Hermes: Grammar and Lexicon

Carlos B. Rivera and Nick Cercone

Technical Report CS-98-02
March, 1998

ISSN 0828-3494
ISBN 0-7731-0360-0

Contents

1 Introduction

This report describes the grammar and lexicon developed for natural language (NL) access to medical databases. The grammar and lexicon, associated semantic extractor and control program form the Hermes system [Riv97a, Riv97b] (the Hermes system is described in a forthcoming technical report [Riv98]). This lexicon was first developed by Shinta Mayasari [May95, May96a, May96b]. The grammar was based on earlier work for the SystemX system at Simon Fraser University [CHJ+90, CHM+94, MC91, VPC93]. The lexicon has been further improved and expanded, and is used to access the healthcare database developed by Weidong Yu [Yu96]. The grammar is based on the Head-Driven Phrase Structure Grammar (HPSG) formalism developed by Pollard and Sag [PS87, PS94]. The current lexicon contains approximately 300 words, half of which have semantic values associated with healthcare database.

This report is divided into a further five sections. Section 2 gives an overview of the healthcare domain covered by the lexicon. Section 3 describes the HPSG formalism. Section 4 describes the Hermes grammar. Section 5 describes the structure of the lexicon. This report concludes with Section 6, which has future research directions and general remarks.

2 Healthcare domain

The lexicon is designed for the healthcare domain. This domain is currently bounded by information contained within Weidong's database. The healthcare database contains information similar to what would be in a healthcare facilities databases (e.g. hospital or medical clinic). This information includes doctor and patient demographic information, patient weight and height data, patient allergies, insurance coverage information and visit information. Appendix A describes the schema for the tables the lexicon covers.

The goal of Hermes was to access this data from a high level, modelling the type of information that a hospital administrator would want. The administrator should be able to access this information without having to know the database schema or a traditional database retrieval language (e.g. SQL). Hermes should be able to answer questions similar to the following:

- Give me the name of the head of radiology.
- List patients seen by head of radiology.
- Office and home phone number for Dr. James Aebig.
- Total number of pediatricians.
- Average age of all patients seen by female surgeons.
- Dr. Aebig's title.

Hermes at present handles query-type questions only (e.g. *Give me the head of radiology*) but not yes/no questions (e.g. *Is James Aebig head of radiology?*). This present a limitation, but it can be overcome. Instead of asking is James Aebig head of radiology ask who is head of radiology and compare.

2.1 Doctor table

The doctor table contains demographic information for the twenty doctors in the database. The primary key is doctor number (DEA#) and secondary key is the first and last names. The doctors address is contained in the street name, post office box number, city, province and postal code columns. The table contains home and office phone numbers. The doctor's gender, date of birth, age, speciality, title, department, graduation date and graduate school is also within this table.

The doctor table contains proper names (as does the demographic table) which must be properly translated. The doctor table contains many nominal compounds (e.g. family doctor, general practitioner, office/home phone number) that must be handled [MPF96]. There are also many doctor specific nouns required for the doctor's speciality (surgeon, dentist, pediatrician, radiologist, gynecologist, obstetrician), department (radiology, family medicine, pediatrics, surgery) and title (dean, director).

2.2 Demographic table

The demographic table contains demographic information for the 1000 patients in the database. The primary key is the patient information number (PID#) and secondary key is the first and last names. The table contains columns with the patients previous last names (if any), phone number, gender, date of birth, age and marital status. The patient's address is contained in the street name, post office box, city, province and postal code columns. The status and date of death columns keep track of whether the patient is alive or deceased.

2.3 Visit table

The visit tables contains information from patient examinations. Each of the 2000 rows in the table represent a single examination of a patient by a doctor. The PID# of the patient being examined is the foreign key for this table. The attending or referring doctor DEA# can also be used to key into this table. The table contains admission and discharge dates, as well the length of time the patient spent in the health care facility. The type of admission and diagnosis are also in this table. The visit table requires some additional nominal compounds: admission date, discharge date. Additionally, phrases such as length of stay must have associated semantic meaning.

2.4 Other tables

There are three other tables that Hermes accesses. The allergy table contains 1000 tuples with PID#, diagnosis date and allergy the patient has been diagnosed with. The insurance table contains 1000 tuples with PID#, insurance number, insurance source and insurance date. The weight and height table contains 1000 tuples with PID#, weight, height and measurement date.

3 HPSG

HPSG was first proposed by Pollard and Sag in [PS87] and further refined in [PS94]. HPSG is a combined semantic and syntactic formalism than handles sentences, phrases and words. HPSG is similar to other grammatical formalisms (categorical grammar, lexical-functional grammar, generalized phrase structure grammar, etc.) in that it is based on *unification* and *subsumption*. Unification is the joining of two structures (partial or whole) into a single structure that contains no more or less information than the original two structures. Subsumption is the ordering of values from general to more specific. Section 3.2 defines unification and subsumption with examples. The data structure central to the HPSG formalism is the Attribute Value Matrice (AVM). AVMs are used to store syntactic and semantic information about words, phrases and sentences. The rest of this section discusses the HPSG formalism. These sections are not intended as a tutorial, since there are good tutorial of HPSG available on the World Wide Web [Man96, Mat97, Sag95].

3.1 Attribute Value Matrice

An AVM is a feature structure that represents a *sign*. A sign is a lexical entry (word), phrase or a partial AVM. Figure 1 shows an AVM for the word *she* taken from [PS94, pg. 20]. The words in capitals are *attribute* (also called *features*), followed by their *values*. The lowercase words in italics are *sorts*. Sorts label the attribute values. For example, there are two sorts a sign takes *word* as shown in Figure 1 and *phrase*. AVMs are required to be *totally well-typed* and *sort-resolved*. Totally well-typed means that all attributes have all subattributes. Thus, all AVMs with the LOCAL attribute have subattributes CATEGORY, CONTENT and CONTEXT. This also means that all attributes must have a value (this value can be unbound e.g. NUM *num*). Sort-resolved means that attribute values have an inherent hierarchy. This means that values must be sorted from general to specific. Values are either simple (or atomic), like *sing(ular)*, which is the value of the attribute NUM(BER), or complex. Complex values include lists, written in angle brackets (⟨⟩); sets, written in curly brackets ({}); or another AVM, for example the value for INDEX.

