Héritage Multiple et Templates dans l'Implantation de HPSG

Graham Wilcock

Centre for Computational Linguistics
University of Manchester Institute of Science and Technology
PO Box 88, Manchester M60 1QD, United Kingdom
graham@ccl.umist.ac.uk

Résumé

L'analyse des propositions relatives en anglais telle que décrite par Sag (1997) se base sur une classification à deux dimensions des constructions syntaxiques en HPSG. Nous présentons ici[1] une implémentation de cette analyse, fondée sur l'héritage multiple et les *templates* à deux dimensions dans le système ProFIT (Erbach, 1995).

1. Introduction

L'analyse des propositions relatives en anglais, proposée par Sag (1997) dans le cadre de HPSG, se base sur une classification à deux dimensions des constructions syntaxiques. Dans la première dimension, les constructions sont classifiées dans une hiérarchie de syntagmes fondée sur les schémas de dominance immédiate en HPSG (head-subject-phrase, head-complement-phrase, head-filler-phrase, ...). Dans la deuxième dimension, les constructions sont classifiées dans une hiérarchie de propositions (declarative clause, interrogative clause, *wh*-relative, non-*wh*-relative, ...). Chaque construction de proposition relative en anglais (par exemple, *wh*-subject relative clause) peut donc appartenir aux deux dimensions, classifiée et par type de syntagme (head-subject) et par type de proposition (*wh*-relative).

Nous décrivrons ici une implémentation informatisée qui traite toute la gamme de constructions de propositions relatives en anglais telles qu'analysées par Sag (1997). Cette implémentation, fondée sur l'héritage multiple et les *templates* (les patrons) à deux dimensions, utilise le système ProFIT: Prolog with Features, Inheritance and Templates (Erbach, 1995). Le système ProFIT donne à Prolog une extension pour traiter les structures de traits typées, qui se base sur l'héritage multiple (Section 2). ProFIT n'offre pas l'implémentation directe de contraintes dynamiques attachées aux types spécifiques. Il faut donc implémenter les contraintes de types au moyen des templates (Section 3).

[1] Voir le poster présenté à TALN–96 (Wilcock & Matsumoto, 1996).

Ces techniques sont ensuite employées dans l'analyse des constructions de propositions relatives en anglais (Section 4).

L'analyse présentée par Sag (1997) contient aussi d'autres idées: les règles lexicales pour la représentation des phénomènes d'extraction sans catégorie vide, l'amalgamation lexicalisée des traits non-locaux, et un traitement nouveau des complémenteurs. Ces idées ont aussi été incluses dans cette implémentation, mais nous ne les décrivons pas ici.

2. Héritage multiple

Mellish (1988) présente une implémentation efficace de la classification croisée multi-dimensionnelle dans le cadre de la grammaire systémique (Systemic Functional Grammar, SFG). S'inspirant des techniques de Mellish, Erbach (1994) propose à son tour une implémentation efficace de l'héritage multiple pour les structures de traits typées en HPSG et en SFG. Dans son système d'héritage multi-dimensionnelle, une structure de traits peut hériter des traits de plusieurs types supérieurs qui appartiennent à des hiérarchies distinctes.

```
phrase > [hd_ph, non_hd_ph] * [clause, non_clause].
  hd_ph > [hd_adj_ph, hd_nexus_ph]
        intro [hd_dtr:sign].
    hd_adj_ph > [simp_hd_adj_ph, hd_rel_ph]
              intro [adj_dtr:sign].
    hd_nexus_ph > [hd_fill_ph, hd_subj_ph, hd_spr_ph, hd_comp_ph].
      hd_fill_ph > [fin_hd_fill_ph, inf_hd_fill_ph]
                 intro [fill_dtr:sign].
        fin_hd_fill_ph > [hd_fill_decl_cl, fin_wh_fill_rel_cl].
        inf_hd_fill_ph > [inf_wh_fill_rel_cl].
      hd_subj_ph > [fin_hd_subj_ph, hd_subj_non_cl]
                 intro [subj_dtr:sign].
        fin_hd_subj_ph > [hd_subj_decl_cl, wh_subj_int_cl, wh_subj_rel_cl,
                          bare_rel_cl].
      hd_spr_ph  intro [spr_dtr:sign].
      hd_comp_ph > [hd_comp_non_cl, yes_no_int_cl, simp_inf_rel_cl, red_rel_cl]
                 intro [comp_dtrs].
  clause > [imp_cl, decl_cl, int_cl, rel_cl].
    decl_cl > [hd_subj_decl_cl, hd_fill_decl_cl].
    int_cl > [wh_subj_int_cl, yes_no_int_cl].
    rel_cl > [wh_rel_cl, non_wh_rel_cl, red_rel_cl].
      wh_rel_cl > [wh_subj_rel_cl, wh_fill_rel_cl].
        wh_fill_rel_cl > [fin_wh_fill_rel_cl, inf_wh_fill_rel_cl].
      non_wh_rel_cl > [bare_rel_cl, simp_inf_rel_cl].
  non_clause > [hd_subj_non_cl, hd_comp_non_cl, hd_adj_ph].
```

