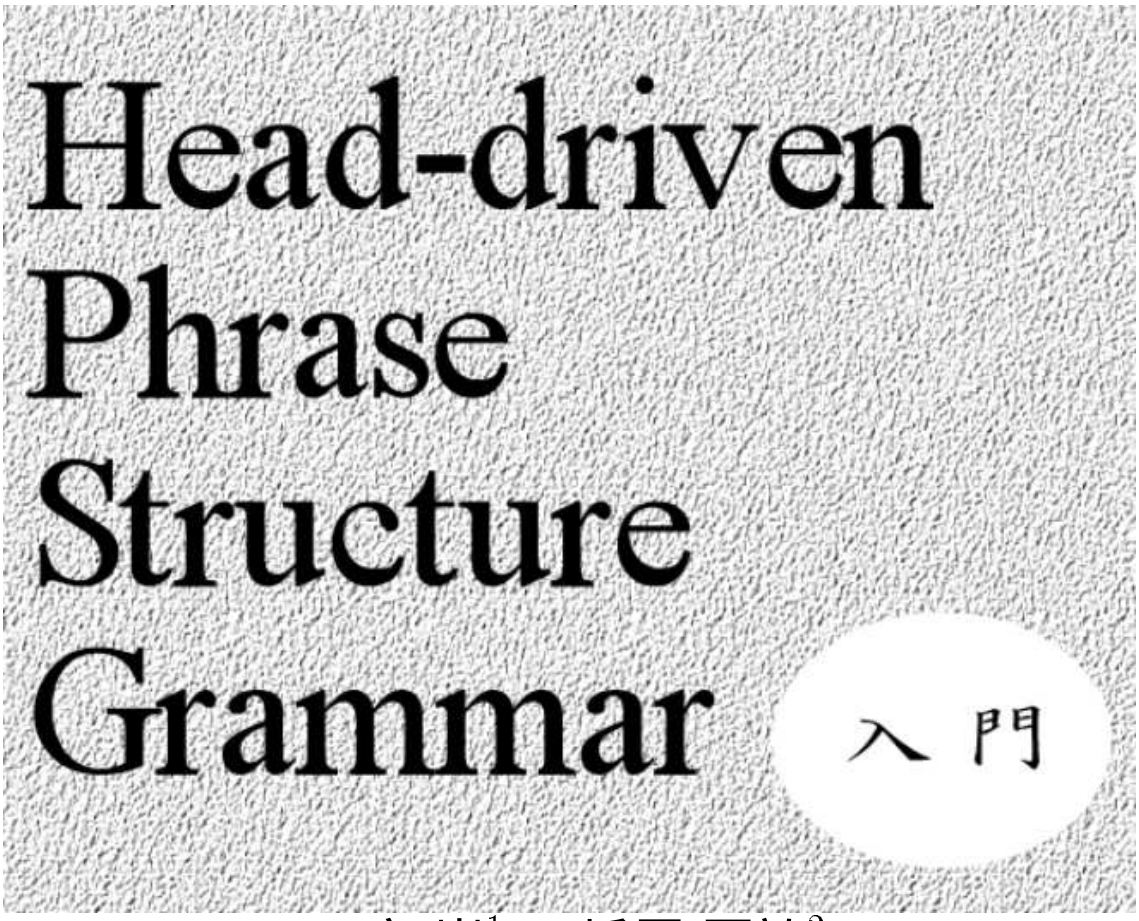


二宮 崇[1] & 坂尾 要祐[2]

[1]ninomi@is.s.u-tokyo.ac.jp
[2]sakao@is.s.u-tokyo.ac.jp

1 はじめに

このテーマではHPSGパーザー、およびHPSG文法をLiLFeSという素性構造を扱える論理型プログラミング言語の上でつくってもらいます。まず、HPSGとは何かということですが、HPSGとは、

- 主辞が中心的な役割を果たす文法枠組
- 辞書の情報を増やすことにより、文法規則をできるかぎり減らす辞書指向
- 素性構造、制約に基づく単一化文法の枠組

です。

HPSGはHead-Driven Phrase Structure Grammarの略であり、日本語にすると、主辞駆動句構造文法と呼ばれます。それでその主辞とはなんぞや？ということですが、たとえば、「美しい花」という名詞句があれば、この主辞は名詞である「花」になります。「彼は美しい花を見た」だと「見た」が主辞になります。この主辞を正確に定義しようとすると非常に難しいのですが、つまり、与えられた句のうち、もっとも重要そうな要素、修飾先がない要素のことを主辞と呼べば差し支えないでしょう。もしくは、関数と引数の関係でたとえるならば、主辞が関数で補語が引数だと思えばいいでしょう。例えば、さきほどの例なら「見た(彼、花(美しい))」となります。それでHPSGというのは概念的には、主辞のまわりに、その主辞を修飾する句がくっついていくことにより、文全体が完成する文法枠組と考えればよいでしょう。

続いてHPSGに限らないことですが、HPSGは制約に基づく単一化文法である、ということです。制約に基づくというのはある意味CFGなどの文法規則により文を導出、生成するという生成文法の概念とは反対の概念になります。つまり、文字の無限集合のうち、こういう形のものは可能、こういう形のものはだめ、と規定することによりある言語に属する文法を記述していきます。つまり、ある言語 G の文字の集合を $L(G)$、導出された記号を S とすると、生成というのは、

$$L(G) = S_1 \vee S_2 \vee S_3 \vee .....$$

というORで記述され、制約に基づくというのは、制約を C としたとき、

$$L(G) = C_1 \wedge C_2 \wedge C_3 \wedge .....$$

とANDで記述されたものと考えればわかりやすいでしょう。どちらが果たして言語を記述する枠組みとして適当なのかは意見がわかれるところですが、HPSGでは後者のアプローチを採用しています。

それから辞書の情報を増やすということですが、一般にCFGだと辞書というのは単語の音韻とそれに対応する非終端記号だけしか情報がなく、動詞がどういった補語をとるのかということは文法規則に書かれます。例えば、

文 $\rightarrow$ 名詞句 動詞
名詞句 $\rightarrow$ 名詞 助詞
名詞 $\rightarrow$ 光石
助詞 $\rightarrow$ が
動詞 $\rightarrow$ 寝る

といったCFGで「光石が寝る」といった文が生成されるわけですが、このように書いた場合には、もし本気で英語文法とか日本語文法を書こうとしたら文法規則を山のように(数百数千)書いていかなくてはいけませんし、些細な方針の変更で ほとんどの文法規則を書き換えなくてはいけなくなります。ですが、HPSGではこういうCFGでの文法規則に対応する構造に関する制約、意味に関する