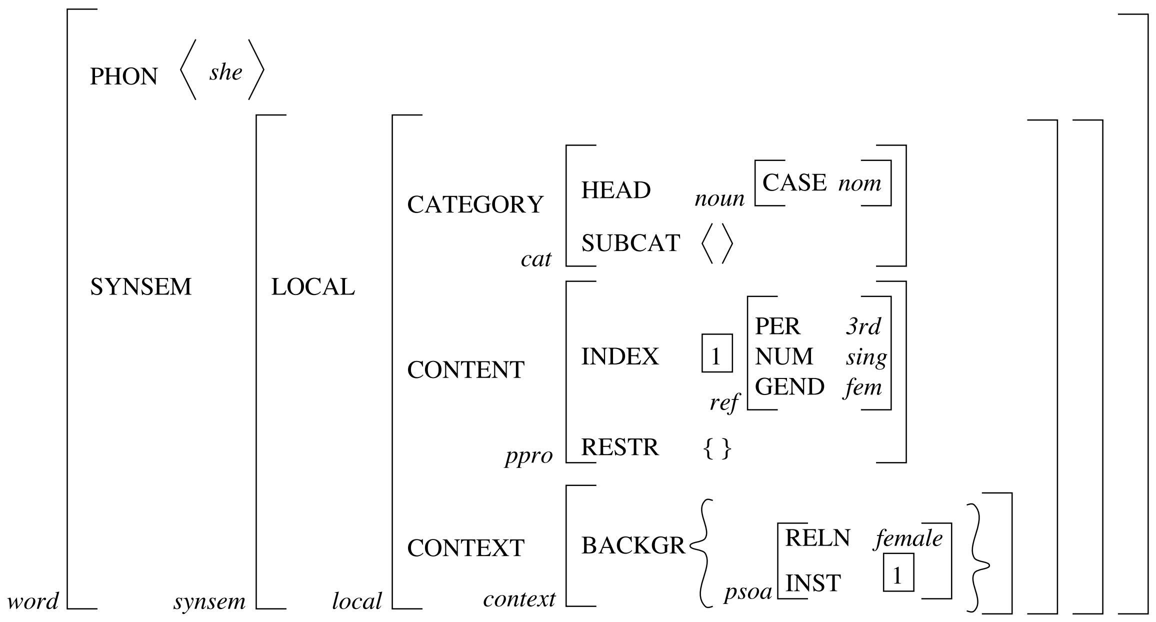

Figure 1: AVM for *she*

There are two attributes that all words have in HPSG: PHON(OLOGY) and SYNSEM. Phrases and sentences have the attributes SYNSEM and QUANTIFIER-STORE (QSTORE). PHON takes the value of a "feature representation that serves as the basis for phonological and phonetic interpretation" [PS94, pg. 15]. SYNSEM contains the semantic and syntactic values for a word or the same for a phrase. QSTORE is used to store information about quantifiers and determiners. The SYNSEM and QSTORE values for *every book* is shown in Figure 2. The $\boxed{1}$ and $\boxed{2}$ are called tags, and indicate *structure sharing*. Structure sharing means that the two structures pointed to are token-identical, which is stronger than just being the accidently the same. Figure 2 shows a path notation often used in HPSG. Instead of showing the complete AVM to the CONTENT value, the attribute names are concatenated with a vertical bar. For example, the path notation for the NUM attribute in Figure 1 is SYNSEM|LOCAL|CONTENT|INDEX|NUM *sing.*

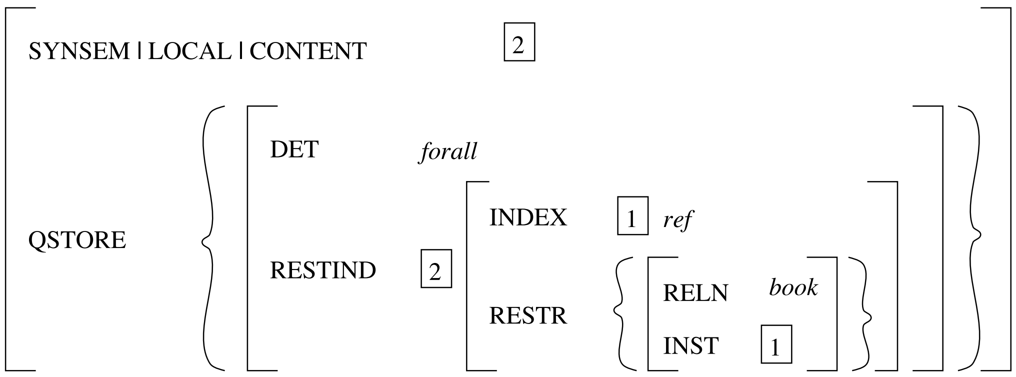

Figure 2: QSTORE and CONT values for *every book*

The SYNSEM attribute's value has two attributes: LO(CAL) and NONL(OCAL). LOC contains the semantic and syntactic information from the current word or phrase made up of adjacent words. NONLOC contains semantic and syntactic information from words (or phrases) that are missing in certain constructions (unbounded dependency phenomena). The sentence *It's Kim who Sandy loves* is an example. The word *loves* is usually a transitive verb (e.g. *Kim loves Sandy*), which means that a noun phrase should follow it, but in the case of the first sentence the pronoun, *who*, fills in for this noun phrase. The description of these type of constructions and how they are handled by HPSG is complicated and outside the scope of this report. For further information refer to Chapter 4 of [PS94].

The LOC attribute's value has attributes CAT(EGORY), CONT(ENT), and CONTEXT (CONX). CAT contains syntactic information of words and required grammatical arguments (e.g. a transitive verb requires a subject and object). CONT contains the semantic interpretation for this word for any phrase that contains it. This information is context-independent (e.g. blue). CONX contains "context-dependent linguistic information" (e.g. beside the desk).

The CAT attribute's value has two attributes: HEAD and SUBCAT. The HEAD attribute contains the part of speech information for words or the head of a phrase. This value is structure-shared with its phrasal projections. The parts of speech are divided into two sorts: *substantive* (subst) and *functional* (func). Substantive parts of speech include *nouns*, *verbs*, *adjectives* (adj), and *preposition* (prep). Functional parts of speech include *determiners* (det) and *markers* (mark, used in complementizers). Different parts of speech take additional attributes as values. CASE for nouns and PREPOSITION-FORM (PFORM) for prepositions are examples. The SUBCAT attribute contains a list of synsem objects corresponding to values of other signs selected as complements. The value for SUBCAT for *she* is empty since pronouns do not take complements. The value for SUBCAT for a verb (e.g. walks) would be an synsem object representing a noun phrase.

The CONT attribute's value has attributes INDEX and RESTR(ICTION). INDEX is a parameter introduced by noun phrases (NP) in situation semantics. INDEX has sorts *ref(erential)*, *there* and *it*. The *there* and *it* sorts are used only for the expletive pronouns *there* and *it* (e.g. *It was surprising he was nominated*). The INDEX attribute's value has attributes: PER(SON), NUM(BER) and GEND(ER). Two words are *coindexed* if there INDEX attributes are structure-shared (e.g. *he* and *himself* in *he shaved himself*). RESTR(ICTION) is used for further semantic restrictions. Its sort is a set of parametrized states-of-affairs (psoas). These psoas have attributes: RELATION (RELN) and INST(ANCE). Figure 3 shows the CONT value for book.

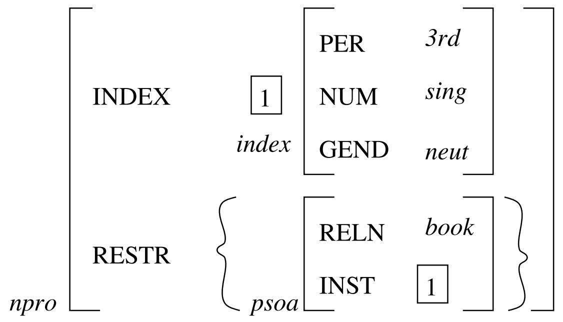

Figure 3: CONT value for book

The CONX attribute's value has a single subattribute, BACKGR(OUND). BACKGR takes a set of psoas as values. These psoas are similar to the ones in the CONT attribute but instead represent "conditions that correspond to presuppositions or conventional implicatures" [PS94, pg. 27]. For example, from Figure 1 the BACKGR value corresponds to the presupposition that *she* refers to a female. The French feminine pronoun *elle* would not have this presupposition since French is a 'grammatical gender' language.

The QSTORE attribute's value has attributes: DET(ERMINER) and RESTRICTED-INDEX (RESTIND). DET takes an atomic value corresponding to the type of the quantifier or determiner (e.g. *the*, *exists*, *forall*). The RESTIND takes the same value as the CONT attribute without the sort. For example the RESTIND value for *her* is the same as the INDEX value in Figure 1 without the *ref* sort.

3.2 Subsumption and unification

The HPSG formalism insists on a there being an order to the attributes and values. This means that some values and attribute are more general or specific than others. For example, the (1) AVM shown in Figure 4 has the NUM attribute set to *num* the default value, but the (2) AVM has the NUM attribute set to *sing*. Since the first AVM is more general than the second it subsumes it. Subsumption plays an important part in unification. Since, unification subsumes one value with another when creating the resultant AVM. As stated previously, unification combines the information from two AVMs, without adding anything more. This results in the most general description that is compatible with the two inputs. For example, from Figure 4 unifying (2) and (3) results in (5). If the information in the two original AVMs is incompatible, unification fails. For example, unifying (2) and (4) fails because neither *sing* nor *plur* subsumes one another. The AVMs in (6), (7) and (8) show the difference between structure sharing and the values just being identical. Unifying (6) and (7) creates the AVM shown in (8). If the value for D and A was not structure shared but just identical then the value for D would be B and E but A would remain only B.

