The Problem of Morphological Variation

(Ranta 2011)

Example

(1) this wine is Italian

(2) these wines are Italian

```
fun
  Pred : Item_Sg -> Quality -> Comment ;
  Pred : Item_Pl -> Quality -> Comment ;
  This , that : Kind_Sg -> Item_Sg ;
  These : Kind_Pl -> Item_Pl ;

lin
  Pred item quality = item ++ "is" ++ quality ;
  Pred item quality = item ++ "are" ++ quality ;
```

In Italian, gender is to be added as well

Parameters and Tables

Example (Parameters)

```
param Number = Sg | Pl
```

Example (Tables)

number	**form**
singular	*pizza*
plural	*pizze*

```
table {Sg => "pizza" ; Pl => "pizze"}
```

Parameters and Tables

Definition (Hypotheses and contexts)

A *context* is a sequence of *hypotheses*, i.e., variable-type pairs. It is written:

$$(x : T)$$

Definition (Parameter Type Definitions)

A *parameter type definition*

$$\texttt{param}\ P = C_1\ G_1 \mid \ldots \mid C_n\ G_n$$

defines a parameter type P with the *parameter constructors* $C_1, \ldots, C_n$ with their respective contexts $G_1, \ldots, G_n$.

- Dependent types are not available in parameter type definitions, so the use of variables is never necessary
- Types in the context must themselves be parameter types

Parameters and Tables

Definition (Parameter Types)

Parameter types

- Given the judgment `param` $P \ldots$, P is a parameter type.
- A record type of parameter types is a parameter type.
- `Ints` n (integers from 0 to n) is a parameter type

Parameter type may not be recursive.

Example (Inflection of an Italian adjective)

```
table {
  Masc => table {Sg => "caldo" ; Pl => "caldi"}
  Fem => table {Sg => "calda" ; Pl => "calde"}
}
```

`table {Sg => pizza ; Pl =>"pizze"} ! Pl` $\Downarrow$ `"pizze"`

Variable and Inherent Features

Variable Features

In English or Italian, nouns have both singular and plural forms. The number is a *variable feature* of nouns.

Inherent Features

In Italian, a noun *is* either masculine or feminine. The gender is an *inherent feature* of Italian nouns.

Example (Agreement)

- Adjectival modification of nouns: the variable gender of the adjective is determined by the inherent gender of the noun.
- Determination: the variable number of the noun is determined by the inherent number of the determiner.

$\Rightarrow$ Asymmetry (contrary to unification grammars).

Types

Linearization Types (Ljunglöf 2004)

- `Str`, the type of strings, is a linearization type.
- If $T_1, \ldots T_n$ are linearization types or parameter types, and at least one of them is a linearization type, then the record type $\{r_1 : \; T_1; \ldots; r_n : T_n\}$ is a linearization type.
- if T is a linearization type and P is a parameter type, then $P \Rightarrow T$ is a linearization type.

Linearization Types and Agreement

Example (English)

```
param
  Number = Sg | Pl ;

lincat
  Comment = {s: Str} ;                  -- a full sentence
  Item    = {s: Str ; n: Number} ;  -- a noun phrase
  Kind    = {s: Number => Str} ;    -- a noun
  Quality = {s: Str} ;                  -- an adjective

lin
  This kind = {s = "This" ++ kind.s ! Sg ; n = Sg } ;
  Mod qual kind = {s = table {n => qual.s ++ (kind.s ! n)}} ;

  Pred item qual =
    { s = item.s ++
          table { Sg => "is" ; Pl => "are" } ! item.n ++
          qual.s } ;
```

Abstract Types

(Ljunglöf 2004)

Definition (Category Declarations)

A *category declaration*

$$\texttt{cat} \;\; C = CG$$

defines the basic types of abstract syntax. A basic type is formed from a category by giving values to all variables in the *context* G.
If the context is empty, the basic type looks the same as the category itself.
Otherwise, application syntax is used: $C \; a_1 \; \ldots \; a_n$

Definition (Function declaration)

A *function declaration*

$$\texttt{fun} \; f \; : \; T$$

defines the *syntactic constructors of abstract syntax*.

An abstract syntax is *context-free* if it has neither dependent types nor higher-order functions.

Questions

- Parsing with non linearity (and deletion)?

  > *context-free GF is strongly equivalent to PMCFG. This equivalence is shown by giving an algorithm converting cf-GF grammars into PMCFG grammars recognizing the same language; and by showing that parse results can be converted back efficiently. The conversion algorithm consists of enumerating all parameter instantiations in a linearization, and then moving the instantiated parameters to the abstract categories. Enumerating all instantiations may lead to an exponential increase of the grammar size. Therefore two alternative conversion algorithms are given, which do not enumerate all possible instantiations, but instead try to only instantiate when it is necessary. (Ljunglöf 2004)*

- Parsing algorithms (Ljunglöf 2004; Angelov 2009; Ranta 2007b) and differences with (Salvati 2010).
- Differences between features at the object or at the abstract level (Ranta 2007a).
- Permutative conversions?