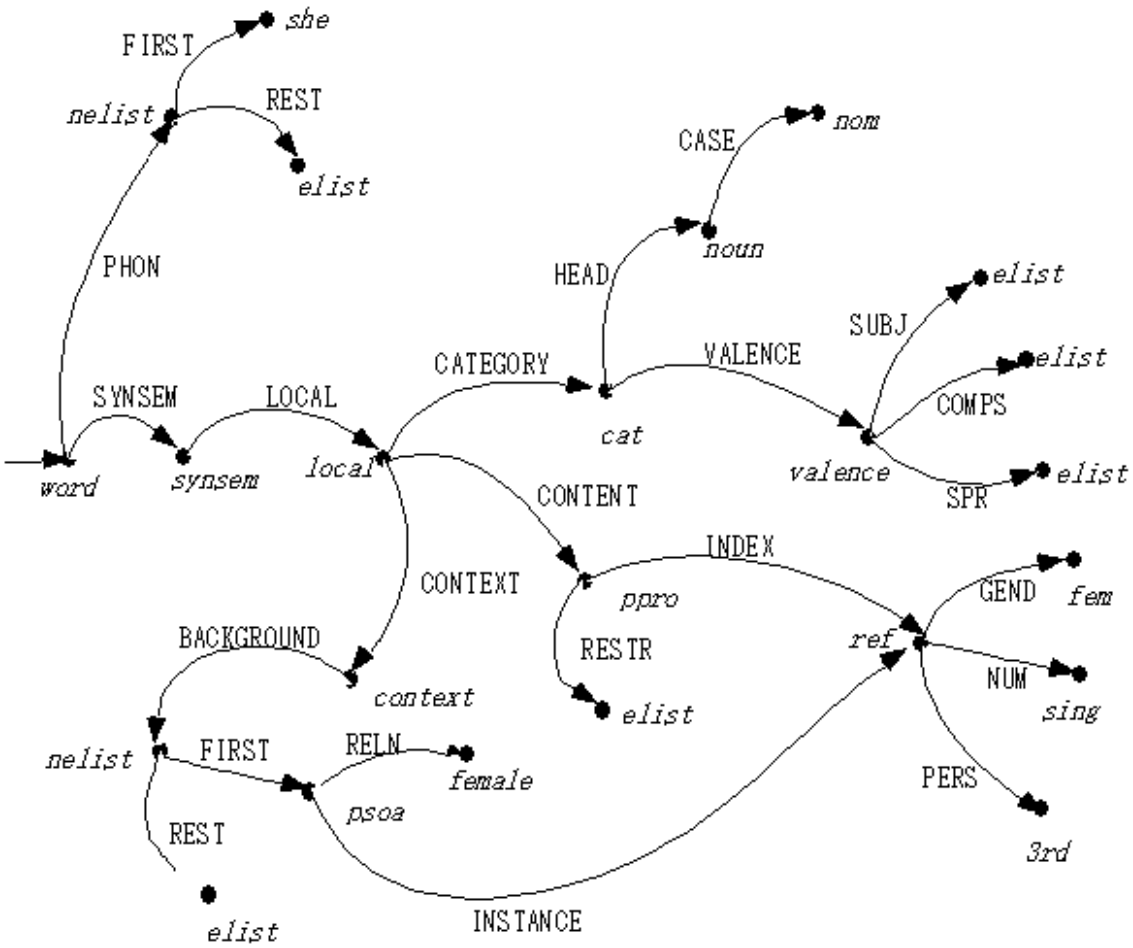
制約、一致（三人称単数の一致とか）に関する制約といった感じで制約をモジュール化し、文法の変更は、モジュールごとに行えるようになっています。また、単語によって違う振る舞いをする現象は辞書の中に書けるため、より抽象化されたモジュールの記述が可能になります。したがって辞書項目の情報を十分に記述できるような柔軟なフレームワークが必要になるわけですが、HPSG においては、その基本的な構造に素性構造という構造が用いられており、各単語、句、構文木、文法規則、など全てがそれを用いて記述されます。して、その素性構造とはなんなのか、ということですが、詳しくは次章で述べるとして、おおざっぱに言えば、各単語、句、構文木の状態、情報を表す有向グラフと考えていただければいいでしょう。例えば、矢印に PERSON(人称) と書かれた先のノードには first(一人称) と書かれていたり、GENDER(性) と書かれた矢印の先のノードには male(男性) と書かれていたりします。HPSG においては、素性構造が単語、句、文法規則、構文木、全てを表すベースとなり、素性構造間に制約を設けることにより、構文解析、文生成を行い、また、その柔軟な記述方式により、統語論、意味論、語用論全てをまとめて取り扱えることが期待されています。ここでの制約とはおおざっぱには prolog における Horn 節みたいなもので、prolog における term に素性構造を置くことができ、この素性構造の単一化を行うことによって制約がかけられると思っていただければ十分でしょう。詳しくは次章以降で説明いたします。

2　素性構造

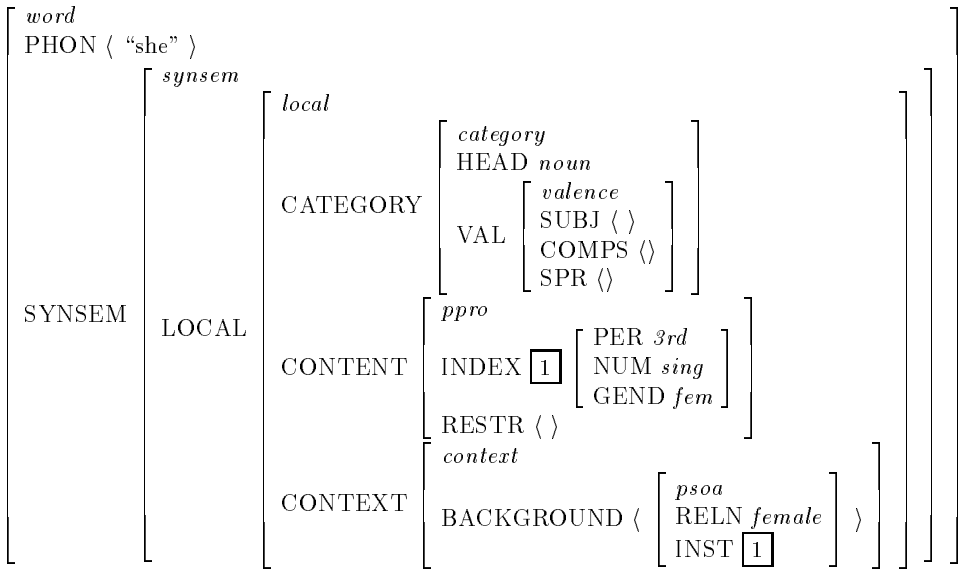
HPSG とは一章であげたような文法枠組になるわけですが、まず、その中心的役割を果たす素性構造について説明いたします。