3.3 Rules (schemas)

The great strength of HPSG is the limited number of rules and principles used to join words into phrases and sentences. These rules and principles control how signs are joined to create larger

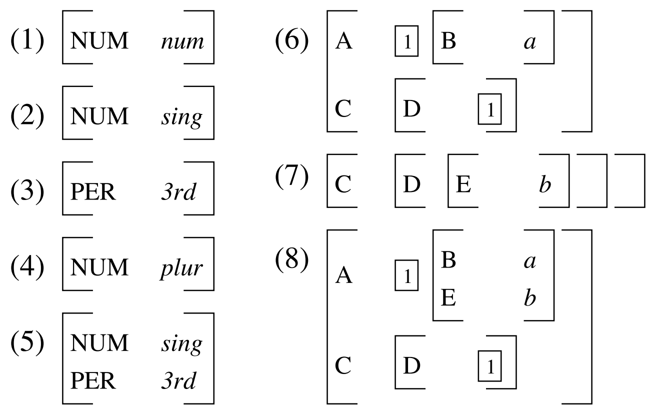

Figure 4: AVMs demonstrating subsumption and unification

signs. There are five schema that handle different English grammatical structures.

Schema 1 (Head-Subject Schema)
This schema licences saturated phrases with a phrasal head daughter and one other daughter that is a complement. This handles many grammatical rules including:

S -> NP, VP	(e.g. She walks.)
NP -> DET, N'	(e.g. the doctor)

Figure 5 shows the general form of a phrase licensed by this schema.

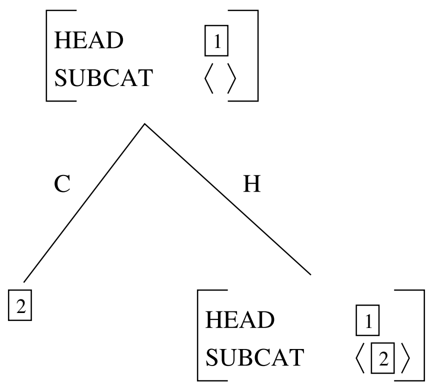

Figure 5: General form of Schema 1

Schema 2 (Head-Complement Schema)
This schema licenses phrases that have a lexical head daughter (a word rather than a phrase) and zero or more complement daughters. These daughters have satisfied all their subcategorization requirements except the subject. This handles many grammatical rules including:

VP -> V, NP	(e.g. is a doctor)
PP -> P, NP	(e.g. of radiology)
VP -> V, PP	(e.g. is after)
VP -> V, NP, NP	(e.g. gave the dog a bone)

Figure 6 shows the general form of a phrase licensed by this schema.

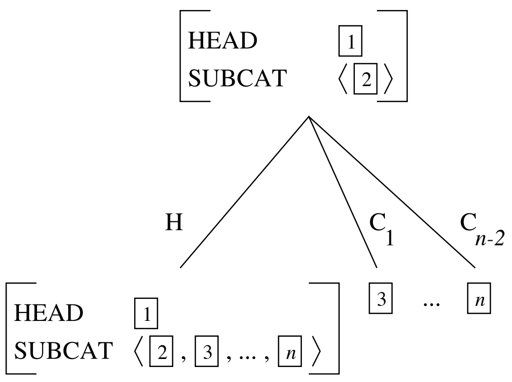

Figure 6: General form of Schema 2

Schema 3 (Head-Subject-Complement Schema)
This schema licenses saturated phrases with 'subject-auxiliary inversion' clauses. This schema requires verbs to have an additional HEAD subattribute INV which takes the value + or -. A + means the verb allows inverted structures (e.g. *Can Kim go?*). Figure 7 shows the general form of a phrase licensed this schema. Additionally the HEAD in this figure must be a lexical sign (a word).

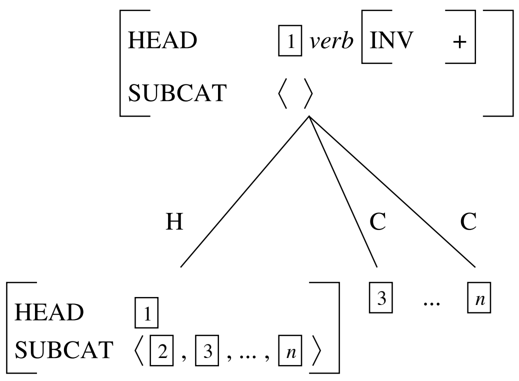

Figure 7: General form of Schema 3

Schema 4 (Head-Marker Schema)
This schema licences phrases containing complementizers (e.g. certain uses of *that* and *it*). Complementizers have a HEAD value of *marker* and are combined with another element that heads the constituents (e.g. *that boy*). Figure 8 shows the general form of a phrase licensed by this schema.
Schema 5 (Head-Adjunct Schema)
This schema licences phrases with an adjunct and the head it selects. Adjuncts in HPSG have to be of sort *substantive* since *functional* sorts are handled by Schema 4. The MOD value of the adjunct is structure shared with the SYNSEM value of the head daughter. The head daughter's SLASH value must be the the empty set. This schema handles grammatical rules such as:

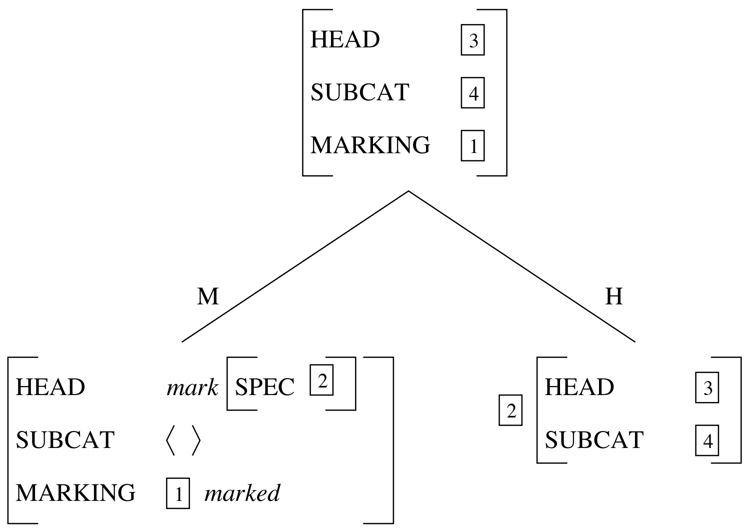

Figure 8: General form of Schema 4

N -> ADJ, N	(e.g. red book)
N' -> N', PP	(e.g. head of radiology)
N' -> NP, N	(e.g. female doctor)
VP -> VP, PP	(e.g. is above the desk)

3.4 Universal principles

HPSG has many principles, some of which are only used with certain languages (e.g. The Relative Uniqueness Principle, The Clausal Rel Prohibition). The most applied principles are: Head feature principle, Subcategorization principle, Specifier principle, Marking principle and Semantic principle.

Head feature principle The HEAD value of any headed phrase is structure-shared with the HEAD value of the head daughter.

Subcat principle The SUBCAT value of any headed phrase is the concatenation of the list value of the SYNSEM values of the COMPLEMENT-DAUGHTERS.

Specifier principle In a headed phrase, whose daughter has a HEAD value of sort *functional*, the SPEC value of that value must be structure-shared with the phrase's SYNSEM value.

Marking principle The MARKING value of a headed phrase is token-identical with the MARKING value of daughter if any.