Abstract and Concrete Syntax

Linearization

`lincat` $C = L$	C has the linearization type of L
`lin` $f\ x_1\ \ldots\ x_\delta = t$	f has the linearization function $\lambda x_1 \ldots x_\delta . t$
`lindef` $C\ x = t$	C has default linearization $\lambda x.t$

$$
\begin{aligned}
(C\ a_1\ \ldots\ a_n)^\circ &= L \text{ if } \texttt{lincat}\ C = L \\
((x_1 : A_1) \to \cdots \to (x_n : A_n) \to A)^\circ &= \texttt{Str} \to \cdots \to A^\circ
\end{aligned}
$$

Canonical Linearization

The concrete syntax of any GF grammar can be partially evaluated to a grammar in canonical form (Ranta 2004):

- All local and global definitions disappear, as well as function applications;
- all tables are instantiated (all patterns are variable-free);
- Hierarchical parameters can be flattened (assumption that parameters are declared by giving a finite set of parameter types

Canonical Linearization Term

Definition (Canonical Term)

A *canonical linearization term* is of the following form:

- A string constant is of type `Str`; and a concatenation $s_1 ++ s_2 : \texttt{Str}$ whenever $s_1, s_2 : \texttt{Str}$;
- A constant parameter $p : P$, whenever $p \in P$;
- A record $\{r1 = \phi_1; \ldots; r_n = \phi_n\}$ is of type $T = \{r_1 : T_1; \ldots; r_n : T_n\}$ whenever each $\phi_i : T_i$;
- A record projection $\phi.r_i : T_i$ whenever ϕ is of the record type $T = \{r_1 : T_1; \ldots; r_n : T_n\}$;
- A table $[p_1 \Rightarrow \phi_1; \ldots; p_n \Rightarrow \phi_n]$ is of type $P \Rightarrow T$ whenever $P = \{p_1, \ldots, p_n\}$ and each $\phi_i : T$;
- A table selection $\phi \,!\, \psi : T$ whenever $\phi : P \Rightarrow T$ and $\psi : P$
- An argument variable $x_i : B_i^\circ$.

Canonical Linearization

Example (Ljunglöf 2004, p.47)

$$vp^{\circ}(x, y) = \{s = [z \Rightarrow x.s\,!\,z +\!+\, y.s]\}$$

and

$$vp^{\circ}(x, y) = \{s = [\mathtt{Sg} \Rightarrow x.s\,!\,\mathtt{Sg} +\!+\, y.s; \mathtt{Pl} \Rightarrow x.s\,!\,\mathtt{Pl} +\!+\, y.s]\}$$

Computation Rules

$$\begin{aligned} s_1 +\!+\, s_2 &= s_1 s_2 \\ \{\ldots; r = t; \ldots\}.r &= t \\ [\ldots; p \Rightarrow t; \ldots]\,!\,p &= t \end{aligned}$$

Generalized Context-Free Grammars (Pollard 1984)

Abstract Grammar A tuple $(\mathcal{C}, S, \mathcal{F}, \mathcal{R})$. For each function symbol $f \in \mathcal{F}$ there is an associated context-free syntax rule:

$$A \longrightarrow f[B_1, \ldots, B_\delta]$$

Concrete Interpretation To each function symbol f i associated a partial linearization function f°

$$f^\circ \in b_1^\circ \times \cdots \times B_\delta^\circ \to A^\circ$$

Variable-Free Notation For a rule $A \longrightarrow f[A_1, \ldots, A_\delta]$ and a linearization $f^\circ(x_1, \ldots, x_\delta) = \phi$ can be rewritten as:

$$A \longrightarrow f[A_1, \ldots, A_\delta] := \hat{\phi}$$

where each occurrence of the variable x_i in ϕ is replaced by the term A_i in $\hat{\phi}$.

Example

$$\begin{aligned} A \longrightarrow f[B^1, A, B^2] \quad &:= aB^1 AbB^3 \\ f^\circ(x, y, z) \quad &= axybx \end{aligned}$$

Generalized Context-Free Grammars

Definition (Part of a term)

If there is a bijective function $\pi : T \to P_1 \times \cdots \times P_n$, Π is said to *form a partition of* T.
Given a term $t : T$, a projected term $p_k : P_k$ os *a part of t* if there is some partition π of T such that $p_k = \pi_k(\pi(t))$.

