An HPSG Lexicon for a Physical Activity Database

Shinta I. Mayasari

Technical Report CS-95-09
September, 1995

Department of Computer Science
University of Regina
Regina, Saskatchewan, CANADA
S4S 0A2

ISSN 0828-3494
ISBN 0-7731-0310-4

Abstract

This paper represents a documentation of the design and implementation of a lexicon for a natural language front-end to a physical activity database. The lexicon uses Head-driven Phrase Structure Grammar (HPSG) as its underlying grammatical theory. The database semantics of words and phrases is represented in HPSG-style feature structure and a Semantic Principle for analysing the meanings of constructed phrases is proposed.

The Attribute Logic Engine (ALE) is used as a lexical representation language. A method to implement HPSG lexicon in ALE and to effectively organize the lexicon is described. The method allows ease of modification and maintenance of the lexicon.

1 Introduction

A lexicon contains information about words used in a natural language system such as a natural language interface to a database. Its importance has become more central with the emergence of lexically-oriented theories such as *Head-driven Phrase Structure Grammar* (HPSG) [14, 15]. These theories have a few grammatical rules and hence place the burden of encoding a vast majority of linguistic information in the lexicon. This lexicalist tradition is suitable to a natural language interface which strives for portability [4]. The interface can be attached to different application domains by focusing on the customization of the lexicon and performing little or no modification on the grammar rules.

The increased complexity of the lexicon calls for a structural approach of lexical representation. Several methods that have been proposed are single inheritance, multiple inheritance, default inheritance, and lexical rules [5, 6, 14, 16, 17]. The goal of this project is to apply such structural or hierarchical methods in constructing a lexicon for a natural language interface to a physical activity database.

We document the design and implementation of an HPSG lexicon for a physical activity database. The rest of this paper is organized as follows. In Section 2 we describe the chosen domain and state the problems to be addressed in constructing the lexicon. The approach taken in solving these problems is discussed in Section 3. The HPSG theory that underlies the lexicon is reviewed and necessary modifications are highlighted. In this section we also describe the lexical representation language selected, that is the ALE [2] system, and show how ALE's devices are used for developing the physical activity lexicon. We conclude in Section 4 by summarizing our accomplishments and identifying areas of future research. The program listing and a sample run can be found in Appendix A and B, respectively.

2 Issues in Constructing a Lexicon

When developing a natural language interface to a database, one needs to define a sublanguage to be recognized by the system. Formulating the sublanguage requires knowledge about the underlying domain. The specification of the project domain considered is given in Section 2.1. Expressions in the sublanguage are interpreted in the context of this domain. The linguistic

problems encountered when interpreting such expressions are described in Section 2.2. Implementational issues related to the underlying grammatical theory are discussed in Section 2.3.

2.1 Domain Specification

The domain selected for this project is a physical activity database. The database is concerned with the measurement of body's strength, stamina, and flexibility for the purpose of assessing one's fitness and wellness [8]. The domain description is largely based on the information obtained from the Dr. Paul Schwann Applied Health and Research Center, in particular, from its database and report forms.

The physical activity domain consists of three major subdomains, namely **Body**, **Muscle**, and **Cardiac**. Figure 1 represents a taxonomy of the domain. The **Body** subdomain refers to