Semantic principle The RETRIEVED value of a headed phrase is a set disjointed from the QSTORE value set. The union of these two sets is the union of the QSTORE value of the daughters. As well, if the ADJUNCT-DAUGHTER or HEAD-DAUGHTER CONT value is a *psoa* then the CONT|NUCLEUS value for the phrase is token identical with that of the semantic head. The CONT|QUANTS value is the concatenation of the RETRIEVED value and the DAUGHTER QUANTS value, else the RETRIEVED value is the empty list and CONT value for the phrase is token-identical with the DAUGHTER.

4 Hermes grammar

The Hermes grammar closely mirrors what is outlined in Pollard and Sag but does not have every feature described. The HPSG formalism describes how to parse phrases, gives description of the structure for signs and required rules, but does not state how these should be implemented. The tool most used for implementing HPSG based grammars and lexicons, and used by Hermes is the Attribute Logic Engine (ALE)[CP94]. ALE is "an integrated phrase structure parsing and definite clause logic programming system in which terms are typed feature structures". ALE is loaded by Sicstus Prolog 2.1#9 and the grammar and lexicon is then compiled.

4.1 AVM

The AVM used by Hermes to describe signs is similar to the one described in the previous section. The Hermes grammar does not have the QSTORE, PHON, NONLOC or CONX attributes, but does have the SYNSEM, LOC, CAT and CONT attributes. The Hermes grammar has sorts *word* and *phrase*, but the attributes in both cases are the same. Figure 9 shows the AVM for the word *doctor* in the Hermes grammar.

The CAT attribute's value has attributes: HEAD, MARKING, SPECIFIER (SPR), and SUBCAT. Hermes' HEAD and SUBCAT attributes serve the same function and structure as the corresponding attributes in HPSG. MARKING is used with some complements (for, that) and conjunctions (and, or). MARKING takes the values, *for* and *that*, when they are used as complements and the values, *and* and *or*, when they are used as conjunctions.The SPR attribute is used to set specifier selection. SPR takes a list of synsem objects that correspond to specifiers the word or phrase takes. For example, common nouns (e.g. *book*, *doctor*) take determiners as specifiers therefore the SPR value for them would have a synsem object for determiners as an element in its list.

The CONT attribute's value has attributes: ACTION, QUERY, REF and SEM_MARK. The ACTION attribute takes an atomic value that corresponds to the database action required. These actions are *average* and *count*. After implementing and testing the grammar a short-coming of handling database action in this manner was found. The Hermes grammar handles conjunctions, such as *weight and height*, but adding *average* to the front of this causes a problem. The scope of the *average* modifier is ambiguous and the ACTION attribute does not limit it. A better method for the next version of the Hermes grammar would be to include an ACTION attribute with each element in the QUERY list.

The QUERY attribute takes a list of elements containing the COLUMNS and TABLE attributes. Each element corresponds to a single table being queried, therefore having more then one unique TABLE value in the list means a join is required. The TABLE attribute takes an atomic value representing a database table. These values are *client_rel*, *doctors_rel*, *allergy_rel*, *insurance_rel*, *wheight_rel* and *visit_rel*. The sort for COLUMNS controls what column is being selected. If more then one column is being selected then multiple elements must be in the QUERY list. The sort values for *doctors_rel* and *client_rel* TABLE values are *name*, *gender*, *specialty*, *title*, *dept*, *age*, *city*, *address*, *phone*, *birthday*, *marital*, *status*, *date_of_death* and *relig*. The *name* sort has subsorts *fname* and *lname* and the *phone* sort has subsorts *hphone* and *ophone*. The sort values for *allergy_rel* are *allergy* and *diag_date*. The sort values for *insurance_rel* are *insurance_num*, *insurance_type* and *insurance_date* and the sort values for *wheight_rel* are *weight*, *height* and *wh_date*. The sorts for *visit* are *visit_age*, *ref_physician*, *att_physician*, *admission_date*, *discharge_date*, *length_stay*, *type_admission*, *medical_service*, *nursing_station*, *outpatient_clinic*, *diagnosis*, *isolation* and *left*.

The number and type of attributes for the COLUMN value differ based on the TABLE attribute. The COLUMNS attribute's value has attributes L_NAME, F_NAME, AGE, SEX, CITY, MAR-

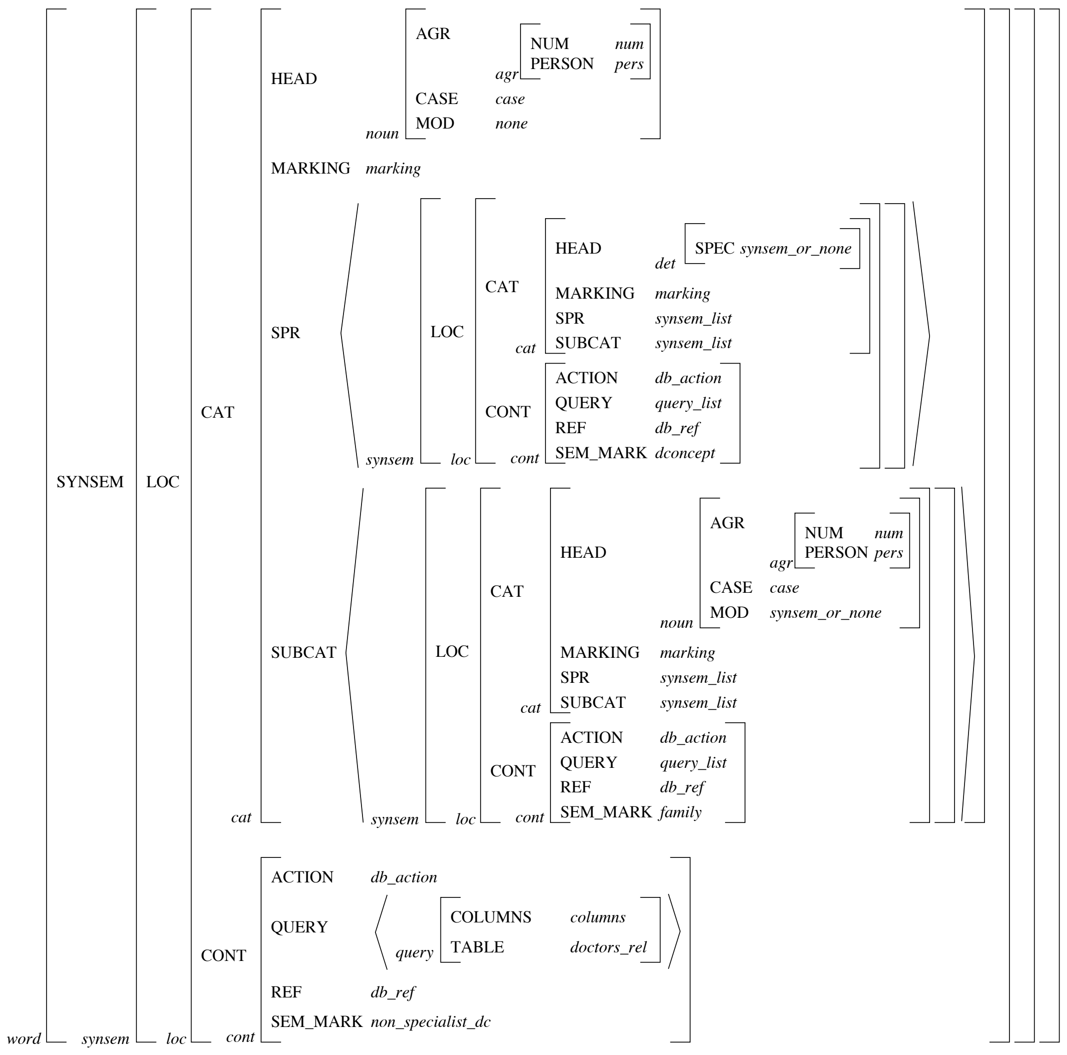

Figure 9: AVM for *doctor*

ITAL_STATUS, PROV, SPECIALTY, TITLE, DEPT, DATE_OF_BIRTH, STATUS and RELIGION for *client_rel* and *doctors_rel* TABLE values. The attributes are ALLERGY and DIAG_DATE for *allergy_rel* and INSURANCE_NUM, INSURANCE_TYPE and INSURANCE_DATE for *insurance_rel*. The attributes are WEIGHT, HEIGHT and WH_DATE for a *weight_height* TABLE value. The attributes are VISIT_AGE, REF_PHYSICIAN, ATT_PHYSICIAN, ADMISSION_DATE, DISCHARGE_DATE, LENGTH_STAY, TYPE_ADMISSION, MEDICAL_SERVICE, NURSING_STATION, OUTPATIENT_CLINIC, DIAGNOSIS, ISOLATION and LEFT for *visit_rel*. These attributes all take attribute DVAL which takes *eq*, *neq*, *lt*, *gt*, *lte*, *gte* and *ltgt* sorts. These sorts all take a single argument of sort *val_type*, except *ltgt* which takes two arguments of sort *val_type*. The different values for *val_type* are shown in Figure 10. The sort *proper names* is the complete list of all proper names in the database.