Subclasses of GCFG

Given a GCFG rule $A \longrightarrow f[B_1, \ldots, B_\delta]$ with its linearization $f(x_1, \ldots, x_\delta) = phi$, the rule is said:

Parallel	if some part of x_i is mentioned twice in ϕ
Linear	if no part of x_i is mentioned twice
Erasing	if some part of x_i is not mentioned at all in ϕ
Non erasing	if all parts of x_i are mentioned in ϕ
Suppressing	if x_i is not mentioned at all in ϕ

Parallel Multiple Context-Free Grammars (Kasami, Seki, and Fujii 1989; Seki et al. 1991)

Definition (PMCFG)

A GCFG such that:

- Linearization types are restricted to tuples of strings
- The only allowed operations in linearization functions are ruple projections and string concatenation

$$f^\circ(\langle x_{1,1}, \ldots, x_{1,d_1} \rangle, \ldots, \langle x_{\delta,1}, \ldots, x_{\delta,d_\delta} \rangle) = \langle \alpha_1, \ldots, \alpha_n \rangle$$

where each α_i is a sequence of variables $x_{j,k}$ or constant strings.
Or, in *record notation*,

$$f^\circ(x_1, \ldots, x_\delta) = \{\mathbf{1} = \hat{\alpha_1}, ; \ldots ; \mathbf{d} = \hat{\alpha_d}\}$$

where each $x_{j,k}$ in α_i is replaced by the projection $x_j.\mathbf{k}$ in $\hat{\alpha}_i$.

PMCFG and GF

Theorem (Ljunglöf 2004)

Every PMCFG is equivalent to a context-free GF grammar.

Theorem (Ljunglöf 2004)

And vice-versa.

Proof.

Easy if, in GF grammars:

- all tables and table selections are instantiated (canonical linearization)
- records containing parameters are not allowed

If there are records that contains a parameter (e.g., $d_m^\circ = \{s =' many'; n = \texttt{Pl}\}$), requires more work...

$\square$

What about ACG?

Theorem

2nd-order ACG are equivalent to linear PMCFG.

What to look at next?

"Building PMCFG Parsers as Datalog Program Transformations" (Ball et al. 2014).

Questions: What extensions to ACGs to make them equivalent to these programs?

Features at the object level (Ranta 2007a)

See the demo.

Bibliography I

- Angelov, Krasimir (2009). "Incremental Parsing with Parallel Multiple Context-Free Grammars". In: *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*. Athens, Greece: Association for Computational Linguistics, pp. 69–76. ACL anthology: E09-1009.
- Ball, Arthur et al. (2014). "Building PMCFG Parsers as Datalog Program Transformations". In: *Logical Aspects of Computational Linguistics*. Ed. by Nicholas Asher and Sergei Soloviev. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–13. DOI: 10.1007/978-3-662-43742-1_1.
- Kasami, Tadao, Hiroyuki Seki, and Mamoru Fujii (1989). "Generalized context-free grammars and multiple context-free grammars". In: *Systems and Computers in Japan* 20.7, pp. 43–52. DOI: 10.1002/scj.4690200705.
- Ljunglöf, Peter (2004). "Expressivity and Complexity of the Grammatical Framework". PhD thesis. Chalmers University of Technology and Göteborg University. URL: http://www.cse.chalmers.se/~peb/pubs/Ljunglof2004a%20-%20Expressivity%20and%20Complexity%20of%20the%20Grammatical%20Framework.pdf.

Bibliography II

- Pollard, Carl (1984). "Generalized Phrase Structure Grammars, Head Grammars, and Natural Language". PhD thesis. Stanford University, CA.
- Ranta, Aarne (2004). "Grammatical Framework: A Type-Theoretical Grammar Formalism". In: *Journal of Functional Programming* 14.2, pp. 145–189. DOI: 10.1017/S0956796803004738.
- Ranta, Aarne (2007a). "Features in Abstract and Concrete Syntax". In: *The 2nd International Workshop on Typed Feature Structure Grammars, 16th Nordic Conference of Computational Linguistics (NODALIDA-2007) workshop*. Tartu. URL: http://www.cse.chalmers.se/~aarne/articles/ranta-tfsg2007.pdf.
- Ranta, Aarne (2007b). "The GF Grammar Compiler". In: *Workshop on New Directions in Type-theoretic Grammars, ESSLLI workshop*. Dublin. URL: http://www.cse.chalmers.se/~aarne/articles/ranta-tfsg2007.pdf.
- Ranta, Aarne (2011). *Grammatical Framework. Programming with Multilingual Grammars*. CSLI Studies in Computational Linguistics. CSLI Publications.

Bibliography III

Salvati, Sylvain (2010). "On the membership problem for non-linear Abstract Categorial Grammars". In: *Journal of Logic, Language and Information* 19.2, pp. 163–183. DOI: 10.1007/s10849-009-9110-0.

Seki, Hiroyuki et al. (1991). "On Multiple Context-Free Grammars". In: *Theoretical Computer Science* 88.2, pp. 191–229. DOI: 10.1016/0304-3975(91)90374-B.