Figure 1: Hiérarchies de types syntagmatiques et propositionels

L'innovation présentée par Sag (1997) est de spécifier une classification croisée qui se base sur une hiérarchie de syntagmes et une hiérarchie orthogonale de propositions. Un certain nombre de contraintes est spécifié, dont il y a des contraintes attachées à des

types syntagmatiques (phrase types) specifiques et aussi des contraintes attachées à des types propositionels (clause types) spécifiques. Dans l'implémentation en ProFIT, il est possible de spécifier directement les deux hiérarchies dans le cadre de la hiérarchie globale parce que ProFIT offre un héritage à plusieurs dimensions. Cependant, les contraintes attachées à des types spécifiques ne sont pas prises en charge directement, elle se fait au moyen des templates.

Les hiérarchies de types syntagmatiques et de types propositionels spécifiées par Sag sont présentées en format ProFIT dans la Figure 1. On voit que les constructions y sont classifiées au niveau des deux dimensions distinctes (types syntagmatiques et types propositionels) en même temps. Cette classification croisée est implantée en ProFIT au moyen de l'héritage multiple. (Voir l'explication de l'héritage multiple en ProFIT présentée par Erbach (1994), bien illustrée par une version préliminaire de l'analyse de Sag). La spécification

```
phrase > [hd_ph, non_hd_ph] * [clause, non_clause]
```

signifie que les syntagmes sont classifiés dans la dimension syntagmatique comme syntagmes à Tête (headed phrases, `hd_ph`) ou syntagmes sans Tête (non-headed phrases, `non_hd_ph`), et sont classifiés d'autre part dans la dimension propositionelle comme propositions (`clause`) ou non (`non_clause`). Dans les deux dimensions, les hiérarchies locales de sous-types syntagmatiques et sous-types propositionnels sont spécifiées normalement.

3. Templates et contraintes

Avant de présenter l'implémentation des contraintes sur les types syntagmatiques spécifiques, il faut introduire des templates pour des contraintes plus générales. On voit dans la Figure 2 comment définir en ProFIT les templates pour implémenter les principes généraux de HPSG. Les templates sont définis par la symbole ':=', qui signifie que le nom du template à gauche sera remplacé par la structure de traits typée à droite. Les structures de traits typées ont la forme `attribut-1!valeur-1 & attribut-2!valeur-2`. Les noms de *types* ont la forme `<type`, et les valeurs partagées (structure-sharing) sont représentées par l'identité de variables Prolog (par exemple: `HF`, `Subj`).

```
/* Head Feature Principle */
'HFP' :=  syn!loc!cat!head!HF &
          hd_dtr!syn!loc!cat!head!HF.

/* Valency Principle */
'VALP'(subj) :=
    syn!loc!cat!val!subj!Subj &
    hd_dtr!syn!loc!cat!val!subj!Subj.
'VALP'(spr) :=
    syn!loc!cat!val!spr!Spr &
    hd_dtr!syn!loc!cat!val!spr!Spr.
'VALP'(comps) :=
    syn!loc!cat!val!comps!Comps &
    hd_dtr!syn!loc!cat!val!comps!Comps.
```

```
/* Empty Comps Constraint */
'ECC' := syn!loc!cat!val!comps![].

/* Semantics Principle */
'SEMP'(head) :=
    syn!loc!cont!Cont &
    hd_dtr!syn!loc!cont!Cont.
'SEMP'(adjunct) :=
    syn!loc!cont!Cont &
    adj_dtr!syn!loc!cont!Cont.
```