The REF attribute is used with prepositions to control the type of nouns they modify. The REF value takes sorts *db_ra* or *db_val*. The *db_ra* has subsorts *db_rel* and *db_attr*. The SEM_MARK attribute is used to control what modifiers or complements a word takes. Figure 11 shows the

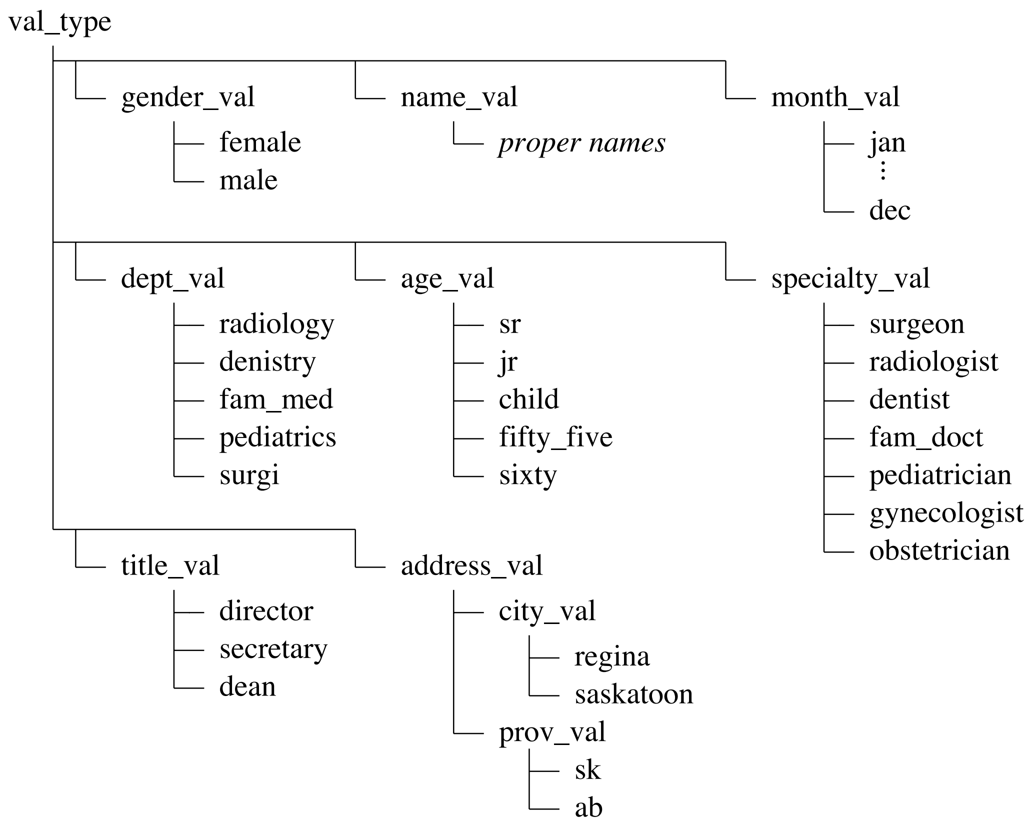

Figure 10: *val_type* hierarchy

hierarchy of sorts for SEM_MARK. Section 5 describes how SEM_MARK is used to control what words are joined.

4.2 Rules and principles

The Hermes grammar has the Head-subject, Head-complement, Head-subject-complement and Head-adjunct schemas described in Section 3.3. There are also schemas in Hermes to handle specifiers (determiners) and conjunctions (and, or). The specifier head rule licenses phrases that have specifiers followed by what they are specifying (e.g. *the doctor*). The conjunct rule license phrases that have two daughters separated by a conjunct. The two daughters must have the same HEAD and SUBCAT values (e.g. the HEAD of both must be nouns with the same SUBCAT value).

The Hermes grammar has the Head feature, Subcat, Specifier (with the added constraint that the SPEC-DAUGHTER's SUBCAT list is empty) and Marking principles that apply the same as corresponding HPSG principles. The Hermes semantic principle is greatly different from the HPSG semantic principle.

The Hermes semantic principles controls how the ACTION, SEM_MARK, REF and QUERY attributes in the semantic mother are formed from the semantic head daughter and complement daughters. The ACTION attribute is obtained by unifying the ACTION attribute of the semantic head daughter and complement daughter. The REF attribute is inherited from the semantic head daughter. The SEM_MARK of the mother is inherited from the semantic head daughter except in

```
sem_mark
 └─ dconcept
     ├─ general_con
     └─ specific_con
         ├─ time
         │   ├─ spec_time
         │   └─ date
         ├─ people
         │   ├─ ppl_name
         │   │   ├─ firstname
         │   │   │   ├─ prename
         │   │   │   └─ midname
         │   │   ├─ lastname
         │   │   └─ names
         │   ├─ ppl_number
         │   │   ├─ person
         │   │   └─ family
         │   ├─ ppl_sex
         │   │   ├─ no_gender
         │   │   └─ gender_specific
         │   └─ ppl_job
         │       ├─ admin_job
         │       ├─ doctor_job
         │       │   ├─ non_specialist_dc
         │       │   └─ specialist_dc
         │       └─ client_job
         ├─ place
         │   ├─ building
         │   │   ├─ residence_bldg
         │   │   └─ work_bldg
         │   │       ├─ dept_dc
         │   │       ├─ dept_name_dc
         │   │       └─ spec_dept_name_dc
         │   └─ non_building
         ├─ things
         │   ├─ comm_device
         │   └─ food
         └─ measures
             ├─ unit_measure
             └─ group_measure
```

Figure 11: SEM_MARK hierarchy

the cast of prepositional phrases. In prepositional phrases, SEM_MARK is obtained by unifying SEM_MARK of the semantic head daughter and complement daughter. Dealing with the QUERY attribute and its subattributes, TABLE and COLUMN, is more involved since tables joins and multiple column selections must be handled. There are three cases that must be dealt with. The first case is when the semantic head and complement daughter TABLE and COLUMN attributes are unifiable. This is true in any case where either or both of the signs have no TABLE or COLUMN values associated with it. It is also true when the words have non-contradicting TABLE and COLUMN values. For example, the word *female* has no TABLE value set but has COLUMN subattribute SEX value of *eq female* and the word *surgeon* has a TABLE value of *doctor_rel* and COLUMN subattribute SPECIALTY value *eq surgeon*. Unifying the semantic content of these two words when forming the phrase *female surgeon* works because there is no conflicting values for COLUMN. The second case is when the semantic head and complement daughter have the same TABLE value (not undefined which the first case handles) but have different COLUMN sort values. This case handles conjunctions (e.g. *height and weight*). In this case, the COLUMN attribute value of the complement daughter is concatenated to the end of the COLUMN list of the semantic head daughter. The third case covers when a join is required. The two TABLE values are compared to see if they have a common database column (e.g. DEA# in Doctor and Visit table). If there is no common database column the semantic principle fails as will the attempt to join the signs. If there is a common database column then the COLUMN and TABLE values for the complement daughter are concatenated to the end of the QUERY list of the semantic head daughter.