2.1　素性構造

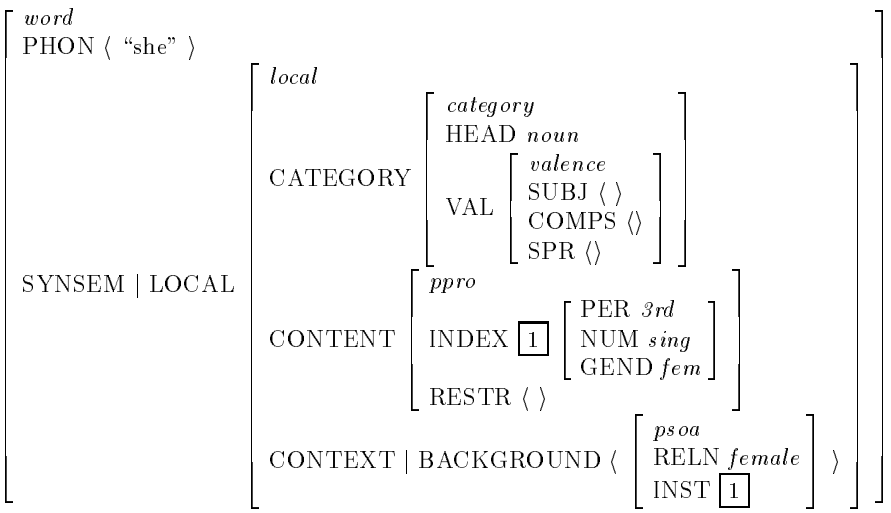
素性構造は単語、句、構文木を表現する根付き有向グラフで、例えば she という単語の素性構造は次のように表されます。



一番左端の矢印が root をあらわしています。ここでは各ノードのラベルをイタリックで記述し、矢印のラベルを大文字であらわしています。例えば、root のノードは word となっていて PHON というラベルの矢印をたどると nelist というノードにたどりつきます。このような各ノードのラベルは HPSG においては sort と呼ばれ、矢印のラベルは素性と呼ばれます。HPSG においては sort には階層があって (2.2 で後述)、各 sort から生えうる矢印 (素性) は sort によって決まっています (well-typed)。また、矢印が生えていない sort のことを atom と呼びます。例えば右下のほうの GEND の矢印がさしている fem は atom です。しかしながら、HPSG を語る上において上記のようなグラフをいちいち書いていたのでは不便でしょうがないので、一般には以下のような AVM(attribute-value matrix) で表記されます。



これは上記の有向グラフとまったく同じものを表しています。つまり、各ボックスが素性構造を表しており、その素性構造の sort はボックスの左上に記述され、各 sort における素性がさらに内側にある素性構造へのパスを示しています。また、グラフにおいて、サイクルになっている場合を含め、矢印の先が共有されている場合 (上記の場合において、SYNSEM, LOCAL, CONTENT, INDEX とたどった場合と SYNSEM, LOCAL, CONTEXT, BACKGROUND, FIRST, INST とたどった先のノードは同一) のことを構造共有と呼び、AVM においてはダグ (1 とか 2 であらわされます。ここでは 1 が構造共有をしめしている) で表現されます。また、省略表記として、リストを <> で表記します。Lisp でいえば、nil がここでは elist に対応し、cons が nelist, car が FIRST, cdr が REST に対応すると思ってください。また、もう一つ重要な省略表記として、値が特に入っていないような素性は (ボトムと呼ばれる特殊な型が入っている) その素性を省略して書きます。また、その結果、あるボックスの素性が一つしかない場合、ボックスを書かずに素性と素性の間に縦棒が一本だけ引くという省略をしたりします。例えば上記の素性構造は一般には次のように省略して書かれます。また、素性構造の素性に DTRS という素性があり、ここに上記のような構文木のノードをあらわすような素性構造を格納することにより、構文木も表現できます。具体例はスキーマの章 (4 章) でみることができます。

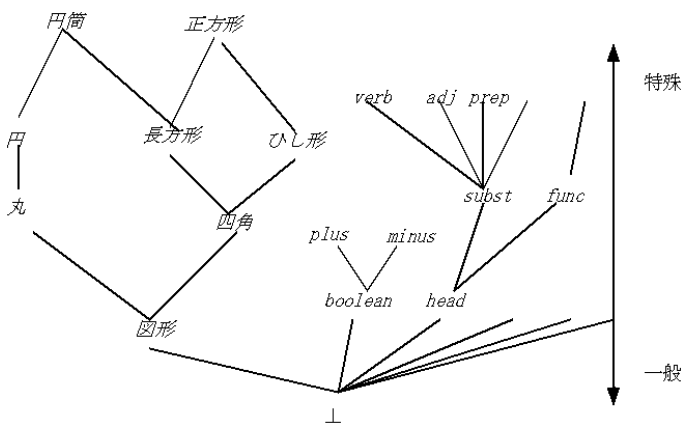


2.2 sort hierarchy と単一化 (unification)

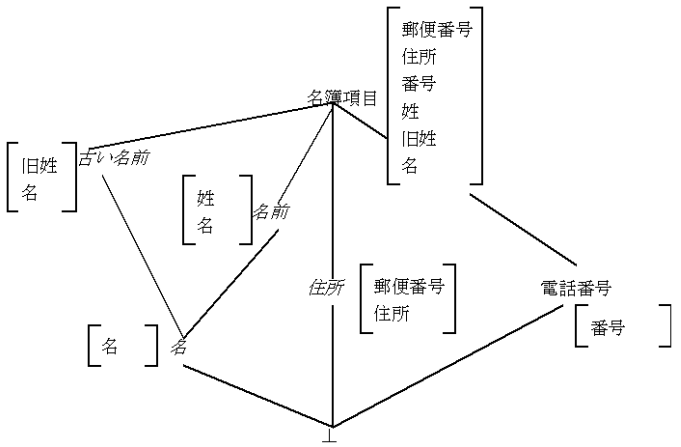
前節において sort には階層があると述べましたが、そのことについて述べ、および、それと大いに関係してくる単一化について説明いたします。

2.2.1 sort hierarchy

sort には階層 (順序関係) があります。まず天下り的ですが、次のような階層構造があると思ってください。



各階層におけるノードが sort を表しています。下にいくほど一般的な sort であり、上にいくほど特殊な sort だと思ってください。sort には以上のような関係があり、各 sort が持ち得る素性は下の階層の sort の素性 + なんらかの素性となります。つまり C++でのクラスでの継承と同じように素性を継承すると思えばいいでしょう。例えば、次のような