Figure 2: Templates pour les principes HPSG

3.1. Contraintes attachées aux syntagmes

L'expansion d'un template se fait partout où il est invoqué, ce qui est bien possible à l'intérieur d'un autre template. Les templates définis dans la Figure 2 sont invoqués dans les définitions des templates plus spécifiques pour les contraintes sur les syntagmes spécifiques présentés dans la Figure 3. Au niveau le plus haut de la hiérarchie syntagmatique, le template pour les syntagmes à Tête défini par `hd_ph := <hd_ph & @'HFP' & @'ECC'` indique que tous les syntagmes à Tête sont de type `<hd_ph`, ils suivent le Principe des traits de Tête par l'invocation (par `@`) du template `@'HFP'`, et ils suivent la Contrainte de COMPS Vide (Empty COMPS Constraint) par l'invocation du template `@'ECC'`.

```
hd_ph := <hd_ph & @'HFP' & @'ECC'.

hd_nexus_ph := <hd_nexus_ph &
  @hd_ph & @'SEMP'(head).

hd_subj_ph := <hd_subj_ph &
  @hd_nexus_ph &
  @'VALP'(spr) & @'VALP'(comps) &
  syn!loc!cat!val!subj![] &
  hd_dtr!syn!loc!cat!val!(
      subj![SubjSyn] &
      spr![]) &
  subj_dtr!syn!SubjSyn.

fin_hd_subj_ph := <fin_hd_subj_ph &
  @hd_subj_ph &
  syn!loc!cat!head!(<verb & vform!<fin).

hd_spr_ph := <hd_spr_ph &
  @hd_nexus_ph &
  @'VALP'(subj) & @'VALP'(comps) &
  syn!loc!cat!val!spr![] &
  hd_dtr!syn!loc!cat!val!(
      subj![] &
      spr![SprSyn]) &
  spr_dtr!syn!SprSyn.

hd_comp_ph := <hd_comp_ph &
  @hd_nexus_ph &
  @'VALP'(subj) & @'VALP'(spr).
```

```
hd_adj_ph := <hd_adj_ph &
  @hd_ph &
  @'VALP'(subj) & @'VALP'(spr) &
  @'VALP'(comps) &
  hd_dtr!syn!HeadSyn &
  adj_dtr!syn!loc!cat!head!mod!HeadSyn.

hd_fill_ph := <hd_fill_ph &
  @hd_nexus_ph &
  @'VALP'(subj) & @'VALP'(spr) &
  @'VALP'(comps) &
  syn!nonloc!slash!Slash &
  hd_dtr!syn!(
      loc!cat!head!<verbal &
      nonloc!slash![Fill|Slash]) &
  fill_dtr!syn!loc!Fill.

fin_hd_fill_ph := <fin_hd_fill_ph &
  @hd_fill_ph &
  hd_dtr!syn!loc!cat!(
      head!(<verb & vform!<fin) &
      val!subj![]).

inf_hd_fill_ph := <inf_hd_fill_ph &
  @hd_fill_ph &
  hd_dtr!syn!loc!cat!(
      head!(<comp & vform!<inf) &
      val!subj![@'PRO']).

'PRO' := <pro &
  loc!(cat!head!case!<acc &
       cont!(<refl & index!<ref)).
```

Figure 3: Templates pour les contraintes sur les syntagmes

Les syntagmes à Tête (`hd_ph`) sont classifiés par Sag (1997) comme syntagmes avec ajouts (head-adjunct phrases, `hd_adj_ph`) ou syntagmes sans ajout (head-nexus phrases, `hd_nexus_ph`). Les templates pour les `hd_adj_ph` et `hd_nexus_ph` invoquent donc le template pour les syntagmes à branche Tête (`@hd_ph`), qui invoque automatiquement les templates `@'HFP'` et `@'ECC'`. L'invocation de templates dans les autres templates est un moyen efficace pour implémenter l'héritage des contraintes attachées aux types spécifiques.