5 Lexicon

One of the strengths of ALE and HPSG is the ease of developing new lexicons. ALE has the ability to define macros that represent arbitrarily complex feature structures. These macros simplify the creation of the lexicon by allowing the user to define macro definitions for parts of speech and semantic representations, then using them in defining words in the lexicon, rather than having to write the full definition. The macro for common nouns is shown below. A macro takes as many arguments as needed and may include other macros in its definition. A macro definition consist of a Prolog atom that labels the macro, Prolog variable enclosed in parenthesis that are the macro arguments, the keyword *macro* followed by the macro definition in parenthesis and a period at the end. The definition enclosed in parenthesis is unified into a feature structure the macro represents. To use a macro the macro label is prefixed with the @ symbol and arguments are supplied (these arguments may be further variables or macros), as shown for the *head_s* and *spr_s* macros used in the *common_s* macro.

```
head_s( X )          macro ( loc:cat:head:X ).
spr_s( X )           macro ( loc:cat:spr:X ).
common_s             macro ( @head_s(noun),
                             @spr_s([@determiner_s]) ).
```

There are semantic macros defined to set all the subattributes in the QUERY value. The following macros do this:

```
cont( Cont )         macro ( synsem:loc:cont:Cont ).
action( Act )        macro @cont( action:Act ).
smarker( Marker )    macro @cont( sem_mark:Marker ).
query( Queries )     macro @cont( query:Queries ).
table( Tbl )         macro @query( hd:table:Tbl ).
ref( Ref )           macro @cont( ref:Ref ).
columns_type( Col )  macro @query( hd:columns:Col ).
columns( Col )       macro @columns_type( hd:Col ).
```

These macros are used to assign the database meaning to lexical entries. There is a macro to assign values to the TABLE attribute and to a sort for the COLUMN attribute. An example would be assigning the noun *weight* the value *wheight_rel* for TABLE and *weight* as a sort for the COLUMN attribute.

```
 db_tbl_col( TblName, ColName )
                     macro ( @table(TblName), @columns(ColName) ).
```

There is also macros to assign values to a COLUMN subattribute including an operator if needed. An example would be assigning the noun *dentist* the value of *eq dentist* for the SPECIALTY subsort of the *doctors_rel*.

```
     db_feat_val( DbFeatures, DbVals )
                         macro ( @columns(DbFeatures:arg1:DbVals) ).
     db_feat_val( DbFeatures, Op, DbVals )
                         macro ( @columns(DbFeatures:(Op, arg1:DbVals)) ).
```

The rest of this section describes the implementation of the seven parts of speech that Hermes handles.

5.1 Nouns

Nouns make up the bulk of the words required to cover the medical domain. These nouns are broken into two categories, those with (*doctor*, *department*, *patient*) and without (*apple*, *bottle*) semantic meaning in the medical domain. The nouns with semantic meaning are broken down further into nouns that takes nouns as complements and act as a modifier of other nouns (*phone* in *phone number* and *home phone*), nouns that take noun complements but do not act as modifiers (*doctor* in *family doctor*), nouns that takes no complements but acts as a modifier (*department* in *department head*) and nouns that take no complements and does not act as a modifier (*radiologist*). To control what types of nouns a noun takes as modifier or complement the SEM_MARK attribute is used. The nouns that acts as complements include in there SUBCAT value a macro defining what type of value the SEM_MARK of the complimenting word must have. For example, the word *doctor* only modifies words with the SEM_MARK value of *family*. The following four macro definitions are used for nouns:

```
% nouns that takes a complement and may act as a modifier
     npCompMod( Comp, Mod )
                         macro ( @head( noun ),
                                 @spr( [@head_s( det )] ),
                                 @mod( ( @head_s( noun ),
                                         @smarker_s( Mod ) )
                                       ),
                                 @subcat( [ ( @head_s( noun ),
                                              @smarker_s( Comp ) ) ]
                                        )
                                 ).

% nouns that take a complement and may not act as a modifier
     npCompNoMod( Comp )
                         macro ( @head( noun ),
                                 @spr( [@head_s( det )] ),
                                 @mod( none ),
                                 @subcat( [ ( @head_s( noun ),
                                              @smarker_s( Comp ) ) ]
                                        )
                          ).
% nouns that take no complement but may act as a modifier
     npNoCompMod( Mod )
                         macro ( @head( noun ),
                                 @spr( [@head_s( det )] ),
                                 @mod( ( @head_s( noun ),
                                         @smarker_s( Mod ) )
                                     ),
                                 @subcat( e_list )
                         ).
% nouns that take no complement but may not act as a modifier
```

```
    npNoCompNoMod
                          macro ( @head( noun ),
                                  @spr( [@head_s( det )] ),
                                  @mod( none ),
                                  @subcat (e_list )
                          ).
```

Proper nouns are treated as a special case of these four cases. The *people* subsort of SEM_MARK is used with proper names. The *prename* sort is used for titles (*Mr.*, *Dr.*) and complements the *firstname* or *lastname* sorts. The *firstname* sort is used with given names (*Peter*, *David*) and complements the *lastname* sort.

5.2 Pronouns

Personal pronouns are broken into three groups for pronouns that are the subject (*I*, *he*, *she*), object (*me*, *him*, *her*) or both (*you*) in a sentence. Due to the nature of the Hermes domain and the limitation that only query-type questions may be asked the only pronoun generally used is *I* (e.g. *I want the information*). Relative pronouns (*who*, *whom*) do not work with the present version of Hermes because of the lack of the NONLOC attribute and associated schemas and principles. Pronouns in Hermes have no semantic information associated with them. The following macros are used to define pronouns:

```
    np_s                  macro ( @head_s( noun ),
                                  @subcat_s( [] ) ).
    npObj_s               macro ( @np_s,
                                  @case_s( acc ) ).
    npSubj_s              macro ( @np_s,
                                  @case_s( nom ) ).
% macro for pronouns that are subjects
    ppronSubj_lex         macro ( word, @npSubj ).
% macro for pronouns that are objects
    ppronObj_lex          macro ( word, @npObj ).
```

5.3 Verbs

Verbs are the most complicated part of speech for any parsing systems due to the many forms they take and the differing number of arguments they need. There are macros for intransitive (e.g. *The earth trembled*), transitive (e.g. *The earthquake destroyed the city*), ditransitive (e.g. *Kim give Sandy books*), control (e.g. *I want to see the report*) and auxiliary (e.g. *can he walk*) verbs. A limited number of verbs have associated semantic meaning in Hermes: *visit*, *seen*, *examined*, *average* and *live*. The following macros are used to define verbs:

```
    vp                    macro ( @head( verb ),
                                  @vform( bse ),
                                  @aux( minus ),
                                  @marking( unmarked ),
                                  @mod( none ) ).
    intrans               macro ( @vp,
                                  @subcat( [ @np_s ] ) ).
    intransPP( PForm )    macro ( @vp,
                                  @subcat( [ @npSubj_s, @prep_s( PForm ) ] ) ).
    trans                 macro ( @vp,
                                 @subcat( [ @np_s, @npObj_s ] ) ).
    ditrans               macro ( @vp,
                                @subcat( [ @np_s,
                                           @npObj_s,
                                         ( @head_s( noun ), @case_s( acc )) ] ) ).
    icontrolv( VForm, CForm )
                        macro ( @verb( VForm ),
                                @mod( none ),
                                @subcat( [ @np_s,
                                           @xcomp_s( CForm ) ] ) ).
    tcontrolv( VForm, CForm )
                        macro ( @verb( VForm ),
                                @mod( none ),
                                @subcat( [ @npSubj_s,
                                           @npObj_s,
                                           @xcomp_s( CForm ) ] ) ).
    auxil( CompForm )   macro ( @head(verb),
                                @vform(fin),
                                @aux(plus),
                                @marking( unmarked ),
                                @subcat( [ @npSubj_s,
                                          (@head_s( verb ), @vform_s( CompForm ),
                                           @subcat_s( ne_synsem_list ) ) ] ) ).
% intransitive verbs
    intrans_lex          macro ( word, @intrans ).
    intransPP_lex( PForm )
                         macro ( word, @intransPP( PForm ) ).
% transitive verbs
    trans_lex            macro ( word, @trans ).
% ditransitive verbs
    ditrans_lex          macro ( word, @ditrans ).
% control verbs
    icontrolv_lex( VForm, Form )
                         macro ( word, @icontrolv( VForm, Form ) ).
    tcontrolv_lex( VForm, Form )
                         macro ( word, @tcontrolv( VForm, Form ) ).
% auxiliary verbs
    aux_lex( CompForm )    macro ( word, @auxil( CompForm ) ).
```