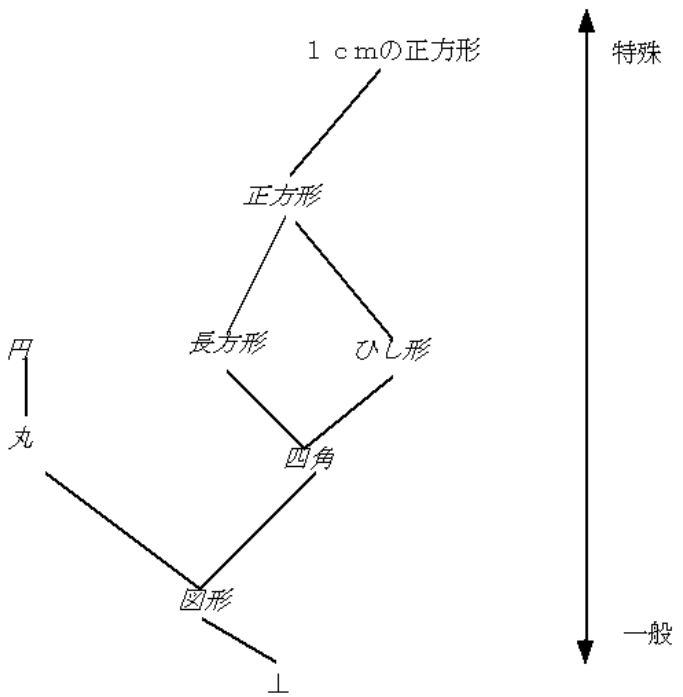


階層があった時 (四角のボックスは素性構造を意味していて、ボックスの中の文字列は各行が素性をあらわしているつもり)、古い名前、名前、名、住所、電話番号、名簿項目という sort があり、各 sort には sort に応じた素性によって指定されるフィールドが用意されます。例えば名前には、「姓」という素性、および、「名」という素性があり、住所には「郵便番号」という素性と「住所」という素性が用意されています。ここにおいて sort 階層ができているわけですが、例えば名前は名を継承して、「名」という素性をもちつつ、新たな素性、「姓」を追加しています。同じく名を継承して古い

名前という sort がありますが、やはり、「名」という素性を継承して、新たに「旧姓」という素性を用意しています。続いて、多重継承する場合なのですが、この場合では名簿項目という sort が古い名前、名前、住所、電話番号を継承しています。すなわち親の全ての素性をマージしたものを素性としてもつような sort になります。注意するのは、「名」という素性はマージされて一つになっていることでしょう。

2.2.2 atom の単一化

素性構造の単一化の前に atom の単一化について理解しておきましょう。次のような sort 階層があったとしましょう。ここでは各 sort は素性をもたない、つまり、atom である sort からなる階層だけを考えてみましょう。



単一化の記号は ⊔ で表記されます。例えば、上の例では、 四角 ⊔ ひし形 というように記述され、ここでは 四角 ⊔ ひし形 = ひし形 となります。例えば prolog における term の単一化の場合を考えてみましょう。

４つの辺をもつ (四角) .
４つの辺をもつ (ひし形) .

というようなファクトがあったとき、

?- ４つの辺をもつ (X) .

というクエリーに対しては

X = 四角;
X = ひし形

という答えが返ってくるのですが、ここで、

?- ４つの辺をもつ (四角) .

とした場合は、 ４つの辺をもつ (四角) . というファクトの term と単一化され、かつ単一化に成功

するため、yes が返ってきます。つまり、四角 ⊔ 四角 = 四角 で成功するのですが、ここでは sort に階層があるため、四角と四角で単一化しなくとも単一化に成功することができます。 どういうことかというと、sort における単一化というのは次のように定義されるからです。

- X ⊔ Y という単一化を行ったとき、X の子孫と Y の子孫が合流したら、その合流した地点における sort を単一化の結果として返す (つまり、X の特殊なケースであり、かつ Y の特殊なケースであるようなものを返す)
- 合流する地点がみつからなかったら失敗
- 一度合流地点が見つかったら、それより上側 (子孫側) の sort をみつけにはいかない (least upper bound)。

例えば、

?- 4 つの辺をもつ (図形) .

とクエリーをだした場合において、prolog ならば 単一化に失敗して no. という答えを返してくるのですが、ここでは yes が返ってきます。というのは、四角は図形の子孫であるから、四角 ⊔ 図形 = 四角 となり、単一化に成功します (一方がもう一方の子孫である場合にはより特殊な方の sort が結果として返ってくる)。ここで重要なのはたしかに 4 つの辺をもつ (図形) で成功はするのですが、、、その term の sort は四角で単一化されているということです。例えば、、、

4 つの辺をもつ (X,X) :- X = 四角.

?- 4 つの辺をもつ (図形,Y) .

とした場合には、Y = 四角が返ってきて、Y = 図形 という答えは返ってこない、ということです。他、 長方形 ⊔ ひし形 = 正方形、 円 ⊔ ひし形 は失敗、、、となります。注意してほしいのは 長方形 ⊔ ひし形 = 1cm の正方形 にはならないということです (least upper bound でないから)。また、一番下にある⊥ (ボトム) という sort は常に一番したにあり、全ての sort はここから派生されます。つまり、どんな sort とも単一化可能ということです。

2.2.3 素性構造の単一化

素性構造の単一化についてですが、素性をいくつかもつような sort でも、やはり atom と同じように単一化したときには sort 階層にしたがって二つの sort の子孫をみていって、合流するところの sort が単一化結果となります。そしてその時の素性と素性の値はどうなるかということですが、上記の例で、

$$\begin{bmatrix} \text{住所} \\ \text{郵便住所 } 790 \\ \text{住所 どこそこ} \end{bmatrix} \sqcup \begin{bmatrix} \text{名} \\ \text{名 崇} \\ \end{bmatrix}$$

の結果は

$$\begin{bmatrix} \text{名簿項目} \\ \text{郵便番号 } 790 \\ \text{住所 どこそこ} \\ \text{名 崇} \end{bmatrix}$$

になります。また、ここでは 790、崇、どこそこというのは sort の一つであるということに注意してください。

$$\left[\begin{array}{l} \text{名前} \\ \text{姓 光石} \\ \text{名 豊} \end{array}\right] \sqcup \left[\begin{array}{l} \text{名} \\ \text{名 崇} \end{array}\right]$$

