sentence, and that *somewhere close to AT* is the **location** of the eating event in the first?

- The **object** of eating is usually something *edible*: Its semantic type is *edible things*.
- The **location** of an event is usually a *place*: Its semantic type is *location*.

Selectional restrictions

Selectional restrictions are associated with word **senses**, not words:

- Do any international airlines serve vegan meals?
(ie, *provide food or drink*)
- Do any international airlines serve Edinburgh?
- (ie, *provide a service*)
- ?? Do any international airlines serve Edinburgh and vegan meals?

Selectional restrictions vary in their specificity:

OBJECT(imagine): a situation

OBJECT(diagonalise): a matrix

⇒ Verbs vary in the specificity of their argument types.

Selectional restrictions and type coercion

Selectional restrictions can change the way we interpret a term:

- Jane Austen wrote 'Emma'.
- I used to read Jane Austen a lot.

- The chicken was domesticated in Asia.
- The chicken was overcooked.

Metonymy is when the referent of a term changes to a related entity, often associated with the demands of a verb's **selectional restrictions**.

Summary

- Many agreement phenomena in NL can be modelled using CFGs with attributes.
- Type systems are also useful in semantic modelling.
- To capture selectional restrictions associated with verb arguments, a very rich system of subtypes is desirable.
- Type coercion is common in Natural Language: changing the type (and often the referent) of an expression to one that fits the verb (predicate) to which it serves as an argument.