Le template pour `hd_adj_ph` force l'implémentation du Principe de Valence (Valency Principle) par l'invocation des trois parties de `@'VALP'`. Le template pour `hd_nexus_ph` force l'implémentation du Principe Sémantique par `@'SEMP'(head)`, en laissant le Principe de Valence aux sous-types `hd_subj_ph`, `hd_spr_ph` et `hd_comp_ph`. Les autres templates pour les types syntagmatiques dans la Figure 3 forcent l'implémentation des contraintes diverses attachées par Sag (1997) aux types syntagmatiques spécifiques.

Le template pour `inf_hd_fill_ph` (infinitival head-filler phrase) spécifie que le sujet de la Tête doit avoir le type *PRO* par l'invocation du template `@'PRO'`, aussi défini dans la Figure 3. Le *PRO* est toujours accusatif (ne peut jamais être le sujet d'un verbe indicatif), toujours réflexif (son liage suit la généralisation de Visser) et a toujours un indice référentiel (ne peut jamais être le sujet d'un syntagme verbal qui demande un sujet phrastique).

3.2. Contraintes attachées aux propositions

La hiérarchie de types dans la Figure 1 contient la hiérarchie de propositions telle que proposée par Sag (1997) ainsi que la hiérarchie des syntagmes. Il y a peu de contraintes attachées aux types spécifiques de propositions. Les contraintes générales sur toutes les propositions sont présentées par Sag comme (1).

(1)
$$\textit{clause} \quad \Rightarrow \quad \begin{bmatrix} \text{SUBJ} & \text{list}(\textit{PRO}) \\ \text{HEAD} & \left[\text{MOD} \;/\; \textit{none}\right] \\ \text{QUE} & \{\} \\ \text{REL} & \{\} \end{bmatrix}$$

Les templates en ProFIT pour implémenter les contraintes diverses sur les propositions sont présentés dans la Figure 4. Ces templates fournissent un résultat identique à celui de la contrainte générale (1), à l'aide de moyens un peu différent. Dans (1), le SUBJ est spécifié comme list(*PRO*), une liste dont tous les membres ont le type *PRO*, alors que dans la Figure 4 il est spécifié comme `subj!([] or [@'PRO'])`, une disjonction d'une liste vide ou une liste d'un seul membre de type *PRO*. Dans (1) HEAD est spécifié comme MOD '/' *none*, une valeur par défaut de *none*, alors qu'il est spécifié dans la Figure 4 comme `mod!<none` dans tous les sous-types de proposition sauf les propositions relatives (relative clause, `rel_cl`).

Les templates dans la Figure 4 force l'implémentation des contraintes spécifiées par Sag (1997). Quatre sous-types de proposition sont spécifiés: déclarative (`decl_cl`), interrogative (`int_cl`), impérative (`imp_cl`), et relative (`rel_cl`), y compris deux sous-types de relative: *Wh*-relatives (`wh_rel_cl`) et non-*Wh* relatives (`non_wh_rel_cl`). Dans toutes les propositions relatives la valeur MOD de HEAD est toujours de type *noun*, spécifié dans le template par `mod!loc!cat!head!<noun`. Pour les raisons données par Sag (1997), la valeur est encore limitée à NP par le template `@np` dans le cas de `wh_rel_cl`, et à Nbar par le template `@nbar` dans le cas de `non_wh_rel_cl`.

```
clause := <clause &
  syn!(loc!cat!val!subj!([] or
                         [@'PRO']) &
          nonloc!(que![] &
                  rel![] &
                  slash![])).

decl_cl := <decl_cl & @clause &
  syn!loc!(cat!head!mod!<none &
           cont!<proposition).

int_cl := <int_cl & @clause &
  syn!loc!(cat!head!mod!<none &
           cont!<question).

imp_cl := <imp_cl & @clause &
  syn!loc!(cat!head!mod!<none &
           cont!<directive).
```

```
rel_cl := <rel_cl & @clause & syn!loc!(
    cat!head!(inv!- & mc!- &
                mod!loc!cat!head!<noun) &
    cont!<proposition).

wh_rel_cl := <wh_rel_cl & @rel_cl &
  syn!loc!cat!head!mod!@np(_).

non_wh_rel_cl :=
  <non_wh_rel_cl & @rel_cl &
  syn!(loc!cat!head!mod!@nbar(Index) &
          nonloc!slash![]) &
  hd_dtr!syn!nonloc!slash![@np(Index)].
```