だと名の 豊 と 崇 は単一化できないためこの素性構造の単一化は失敗となります。つまり、各素性の値に対して再帰的に単一化することによって素性構造の単一化が行われます。また、構造共有されている場合には注意が必要になります。次のような例をみてみましょう。構造共有がない場合は、

$$\left[\begin{array}{l} \mathrm{A} \mid \mathrm{B}\ a \\ \mathrm{C} \mid \mathrm{D} \mid \mathrm{B}\ a \end{array}\right] \sqcup \left[\ \mathrm{C} \mid \mathrm{D} \mid \mathrm{E}\ b\ \right] = \left[\begin{array}{l} \mathrm{A} \mid \mathrm{B}\ a \\ \mathrm{C} \mid \mathrm{D} \left[\begin{array}{l} \mathrm{B}\ a \\ \mathrm{E}\ b \end{array}\right] \end{array}\right]$$

となりますが、これが構造共有をしている場合

$$\left[\begin{array}{l} \mathrm{A}\ \boxed{1} \left[\ \mathrm{B}\ a\ \right] \\ \mathrm{C} \mid \mathrm{D}\ \boxed{1} \end{array}\right] \sqcup \left[\ \mathrm{C} \mid \mathrm{D} \mid \mathrm{E}\ b\ \right] = \left[\begin{array}{l} \mathrm{A}\ \boxed{1} \left[\begin{array}{l} \mathrm{B}\ a \\ \mathrm{E}\ b \end{array}\right] \\ \mathrm{C} \mid \mathrm{D}\ \boxed{1} \end{array}\right]$$

となります。つまり、構造共有しているところでも単一化が行われ sort 階層の中で合流できなかった sort を含むような素性構造の単一化は失敗となります。また、構造共有そのものも単一化されるということに注意してください。例えば、

$$\left[\begin{array}{l} \mathrm{A}\ \boxed{1}\ \bot \\ \mathrm{B}\ \boxed{1} \\ \mathrm{C}\ \bot \end{array}\right] \sqcup \left[\begin{array}{l} \mathrm{A}\ \bot \\ \mathrm{B}\ \boxed{2}\ \bot \\ \mathrm{C}\ \boxed{2} \end{array}\right]$$

はどうなるでしょう？次のようになります。

$$\left[\begin{array}{l} \mathrm{A}\ \boxed{1}\ \bot \\ \mathrm{B}\ \boxed{1} \\ \mathrm{C}\ \boxed{1} \end{array}\right]$$

3 選択素性

ここからようやく HPSG の仕組みそのものの話になるわけですが、理解をスムーズにしてもらうために、覚えておけば便利なターミノロジー、および素性名を紹介しておきます。

[補語 (**complement**)] 英語でならった SVOC の C のことじゃなくて、主辞がとることができる句、単語のこと。つまり、SVOC の表記なら、主辞は V になって、残りの S,O,C が補語ということになる。

[下位範疇化 (**subcategorization**)] 主辞が補語と結合して、より大きな句を作ること。例えば、動詞は、目的語をとって動詞句になるようなこと。

[主格 (**nominative**)] ひらたくいえば SVOC の S である。

[対格 (**accusative**)] ひらたくいえば SVOC の O である。

[指定部 (**specifier**)] countable な単数の名詞が手前にもつべき単語、句。例えば、dog は dog の前に the や、a、my、their といった単語がこないと非文になる。そういう the や a のことを指定部という。

[**N**] 名詞 (noun) のこと

[**V**] 動詞 (verb) のこと

[**A**] 形容詞 (adjective) のこと

[**P**] 前置詞 (preposition) のこと

[**S**] 文 (sentence) のこと

[**NP**] 名詞句 (noun phrase) のこと

[**VP**] 動詞句 (verb phrase) のこと

[**AP**] 形容詞句 (adjective phrase) のこと

[**PP**] 前置詞句 (prepositional phrase) のこと

続いて素性名についてですが、

[**PHON**] 音韻 (phonology) をあらわす素性。文にあらわれる単語そのものと思えばよい。

[**SYNSEM**] syntax and semantics のこと。HPSG は昔、SYN という素性と SEM という素性に別れていましたが、最近はこれがくっついて SYNSEM という一つの素性になりました。

[**NONLOCAL**] 文中において遠い関係をあらわす情報を格納する。例えば、英語において book which I read という名詞句があったとき、この book というのは I read の目的語になるわけですが、この book と I read の目的語が入るべき位置とはちょっと遠い関係にある。こういう関係の情報を格納する素性。

[**LOCAL**] NONLOCAL に対して、比較的、局所的な情報を格納しているが、基本的にはここには単語、句に関する全ての情報が書かれていて、NONLOCAL を通して単語の情報が遠いところにまで輸出される、と考えるとよいだろう。

[**CATEGORY**] CAT とよく省略して書かれる。これは日本語では範疇と呼ばれている。この素性に属する素性構造によって、性、数、格、人称、時制、相、法、態、定不定、可算不可算などの情報が記述される。CFG でいうところの非終端記号を表している素性構造と思えばよい。

[**HEAD**] 主辞に関する情報が入っている素性。重要な役割を果たす素性です。

[**DTRS**] daughters の略。言語学においては何故かノードの親子関係を「parent, child」と呼ばず、「mother, daughter」と呼ぶ。ここに属する値としては HEAD_DTR とか SUBJ_DTR といった素性をもつ素性構造があり、ここに子供の素性構造を格納することにより、構文木を記述する

つづいて構文木を作るのに関わる重要な素性について説明します。基本となるのは、

- 主辞がどの補語を下位範疇化するかということ (例えば、gives という単語は目的語として名詞を二つとり、主語として名詞を一つとる)
- 補語がどういう主辞を修飾できるかということ (例えば、pretty という単語は名詞を修飾するが、形容詞、動詞を修飾したりはしない)

