Analyse grammaticale du français : des concepts théoriques ou de la bidouille ?

Christophe Raffalli

LAMA

Chambéry 2009

Le problème :

Étant donnée une phrase, trouver les arbres grammaticaux qui peuvent lui correspondre en français.

Restrictions actuelles

- Phrase affirmative avec verbe.
- Sinon prendre en compte le maximum de choses.

Outils théoriques ?

- Théorie des types (GF de Aarne Ranta)
- MELL, IMELL (Bruno Guillaume)
- ...

Est-ce utile ?

Ne manque-t-il pas juste un vrai outils pratique pour rendre le code raisonnable ?

Quels outils ? Stanford NLP ?

dypgen by Emmanuel Onzon

http ://dypgen.free.fr

Les outils

GLR parsing

- Grammaires ambigües et $O(n^3)$.
- Efficace sur les grammaires non ambigües.

Ça ne suffit pas !

- Problèmes de morphologie (lien parser/lexer).
- Hors context avec duplication.

Les plus dans dypgen

- Grammaires et lexers extensibles (non utilisé)
- Génération du lexer (pratique mais indispensable)
- Local et Global Data
- Action prématurée
- Rejet d'une règle
- Pattern-matching sur le résultat des actions
- Priorité par relation
- Contrôle du layout

Contrôle du Layout et priorité

```
%layout [' ' '\t' '\n' '\r']

%relation atom<comma
%relation atom<list

adj:
  adj(<=comma) ("et" | "ou") adj(=atom)
    list
| adj(<=comma) - "," adj(=atom)
    comma
```

Pattern-matching sur le résultat des actions

```
%relation atom<avec_adj

noun:
| adj(<=list)<AdjGauche|AdjBi> noun(=atom)
    avec_adj
| adj(<=atom)
    atom
```

Action prématurée et rejet d'une règle

```
%relation atom<avec_adj

noun:
| adj(<=list)<adj> noun(=atom)
    { if adj <> AdjGauche && adj <> AdjDroit
      then raise Giveup }
    avec_adj
| adj(<=atom)
    atom
```

Tous les langages récursifs !

Global data

Gestion des accords : résolutions de contraintes
Calcul des prédicats existentiels avec des symboles de fonctions et l'égalité.

Exemple 1 : «elle les a mangés»

x : nbr(elle), y : nbr(les)z : nbr(a), t : nbr(mangés) $\vdash x = z \wedge y = t$

Exemple 2 : «toi et moi»

x : person(toi), y : person(moi), z : person(toi et moi) $\vdash z = f(x, y)$

Global data

- D'habitude : les contraintes dans une variable globale

  Beurk

  Impossible avec GLR : global data

```
groupe_nominal:
| det<tr1,gn1> nom<(tr2,gn2,Commun)>
    @{
        let st = dyp.global_data in
        let st = unify2 st gn1 gn2 in
        (Node[tr1;tr2],gn1), [Global_data(st)]
    } atom
```

Global data

- D'habitude : les contraintes dans une variable globale
- Beurk

  Impossible avec GLR : global data

```
groupe_nominal:
| det<tr1,gn1> nom<(tr2,gn2,Commun)>
    @{
        let st = dyp.global_data in
        let st = unify2 st gn1 gn2 in
        (Node[tr1;tr2],gn1), [Global_data(st)]
    } atom
```

Global data

- D'habitude : les contraintes dans une variable globale
- Beurk
- Impossible avec GLR : global data

```
groupe_nominal:
| det<tr1,gn1> nom<(tr2,gn2,Commun)>
    @{
        let st = dyp.global_data in
        let st = unify2 st gn1 gn2 in
        (Node[tr1;tr2],gn1), [Global_data(st)]
     } atom
```

Merge et global data

- Lorsqu'un parser GLR produit 2 parse-trees avec les mêmes tokens, pour le même non-terminal, il peut appeler une fonction `merge` fournie par le programmeur.
- Permet du partage.
- Bon pour la compléxité ?

Et les contraintes ?

On fait une disjonction !

Merge et global data

- Lorsqu'un parser GLR produit 2 parse-trees avec les mêmes tokens, pour le même non-terminal, il peut appeler une fonction `merge` fournie par le programmeur.
- Permet du partage.
- Bon pour la compléxité ?

Et les contraintes ?

On fait une disjonction !

Local Data

```
que: "que" @{
  Leaf($1), [Local_data
    { dyp.local_data with
        sentence_kind = `Que_sentence }] }

gn:
|  gn que phrase {...}

is_not_que: {
   if dyp.local_data.sentence_kind = `Que_sentence
   then raise Giveup; }

gv:
| is_not_que gv<{cod = None}> accusative gn { ... }
```

Conclusion

- Est-ce de la recherche ?
- Faut-il continuer ?
- Quelqu'un veut-il s'amuser avec moi ?
- Et la sémantique ?
  - Connexion avec le travail d'Humayoun.
  - Sémantique statistique.