5.4 Adjectives

Adjectives are broken into two groups those that modify nouns with semantic meaning (in which case the adjective probably has additional semantic information) and those that modify nouns without semantic meaning. Adjectives have noun synsem object for there MOD attribute value. Adjectives that modify nouns with semantic meaning use the SEM_MARK attribute to control what nouns they modify (e.g. the adjective *female* may only modify nouns that have a *ppl_job* sort value for SEM_MARK). The `db_tbl_col`, `db_feat_val` and `action` macros are used to set semantic meaning for adjectives. There is a limited number of adjectives with associated semantic meaning in Hermes: *first*, *last*, *full*, *female*, *male*, *total* and *average*. Hermes presently does not have any adverbs defined, but they would be handled similarly to adjectives but modify verbs instead of nouns. The following macros are used to define adjectives:

```
    adjective           macro ( @head( adj ),
                                @subcat( [] ),
                                @spr( e_list ),
                                @marking( unmarked ),
                                @mod( @common_s ) ).
    adj( Mark )         macro ( @adjective,
                                @mod( @smarker_s( Mar k) ) ).
% adjectives that modify nouns with semantic meaning
    adj_lex( Marker )   macro ( word, @adj( Marker ) ).
% adjectives that modify nouns without semantic meaning
    adjective_lex        macro ( word, @adjective ).
```

5.5 Prepositions

Prepositions are handled with a single macro in Hermes that takes the sort for the preposition as argument (e.g. a sort of *after* for *after*). The preposition *for* and *of* are exceptions in Hermes, since they have semantic meaning associated with them. The preposition *of* only modifies nouns whose SEM_MARK value has sort *specific_con* or one of its subsorts. This allows prepositional phrase such as *of radiology* and *of birth*. The preposition *for* only modifies nouns SEM_MARK value has sort *db_ra*. The following macros are used to define prepositions:

```
    prep( PForm )       macro ( @head( prep ),
                                @spr( e_list ),
                                @pform( PForm ) ).
    for_prep            macro ( @prep( for ),
                                @marking( Marking ),
                              ( @mod( ( @head_s( noun ),
                                        @ref_s( db_ra )) );
                                @mod( @head_s( verb ) ) ),
                                @subcat( [ @head_s( dummy ),
                                         ( @head_s( noun ),
                                           @marking_s( Marking ),
                                           @ref_s( db_val ) ) ] ) ).
    preposition( PForm ) macro ( @prep(PForm),
                                @marking( Marking ),
                              ( @mod( @head_s( noun) );
                                @mod( @head_s( verb ) ) ),
                                @subcat( [ @head_s( dummy ),
                                         ( @head_s( noun ), @marking_s( Marking ) ) ] ) ).
    preposition( PForm, Mark )
                        macro ( @prep( PForm ),
                              ( @mod( ( @head_s( noun ),
                                        @smarker_s( Mark ) ) ) ),
                                @subcat( [ @head_s( dummy ), @head_s( noun ) ] ) ).
% normal prepositions
    preposition_lex( PForm )
                        macro ( word, @preposition( PForm ) ).
% prepositions that only modify certain nouns
    preposition_lex( PForm, Mark )
                        macro ( word, @preposition( PForm, Mark ) ).
$ for
    for_lex             macro ( word, @for_prep ).
```

5.6 Conjunctions

The Hermes lexicon currently handles only the *and* and *or* conjunctions. Conjunctions are handled by the conjunction rule described in Section 4.2. Combining the semantic meaning of the two signs being joined is handled by the semantic principle. The following macros are used to define conjunctions:

```
    conj( ConjForm )     macro ( @head( mark ),
                                 @spr( e_list ),
                                 @subcat( e_list ),
                                 @marking( ConjForm ) ).
    conj_lex( ConjForm ) macro ( word, @conj( ConjForm ) ).
% and
    and_coord_lex        macro ( @conj_lex( and ) ).
% or
    or_coord_lex         macro ( @conj_lex( or ) ).
```

5.7 Determiners

Determiners (articles) are easily handled in Hermes by being specifiers for nouns. The only exception is the the possessive determiner (*'s* in *John's car*). Hermes handles this type of determiner by treating this as three separate words (*john*, *s*, *car*). The *s* takes a noun as the object it specifies (licensing the phrase *s car*) and having SUBCAT value *noun* that handles *john*. The following macros are used to define determiners:

```
    determiner           macro ( @head( det ),
                                 @spec( @head_s( noun ) ),
                                 @spr( e_list ),
                                 @marking( unmarked ),
                                 @subcat( [] ) ).
    possesiveDet         macro ( @head( det ),
                                 @spec( @head_s(noun) ),
                                 @spr( e_list ),
                                 @marking( unmarked ),
                                 @subcat( [ @head_s( noun ) ] ) ).
% normal determiners
    determiner_lex       macro ( word, @determiner ).
% possive determiners
    possesiveDet_lex     macro ( word, @possesiveDet ).
```

6 Conclusion

6.1 Future research

There are several areas of future research:

lexicon expansion As stated, the lexicon currently has approximately 300 words, half of which have associated semantic meaning. These words give good coverage of some of the tables contained in the Healthcare database developed by Weidong but additional words including many with narrow semantic meaning (patient diagnosis and allergies) would be needed to cover the entire database developed by Weidong. Additionally, there is a desire to expand the words that do not have associated semantic meaning so that the Hermes grammar and lexicon could be used to parse more general phrases and sentences.

testing with actual medical database The tables of the database that Hermes cover were artificially created based on a knowledge of the tables SaskHealth keeps. The testing of the grammar and lexicon with actual SaskHealth data would help find any problems with the current grammar and lexicon design.

loosen the tie between semantic representation to database structure The current Hermes grammar is very closely tied to the medical database created by Weidong. This makes changing the grammar and lexicon for a different database and knowledge domain time consuming. On the other hand, the semantic principles developed for the Hermes grammar could be easily ported to any other relational database.

handling other type of queries The lack of knowledge from one query to the next does not allow Hermes to handle some queries that would be wanted. For example, the queries *Who is head of radiology?* and *What is his phone number?* can not be handled by the Hermes grammar and lexicon. Additional knowledge of what queries have asked in the past would better allow these type of queries to be handled.

better handle quantifier scope The current method for handling quantifier scope where only a single quantifier may be used for a phrase (e.g. *average weight and height*) is too limiting. As stated, a better method would be to encode a quantifier with each column selected.

6.2 Concluding remarks

HPSG formalism is a very good tool for developing a grammar or lexicon that handles both sentences, phrases and single words. The ability to handles phrases that do not form complete sentences is considered very important for Hermes. This ability allows the user more flexibility to form queries quickly and accurately. Separating the semantic and syntactic content for words allows a quick and relatively easily expandable lexicon. The use of ALE as the basis for the HPSG formalism makes development of the grammar and lexicon even easier through the use of macros and the parsing and anaylsis tools included in ALE. The limited number of rules and principles contributes to the ease of use of HPSG and the ease of expansion of Hermes.