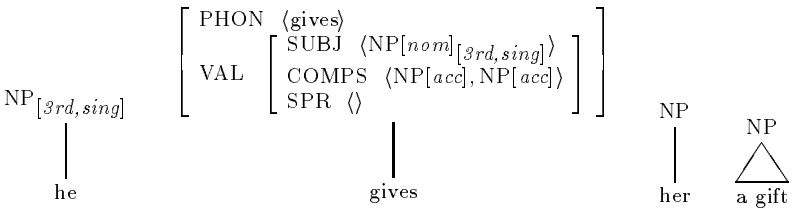
ということが書かれている素性なのですが、こういう素性のことを選択素性 (selecting feature) と呼びます。それらのうち、下位範疇化に関する素性のことを HPSG では valence feature と呼び、それらは SUBJ, COMPS, SPR の 3 つです。これらは、SYNSEM|LOCAL|CAT|VAL の下にあります。つまり、この VAL には valence という sort が入るのですが、この valence が持ち得る素性は SUBJ,COMPS, SPR(主語、補語、指定詞) になっていて、各素性はリストを値としてとることができます。リストの要素は主辞によって下位範疇化されうる素性構造であり、主辞が下位範疇化した要素を抜いたリストが親の valence feature のリストになります。例えば、gives という単語は次のような選択素性、および値をもっています。

$$\begin{bmatrix} \text{PHON} \langle \text{ gives } \rangle \\ \text{SYNSEM} \mid \ldots \mid \text{VAL} \begin{bmatrix} \text{SUBJ} \langle \text{ NP}[nom]_{[3rd,sing]} \rangle \\ \text{COMPS} \langle \text{ NP}[acc], \text{NP}[acc] \rangle \\ \text{SPR} \langle \, \rangle \end{bmatrix} \end{bmatrix}$$

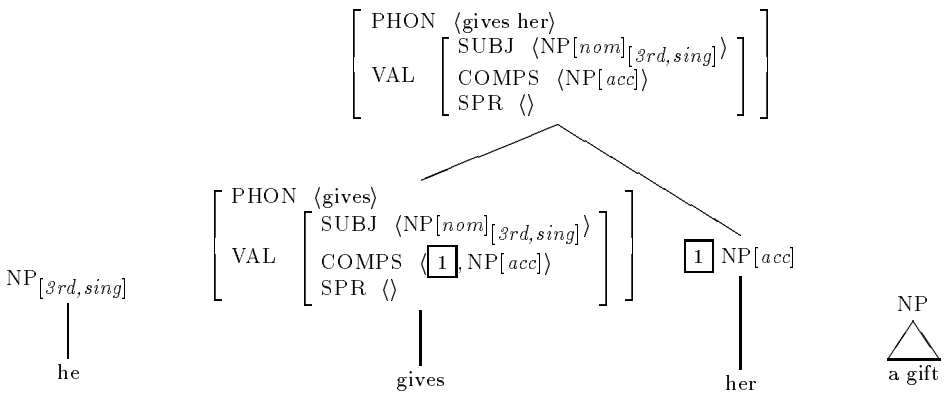
ここで、NP[nom][3rd,sing] とは次の略記をあらわしています。。ちなみに nom というのは主格 (nominative) を表していて、acc というのは対格 (accusative)、sing というのは単数 (singular) を表しています。また、単数に対して、複数には plu(plural) という sort があります。

$$\begin{bmatrix} \text{SYNSEM} \mid \text{LOCAL} \begin{bmatrix} \text{CAT} \mid \text{HEAD} \begin{bmatrix} noun \\ \text{CASE } nom \end{bmatrix} \\ \text{CONTENT} \mid \text{INDEX} \begin{bmatrix} index \\ \text{PER } 3rd \\ \text{NUM } sing \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

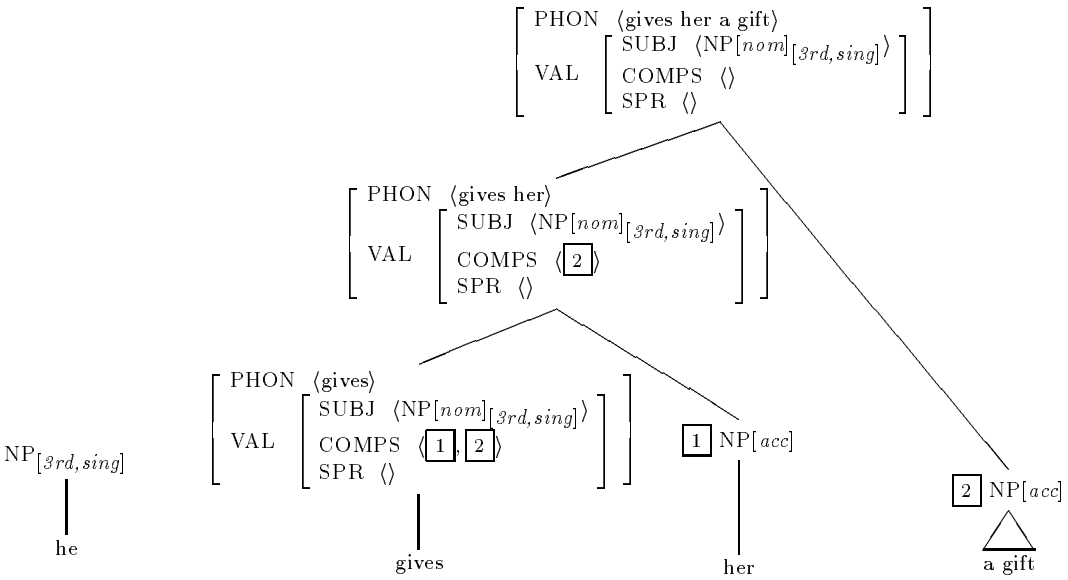
つまり、gives は主語として NP[nom][3rd,sing] をとり、目的語として、NP[acc] と NP[acc] の二つをとる、ということを意味しています。すなわち、パーズするときには、この SUBJ,COMPS の要素と補語を単一化して、単一化に成功するならば、この gives とその補語を子供として新たに親をつくるということをすればよい、ということになります。この gives を用いて、 he gives her a gift. という文をボトムアップにパーズしたら、この選択素性の値がどうなるかみてみましょう。