Figure 4: Templates pour les contraintes sur les propositions

4. Les constructions de propositions relatives

Les contraintes sur les types syntagmatiques et les types propositionels sont combinées pour spécifier les constructions de propositions relatives. En général, selon Sag (1997), il y aura une interaction entre les contraintes spécifiques (sur les types syntagmatiques et propositionels spécifiques) et les contraintes générales sur les schémas de dominance immédiate et les principes généraux de HPSG. Cette interaction de contraintes fait en sorte que les constructions de propositions relatives bien formées soient admises, et que les constructions mal formées soient exclues. Mais le système ProFIT n'a pas de mécanisme général pour les contraintes dynamiques attachées aux types spécifiques. L'implémentation des contraintes se fait donc par d'autres moyens. La méthode se base sur les constructions syntaxiques: pour chaque construction, il y a une règle de construction. Dans chaque règle, il y a une spécification de type syntagmatique ainsi qu'une spécification de type propositionel, faites par les templates à deux dimensions que nous avons déja rencontrés.

Dans l'analyse de Sag, il y a quatre sous-types distincts de syntagme sujet (head-subject phrase), ou pour être plus exact, il y a quatre sous-types distincts de syntagme sujet fini (finite head-subject phrase), dans la hiérarchie présentée dans la Figure 1. C'est-à-dire: head-subject declarative clause (`hd_subj_decl_cl`), *Wh*-subject interrogative clause (`wh_subj_int_cl`), *Wh*-subject relative clause (`wh_subj_rel_cl`), et bare relative clause (`bare_rel_cl`).

Les règles de construction des propositions déclaratives sujet (`hd_subj_decl_cl`) et des propositions relatives sujet(`wh_subj_rel_cl`) sont présentées dans la Figure 5. Les règles sont dérivées du schéma général de dominance immédiate des syntagmes sujet (head-subject phrases). On voit que les types des syntagmes dominants sont `<hd_subj_decl_cl` et `<wh_subj_rel_cl` respectivement. Les contraintes spécifiques sur les constructions sont imposées par les deux templates de type syntagmatique et de type propositionel. Dans les deux cas, l'expression `@fin_hd_subj_ph` impose toutes les contraintes attachées aux

syntagmes *finite head-subject phrase* par le template `fin_hd_subj_ph` défini dans la Figure 3. Dans le cas de la proposition declarative, l'expression `@decl_cl` impose toutes les contraintes spécifiées par le template `decl_cl` défini dans la Figure 4. Par contre, dans le cas de la proposition *Wh-relative clause*, l'expression `@wh_rel_cl` impose un ensemble différent de contraintes spécifiées par le template `wh_rel_cl` qui est aussi défini dans la Figure 4.

```
phrase(<hd_subj_decl_cl &
    @fin_hd_subj_ph &
    @decl_cl &
    @'SLIP' &
    @'WHIP'(que) & @'WHIP'(rel) &
    syn!loc!cat!head!inv!- &
    hd_dtr!(Head & <phrase) &
    subj_dtr!(Subj & <phrase))
-->
    phrase(Subj),
    phrase(Head).
```

```
phrase(<wh_subj_rel_cl &
    @fin_hd_subj_ph &
    @wh_rel_cl &
    @'SLIP' & @'WHIP'(que) &
    syn!loc!cat!head!mod!@np(Index) &
    hd_dtr!(Head & <phrase &
            syn!nonloc!rel![Index]) &
    subj_dtr!(Subj & <phrase &
              syn!nonloc!rel![Index]))
-->
    phrase(Subj),
    phrase(Head).
```

Figure 5: Proposition déclarative sujet et proposition relative sujet

La règle pour les constructions *bare relative* est présentée à gauche dans la Figure 6. Encore une fois, des contraintes spécifiques sont imposées par les deux templates de type syntagmatique et de type propositionel. Le template `@fin_hd_subj_ph` reste comme dans les cas précédents, mais le template `@non_wh_rel_cl` impose un ensemble différent de contraintes attachées aux propositions *non-Wh-relative*, définies dans la Figure 4.