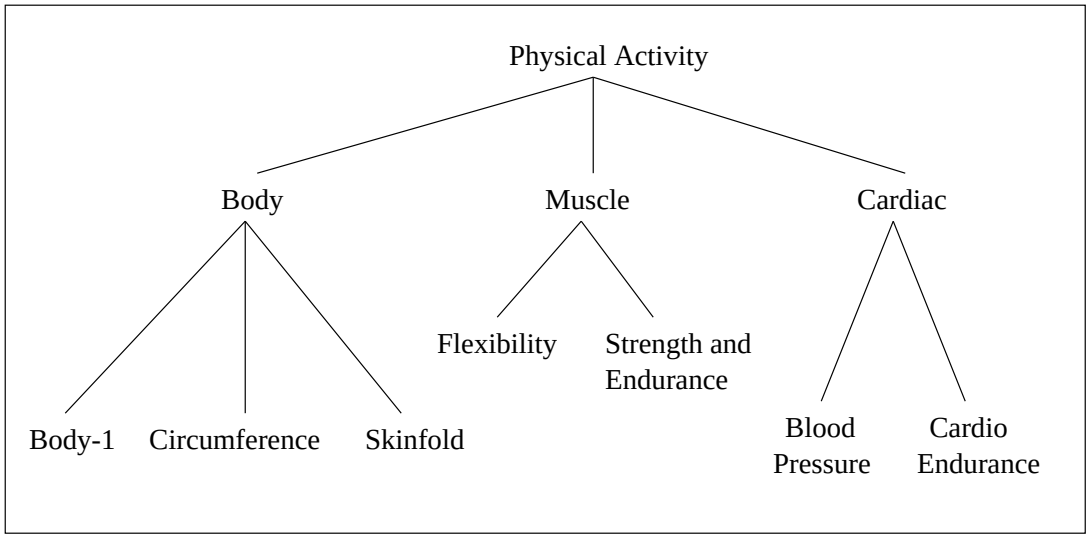

Figure 1: The Taxonomy of Physical Activity Domain.

the assessment of fat/nonfat components of the human body. This subdomain is further divided into three subareas according to the type of tests performed. These subareas include **Body-1**, **Circumference**, and **Skinfold**. The **Muscle** subdomain is classified into two subareas corresponding to **Muscular Flexibility** and **Muscular Strength and Endurance**. The **Cardiac** subdomain is comprised of two subareas, **Blood Pressure** and **Cardiovascular Endurance**. The physical activity domain can be represented by the database scheme depicted in Figure 2.

In this particular case, each specific subarea in the domain hierarchy corresponds to a specific table in the database. All of the subareas have two common attributes, CLIENT_ID and TDATE. The CLIENT_ID attribute refers to the identity of a client, whereby the TDATE attribute represents the date when a test is performed. In addition to these common attributes, each subarea has its own subdomain-specific attributes.

Relation	Attributes
body_rel	CLIENT_ID, TDATE, HEIGHT, WEIGHT, BD_MASS, RISK_B
circ_rel	CLIENT_ID, TDATE, CHEST, WAIST, HIP, THIGH, WST_HIP_RATIO, RISK_C
skin_rel	CLIENT_ID, TDATE, TRICEPS, BICEPS, CALF, SUBSCAP, ILLIAC, SKINFOLDS, TWO_SKINFOLDS, RISK_S
mflex_rel	CLIENT_ID, TDATE, SITNREACH, RANK_FSR, SHOULDER, RANK_FSF, FHIP, RANK_FHF
mstren_rel	CLIENT_ID, TDATE, GRIP, RANK_SGS, PUSH_UP, RANK_SPU, SIT_UP, RANK_SSU
card_rel	CLIENT_ID, TDATE, TTYPE, TIME, MAXVO2, RISK_CRD
bloodp_rel	CLIENT_ID, TDATE, SYSTOLIC, DIASTOLIC, RISK_BP
clien_rel	CLIENT_ID, NAME, B_DAY, GENDER

Figure 2: The Database for Physical Activity Domain.

The **Body-1** subdomain has one table, *body_rel*, containing four subdomain-specific attributes, HEIGHT, WEIGHT, BD_MASS (height and weight ratio), and RISK_B which refers to a risk factor determined from the height and weight ratio. The risk factor divides clients into those with low, medium, and high factors of developing diseases, such as coronary heart disease.

The **Circumference** subdomain has six specific attributes, CHEST, WAIST, HIP, THIGH, WST_HIP_RATIO (waist to hip ratio), and RISK_C (risk factor based on the result of girth measurements).

The **Skinfold** subdomain consists of eight domain-specific attributes, namely BICEPS, TRICEPS, CALF, SUBSCAP, ILLIAC, SKINFOLDS (the total of all skinfold measurements), TWO_SKINFOLDS (the sum of measurements performed on subscap and illiac crest), and RISK_C (risk factor based on the results of skinfold test).

The **Flexibility** subdomain has six domain-specific attributes, SITNREACH (result from sit and reach test), RANK_FSR (flexibility ranking based on the sit and reach test), SHOULDER (result from shoulder flexion test), RANK_FSF (rank of shoulder flexion test), FHIP (result from forward hip flexion test), and RANK_FHF (rank of forward hip flexion test). The **Strength and Endurance** subdomain also has six specific attributes, GRIP (result from hand grip strength test), RANK_SGS (rank of grip strength), PUSH_UP (result from push-up exercise), RANK_SPU (rank of push-up test), SIT_UP (result from sit-up exercise), and RANK_SSU (rank of sit-up exercise).