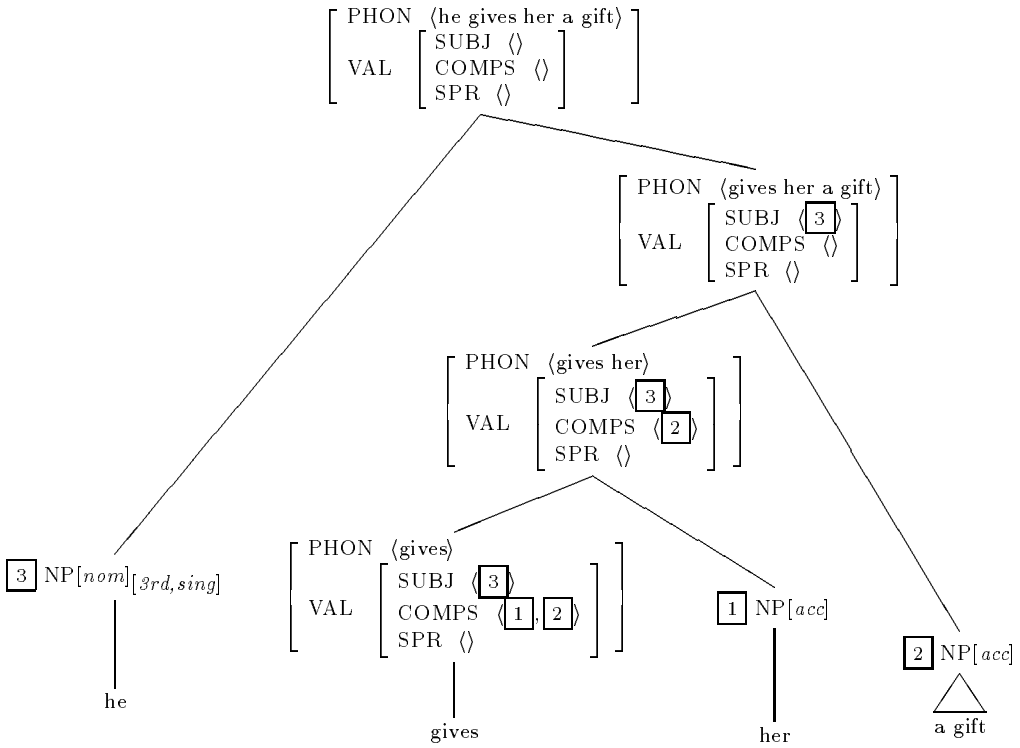
a gift の部分は主辞 gift が補語である a をとって、a gift という名詞句になったとすると、上記のように単語が並びます。



つづいて、主辞である gives が her をとってより大きな句を作っていくのですが、まず、gives が her をとれるかというチェックは COMPS の第一要素と her に対応する素性構造が単一化可能かということによってチェックされ、実際、単一化可能なら単一化してしまいます。 COMPS の要素に記されている構造共有のタグがそれをあらわしています。単一化されることによって、her という単語は accusative であるという情報が her の上に書かれてある NP に付与されていることがわかります。



つづいて、a gift を主辞である gives her がとって、さらに大きな句を作ります。



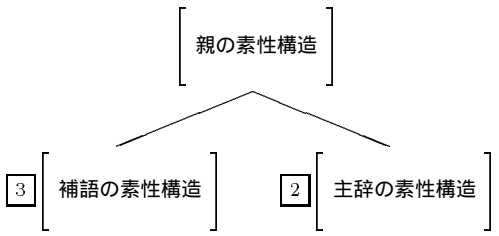
最後に主語をとって文全体をなす構文木が完成します。

このようにしてパーズが行われていくのですが、COMPS や SUBJ の要素を他の素性構造と単一化させるということや下位範疇化された要素をリストから抜いて親に伝えるということはどうやってやるのでしょう。パーザーを書く人がどこそこを単一化してこういう親を作るというプログラムを書くのでしょうか？

HPSG ではこういう構文木を作るときに、どことどこを単一化させて親を作るかということを文法の一つとしてやはり素性構造で表現します。その素性構造のことを HPSG ではスキーマと呼んでいます。

4　スキーマ

スキーマは子供同士の単一化の位置を指定し、親の素性構造の選択素性の形を決定する素性構造です。CFG においては、書き換え規則にあたります。つまり、



という形をしていて、親と子供の関係が記述されます。具体的には、
親の素性構造が以下のようなスキーマになっていて、DTRS に補語と主辞の素性構造が記述され、

$$\left[\begin{array}{l} \text{SYNSEM} \mid \text{LOCAL} \mid \text{CAT} \mid \text{VAL} \left[\begin{array}{l}\text{SUBJ } \langle\rangle \\ \text{COMPS } \langle\rangle \\ \text{SPR } \boxed{1}\end{array}\right] \\ \text{DTRS } \boxed{2} \left[\begin{array}{l}\text{HEAD_DTR } \boxed{2} \left[\text{SYNSEM} \mid \text{LOCAL} \mid \text{CAY} \mid \text{VAL} \left[\begin{array}{l}\text{SUBJ } \langle\, \boxed{3} \,\rangle \\ \text{COMPS } \langle\rangle \\ \text{SPR } \boxed{1}\end{array}\right]\right] \\ \text{SUBJ_DTR } \boxed{3}\end{array}\right] \end{array}\right]$$

というような形になります。これは Head-Subject Schema と呼ばれるスキーマで、英語において、動詞句 (主辞句) が主語を下位範疇化して親になるというスキーマです。ここで重要なのは主辞句 ($\boxed{2}$) の VAL|SUBJ のリストの要素は一つしかない、ということ、および親の SUBJ の値は空のリストになるということです。また、COMPS が空であるというのも重要なのですが、これは文法を書いているうちにわかってくるでしょう。ボトムアップにパーズすると考えるならば、親をつくるために、ある左側の子供と右側の子供ができていたとして、

- $\boxed{3}$ と左側の子供を単一化
- $\boxed{2}$と右側の子供を単一化
- そうすると、親の素性構造 (スキーマの SYNSEM—LOCAL... 部分) が得られる

子供の情報が親に何も伝わっていないと思うかもしれませんが、実際そのとおりで、ここでは「このスキーマを適用する条件を満たす子供の選択」をしているということ、「親の選択素性の形を決定している」ということだけが重要であって、他の情報は次章以降で説明されるプリンシプルによって伝播されていきます。実際に he runs という文をスキーマだけを用いてパーズしてみましょう。まず、以下のようにふたつの単語 ($\boxed{5}$ と $\boxed{6}$) が与えられます。

$$
\boxed{5}\begin{bmatrix}
word \\
\text{PHONE } \langle he \rangle \\
\text{SYNSEM} \mid \text{LOCAL} \begin{bmatrix}
local \\
\text{CATEGORY} \begin{bmatrix} category \\ \text{HEAD } noun \\ \text{VALENCE} \begin{bmatrix} valence \\ \text{SUBJ } \langle\rangle \\ \text{COMPS } \langle\rangle \\ \text{SPR } \langle\rangle \end{bmatrix} \end{bmatrix} \\
\text{CONTENT} \begin{bmatrix} \text{INDEX } \boxed{1} \begin{bmatrix} ppron \\ \text{PER } 3rd \\ \text{NUM } sing \\ \text{GEND } mail \end{bmatrix} \\ \text{RESTR } \langle\rangle \end{bmatrix} \\
\text{CONTEXT} \mid \text{BACKGROUND } \langle \begin{bmatrix} posa \\ \text{RELN } male \\ \text{INST } \boxed{1} \end{bmatrix} \rangle
\end{bmatrix}
\end{bmatrix}
$$

$$
\boxed{6}\begin{bmatrix}
\text{PHONE } \langle runs \rangle \\
\text{SYNSEM} \mid ... \mid \text{VAL} \begin{bmatrix} \text{SUBJ } \langle \text{NP}[nom]_{[3rd,\,sing]} \rangle \\ \text{COMPS } \langle\rangle \\ \text{SPR } \langle\rangle \end{bmatrix}
\end{bmatrix}
$$