Les constructions que nous avons examinées jusqu'ici sont toutes des sous-types de syntagme sujet (head-subject phrases), donc elles sont toutes construites avec le template `@fin_hd_subj_ph` pour les contraintes de type syntagmatique. Les différences entre les constructions ont uniquement été spécifiées par des templates différents pour les contraintes des types propositionels. Nous comparons maintenant la règle de construction des propositions de type *bare relative clause* avec la règle de construction des propositions de type *simple infinitival relative clause*, qui est présentée à droite dans la Figure 6. Encore une fois, les contraintes spécifiques sur les constructions sont imposées par les deux tem-

```
phrase(<bare_rel_cl &
    @fin_hd_subj_ph &
    @non_wh_rel_cl &
    @'WHIP'(que) & @'WHIP'(rel) &
    hd_dtr!(Head & <phrase) &
    subj_dtr!(Subj & <phrase))
-->
    phrase(Subj),
    phrase(Head).
```

```
phrase(<simp_inf_rel_cl &
    @hd_comp_ph &
    @non_wh_rel_cl &
    @'WHIP'(que) & @'WHIP'(rel) &
    hd_dtr!(Head & <word &
        syn!loc!cat!(
            head!(<comp & vform!<inf) &
            val!comps![CompSyn])) &
    comp_dtrs![Comp & <phrase &
               syn!CompSyn])
-->
    word(Head),
    phrase(Comp).
```

Figure 6: Constructions de propositions relative *bare* et relative infinitive simple

plates de type syntagmatique et de type propositionel. Mais cette fois, le template de type propositionel `@non_wh_rel_cl` est le même dans les deux constructions. C'est le template de type syntagmatique qui fait la distinction entre les constructions. Pour les propositions de type *simple infinitival relative clause*, le type syntagmatique est le *head-complement phrase*, et les contraintes spécifiques sont imposées par le template `@hd_comp_ph` défini dans la Figure 3.

5. Conclusion

Nous ne décrivrons pas ici chaque type de construction de proposition relative, mais il faut noter que l'implémentation présentée peut traiter toutes les constructions de proposition relative en anglais telles qu'analysées par Sag (1997). Nous n'abordons pas l'implémentation des règles lexicales pour la représentation des phénomènes d'extraction sans catégorie vide, ni l'amalgamation lexicalisée des traits non-locaux, ni le traitement des complémenteurs. Ces idées de Sag (1997) sont aussi inclues dans l'implémentation.

Nous n'abordons pas les liens entre les idées présentées par Sag (1997) dans le cadre de HPSG, et les idées de la grammaire systémique SFG, dont l'implémentation est examinée par Mellish (1988) et Erbach (1994). Les idées présentées ici sont la représentation des contraintes au moyen des templates, l'héritage multiple en ProFIT, et la spécification des constructions syntaxiques par la combinaison des templates à deux dimensions orthogonales.

Références

Erbach G. (1994). Multi-dimensional inheritance. In H. Trost, Ed., *Proceedings of KONVENS '94*, p. 102–111, Vienna: Springer.

Erbach G. (1995). ProFIT: Prolog with Features, Inheritance, and Templates. In *Seventh Conference of the European Chapter of the Association for Computational Linguistics*, p. 180–187, Dublin.

Mellish C. S. (1988). Implementing systemic classification by unification. *Computational Linguistics*, **14(1)**, 40–51.

Sag I. A. (1997). English relative clause constructions. *Journal of Linguistics*, **33**(2), 431–484.

Wilcock G. & Matsumoto Y. (1996). Implementing HPSG with modular tools for fast compiling and parsing. In *Actes de TALN–96*, p. 65–66, Marseille.