The grammar and lexicon developed for Hermes give good coverage of the doctors, patient, weight and height tables. The ability to join these tables with the visit table gives a much wider range of queries that may be asked. These queries are more interesting, as well, allowing the database to be analyzed in much greater detail. Further development is required to expand the non-semantic portions of the lexicon to allow parsing of phrases that do not directly relate to the health care domain.

References

[CHJ+90] N. Cercone, G. Hall, S. Joseph, M. Kao, W. Luk, P. McFetridge, and G. McCalla. Natural Language Interfaces: Introducing SystemX. In T. Oren, editor, *Advances in Artificial Intelligence in Software Engineering*, pages 169–250. JAI Press, Greenwich, Conn., 1990.

[CHM+94] N. Cercone, J. Han, P. McFetridge, F. Popowich, Y. Cai, D. Fass, C. Groeneboer, G. Hall, and Y. Huang. SystemX and DBLearn: easily getting more from your Relational Database. *Integrated Computer-Aided Engineering*, 1(4):311–339, 1994.

[CP94] Bob Carpenter and Gerald Penn. *ALE: The Attribute Logic Engine User's Guide. Version 2.0.1*, December 1994.

[Man96] Suresh Manandhar. *A Course on HPSG grammars and typed feature formalisms.* Language Technology Group, Human Communication Research Centre, University of Edinburgh, http://www.ltg.hcrc.ed.ac.uk/projects/ledtools/grammar _writing/, 1996.

[Mat97] Colin Matheson. *HPSG Grammars in ALE.* University of Edinburgh, http://www.ltg.hcrc.ed.ac.uk/projects/ledtools/ale-hpsg/, 1997.

[May95] Shinta Mayasari. An HPSG Lexicon for a Physical Activity Database. Technical Report CS-95-09, Computer Science Department, University of Regina, September 1995.

[May96a] Shinta Mayasari. *A Natural Language Interface to a Physical Activity Database: Design and Its Implementation.* University of Regina, Summer 1996.

[May96b] Shinta Mayasari. *An Overview of System Y.* University of Regina, Fall 1996. This paper is incomplete.

[MC91] P. McFetridge and N. Cercone. Installing an HPSG Parser in a Modular Natural Language Interface. In *Computational Intelligence III*, pages 169–178. North Holland, Amsterdam, 1991.

[MPF96] P. McFetridge, F. Popowich, and D. Fass. An analysis of compounds in HPSG (Head-driven Phrase Structure Grammar) for database queries. *Data and Knowledge Engineering*, (20):195–209, 1996.

[PS87] Carl Pollard and Ivan A. Sag. Information-Based Syntax and Semantics, Volume 1: Fundamentals. CSLI Lecture Notes 13, Standford: Center for the Study of Language and Information, 1987.

[PS94] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar.* The University of Chicago Press, 1994.

[Riv97a] Carlos B. Rivera. Hermes: Design and Implementation of a Natural Language front-end to a Medical Database. November 1997.

[Riv97b] Carlos B. Rivera. *Hermes: User's manual.* University of Regina, December 1997.

[Riv98] Carlos B. Rivera. Hermes: Natural Language Interface to Medical Databases. Technical report, University of Regina, Forthcoming 1998.

[Sag95] Ivan Sag. *Materials for teaching HPSG.* Stanford University, http://hpsg.stanford.edu/hpsg/lecture-materials/lecture-materials.html, 1995.

[VPC93] C. Vogel, F. Popowich, and N. Cercone. Logic Based Inheritance Reasoning. In Morris Sloman, editor, *Prospects for Artificial Intelligence.* IOS Press, 1993.

[Yu96] Weidong Yu. *Document of Project: Natural Language Interface to Healthcare Database (database part).* University of Regina, September 1996.

Appendix A - Database schema

There are at present six tables that are accessed through Hermes:
1) Doctor - demographic information for doctors
2) Patient - demographic information for patients
3) Allergy - allergy information for patients
4) Insurance - insurance information for patients
5) Weight and height - height and weight for patients
6) Visit - information for patient visit

Doctor table

Each tuple in the doctor table shows the demographics information for a single doctor. The following list shows all the columns contain within the doctor table:
DEA# - primary key, doctor identification number
L_NAME - doctor's last name
F_NAME - doctor's first name
STREET - street name of doctor's home
POBOX - doctor's post office box (empty column)
CITY - city that doctor lives in
PROV - province that doctor lives in
POSTCODE - postal code of doctor's home
H_PHONE - doctor's home phone number
O_PHONE - doctor's office phone number
SEX - doctor's gender
DATE_OF_BIRTH - doctor's date of birth
AGE - doctor's age in years
SPECIALTY - doctor's specialty (dentist, family physician, pediatrics, radiologist, surgeon)
TITLE - doctor's title (dean, director)
DEPT - doctor's department(dentist, family physician, pediatrics, radiologist, surgeon)
GRAD_DATE - doctor's graduation date
GRAD_UNIV - doctor's university
MEMO - (empty column)

Patient table

Each tuple in the patient table shows the demographics information for a single patient. The following list shows all the columns contain within the patient table:
PID# - primary key, patient's identification number
L_NAME - patient's last name
F_NAME - patient's first name
PRE_NAME - patient's previous name, if any (empty column)
STREET - street name of patient's home
POBOX - patient's post office box (empty column)
CITY - city that patient lives in
PROV - province that patient lives in
POSTCODE - postal code of patient's home
PHONE - patient's phone number

SEX - patient's gender
MARITAL_STATUS - patient's marital status (divorced, engaged, married, separated, single)
DATE_OF_BIRTH - patient's date of birth
AGE - patient's age
STATUS - is the patient alive (0) or dead (1)
DATE_OF_DEATH - the date of death for this patient (empty column)
C_I_D - (empty column)
C_T_D - (empty column)
INDICATORS_INDIAN - (empty column)
INDICATORS_SAP - (empty column)
RELIGION - (empty column)

Allergy table

Each tuple in the allergy table shows a single allergy suffered by a patient. The following list shows all the columns contain within the allergy table:
PID# - foreign key, the patient this tuple involves
ALLERGY - an allergy that affects this patient
DIAG_DATE - the date this allergy was diagnosed

Insurance table

Each tuple in the insurance table shows the insurance information for a patient. The following list shows all the columns contain within the insurance table:
PID# - foreign key, the patient this tuple involves
INSURANCE# - patient's insurance number
INSURANCE_TYPE - insurance company providing insurance (SaskHealth, Crown Life)
I_DATE - date insurance took affect

Weight and height table

Each tuple in the weight-height table shows weight and height values for a patient. The following list shows all the columns contain within the weight_height table:
PID# - foreign key, the patient this tuple involves
WEIGHT - patient's height in meters
HEIGHT - patient's weight in kilograms
WH_DATE - measurement date

Visit table

Each tuple in the visit table describes a hospital visit by a patient. The following list shows all the columns contain within the visit table:
PID# - foreign key, the patient this tuple involves
REF_PHYSICIAN - foreign key, referring doctor for this patient
ATT_PHYSICIAN - foreign key, attending doctor for this patient
ADMISSION_DATE - date patient was admitted
DISCHARGE_DATE - date patient was discharged
LENGTH_STAY - patient's length of stay in hospital
TYPE_ADMISSION - the doctor specialty this patient was admitted under

TYPE_VISIT - (empty column)
MEDICAL_SERVICE - (empty column)
NURSING_STATION - (empty column)
OUTPATIENT_CLINIC - (empty column)
DIAGNOSIS - (empty column)
ISOLATION - (empty column)
LEFT - (empty column)