上の単語と下の単語を DTRS の下に格納するような素性構造とスキーマを単一化させてみましょう。

$$
\begin{bmatrix} \text{DTRS} \begin{bmatrix} \text{SUBJ_DTR } \boxed{5} \\ \text{HEAD_DTR } \boxed{6} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} HEAD\ SUBJECT\ SCHEMA \end{bmatrix}
$$

すると次のような結果になります。見てわかるようにこれが親の素性構造になります。

$$
\begin{bmatrix}
\text{SYNSEM} \mid \text{LOCAL} \mid \text{CAT} \mid \text{VAL} \begin{bmatrix} \text{SUBJ } \langle\rangle \\ \text{COMPS } \langle\rangle \\ \text{SPR } \boxed{1} \end{bmatrix} \\
\text{DTRS} \begin{bmatrix}
\text{SUBJ_DTR } \boxed{2} \begin{bmatrix}
\text{PHON } \langle \text{ he } \rangle \\
\text{SYNSEM} \mid \text{LOCAL} \begin{bmatrix}
\text{CAT} \begin{bmatrix} \text{HEAD } \boxed{\boxed{\begin{bmatrix} noun \\ \text{CASE } nom \end{bmatrix}}} \\ \text{VALENCE} \begin{bmatrix} \text{SUBJ } \langle\rangle \\ \text{COMPS } \langle\rangle \\ \text{SPR } \langle\rangle \end{bmatrix} \end{bmatrix} \\
\text{CONTENT} \begin{bmatrix} \text{INDEX } \boxed{\boxed{4} \begin{bmatrix} \text{PER } 3rd \\ \text{NUM } sing \\ \text{GEND } male \end{bmatrix}} \\ \text{REST } \langle\rangle \end{bmatrix} \\
\text{CONTEXT} \mid \text{BACKGROUND } \langle \begin{bmatrix} \text{RELN } male \\ \text{INST } \boxed{4} \end{bmatrix}
\end{bmatrix}
\end{bmatrix} \\
\text{HEAD_DTR} \begin{bmatrix}
\text{PHON } \langle \text{ runs } \rangle \\
\text{SYNSEM} \mid \text{LOCAL} \mid \text{CAT} \mid \text{VAL} \begin{bmatrix} \text{SUBJ } \langle \boxed{2} \rangle \\ \text{COMPS } \langle\rangle \\ \text{SPR } \boxed{1} \end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

ここで、2の HEAD の四角い枠で囲んだ部分に注目してみてください。CASE nom という情報が増えています (he という単語には特に CASE の値は定まっていなかった。つまり、ボトムがはいっていた。)。これは主辞である runs の SUBJ に書かれていた情報なのですが、単一化することにより、お互いの情報が一体化して、同一の素性構造をさすようになります。

また、その下側の手書き枠の部分ですが、これは he にも runs にも書かれているため、お互い単一化可能かどうかのチェックになります。たとえば、he のかわりに they だったりしたら、NUM の値は plu (複数) になるので、単一化できません。つまり、they runs というのは非文である (人間が判断して文法的にみえない文のことを非文という) ということがチェックされているわけです。このように主語、動詞句でくっつく場合のスキーマの他、動詞と補語がくっついて動詞句をつくるスキーマとか、関係詞とくっつくスキーマとかいろいろあります。

ちなみに LiLFeS で書くなら、次のような感じになるでしょう。

```
head_subject_schema($LEFT,$RIGHT,$HEAD,$SUBJ,$MOTHER) :-
        $MOTHER = (SYNSEM\LOCAL\CAT\VAL\(SUBJ\ [],
                                         COMPS\ [],
                                         SPR\ $Z),
                   DTRS\(HEAD_DTR\ $HEAD,
               SUBJ_DTR\ $SUBJ)),
        $HEAD = (SYNSEM\LOCAL\CAT\VAL\(SUBJ\ [$SUBJ],
                                       COMPS\ [],
                                       SPR\ $Z)),
        $LEFT = $SUBJ,
        $RIGHT = $HEAD.
```

ボトムアップにパーズすると考えるならば$LEFT が左側の子供としての入力になり、$RIGHT が右側の子供としての入力となり、$MOTHER が親としての出力と考えればいいでしょう。

5　プリンシプル

　プリンシプルとは全ての親と子供の間で守られていなければならない制約のことです。
HPSG には THE HEAD FEATURE PRINCIPLE(主辞素性原理) というプリンシプルがあって、これは非常に重要な制約なのですが、以下のような形をしています。

$$\left[\begin{array}{l} \text{SYNSEM} \mid \text{LOCAL} \mid \text{CAT} \mid \text{HEAD} \boxed{1} \\ \\ \text{DTRS} \left[\text{HEAD_DTR} \left[\text{SYNSEM} \mid \text{LOCAL} \mid \text{CAT} \mid \text{HEAD} \boxed{1}\right]\right] \end{array}\right]$$

親の HEAD 素性と子の HEAD 素性の値は構造共有されていないといけない、ということです。
HEAD の下には主辞の重要な情報がはいっていて、主辞の重要な要素は親にひきつがれる、ということです。(補語の HEAD の値は捨てられる、と思えばいいでしょう。)

ちなみに LiLFeS で書くなら以下のような形になるでしょう。

```
head_feature_principle($HEAD,$MOTHER) :-
        $MOTHER = (SYNSEM\LOCAL\CAT\HEAD\ $X),
        $HEAD = (SYNSEM\LOCAL\CAT\HEAD\ $X).
```

　他にもたくさんのプリンシプルがあって、こういったプリンシプルが主辞の親の形を形成し、しいては HPSG の構文木が決まるわけです。また、実はスキーマは THE ID PRINCIPLE (主辞句は必ずスキーマのうちどれか一つと単一化されないといけない) と呼ばれているプリンシプルで用いられています。

つまり、HPSG における親子の制約は、

HPSG における親子の制約 = (THE-ID-PRINCIPLE) $\wedge$ (THE-HEAD-FEATURE-PRINCIPLE) $\wedge$ (その他のプリンシプル $_1$) $\wedge$ ... $\wedge$ (その他のプリンシプル $_n$)

となり、THE-ID-PRINCIPLE は、

THE-ID-PRINCIPLE = (スキーマ $_1$) $\vee$ (スキーマ $_2$) $\vee$ (スキーマ $_3$) $\vee$ ... $\vee$ (スキーマ $_n$)

となります。