The **Cardio Endurance** subdomain consists of four domain-specific attributes, TTYPE (test type), TIME (time required to complete the test), MAXVO2 (maximum oxygen uptake), and RISK_CRD (risk factor based on the oxygen uptake). The simplest subdomain is **Blood Pressure** which has three specific attributes, SYSTOLIC, DIASTOLIC, and RISK_BP (risk

factor based on the blood pressure test result).

The database also contains an additional table which refers to client information. It has three attributes, NAME, B_DAY (birth date), and GENDER which discriminates clients based on their gender.

2.2 Natural Language Problems

The lexicon constructed for this project covers a set of English expressions used when querying the physical activity database (Figure 2). For the interface to "understand" these queries, the lexicon must have knowledge about their meanings. A problem peculiar to a database interface is that the semantics of words and phrases are not always coherent with natural language semantics. It is therefore crucial to analyze their meanings in terms of the database schema.

Furthermore, general linguistic problems such as ambiguities are still present even though the scope of the language is restricted to a narrow domain. Two problems addressed in this project include:

- **Ambiguity**
  A word or a phrase may refer to different tables or different attributes. For example, the word *hip* has different meanings in the body and muscle subdomains. It refers to either the HIP attribute in the *circ_rel* table or the FHIP attribute in the *mflex_rel* table.

- **Nominal Compound** Most of the expressions used in querying the database are compound noun phrases. Some examples are

  - *body mass measurement for Smith*
  - *Smith's muscular flexibility assessment*
  - *overweight male clients*
  - *female clients with high risk of developing coronary heart disease*

  Nominal compounds present a well-documented set of problems [7, 11, 13] such as:

  1. The interpretation of a nominal compound is not always a result of combining the meanings of its constituents as is commonly done in a strict compositional approach. In some cases, the meaning of a two-word phrase is formed by one meaning overwriting the meaning of the other word. For example, in our domain, the meaning of the phrase *body height* is that of the word *height*.
  2. Nominal compounds exhibit structural ambiguity, which often coincides with semantic ambiguity. For instance, a machine may have difficulties assigning the correct interpretation for the phrase *Smith's blood pressure test*.
  3. Nominal compounds appear in paraphrase relations with other structures. For example, *the result of blood pressure test for Smith* and *Smith's blood pressure test result* have the same meaning. Analysis of such phrases should yield the same interpretation.

2.3 Implementation Issues

For the lexicon to be useful for something other than parsing, it should at least contain syntactic and semantic knowledge [7]. In the HPSG formalism, this knowledge is represented in the form of feature structures or attribute-value matrices (AVMs). However, in a natural language interface to a database, the meanings of words and phrases often depend on the structure of the database itself. Thus, the problem is to find a feature structure representation of the database semantics and incorporate it into the HPSG feature structure such that syntactic and semantic interpretation can be analyzed in parallel.

Another issue is related to the fact that in unification formalisms lexical entries often contain complex and redundant information. Two major sources of redundancy are vertical redundancy and horizontal redundancy [14]. Vertical redundancy refers to words that share some common properties. For example, English has many simple intransitive verbs like *run*. Clearly, it would be impractical to represent each such word as a separate AVM. The second source of redundancy refers to groups of words that are related based on some recurrent patterns. For example, every English verb has a third-singular finite form which is usually formed by adding *s* to the base form. Exhaustive listing of all inflectional forms of every word would yield a very large lexicon. It is thus important to be able to capture linguistic generalizations and to reduce both types of redundancies in the lexicon.

HPSG theory employs two methods for organizing its lexicon; these are *multiple inheritance hierarchy* and *lexical rules* which will be described in the following section. The implementation of these methods, however, requires a tool that can hierarchically structure lexical entries in a way that allows ease of lexicon maintenance and modification.

3 Approach to the Lexicon Implementation

The physical activity lexicon is developed within the HPSG formalism. This grammar has also been used in the design and implementation of SystemX [4]. Our work differs from SystemX in that we use ALE [2] as our lexical representation language, whereby SystemX uses HPSG-PL [10].

There are several reasons for choosing ALE. First, ALE offers a variety of devices for organizing the lexicon including *type hierarchy*, *macro*, *lexical rule*, and *definite clause attachment*. These methods can be used to eliminate vertical redundancy and horizontal redundancy. In contrast, HPSG-PL uses macro as its sole mechanism for structuring the lexicon. The HPSG-PL macro eliminates vertical redundancies, but not horizontal redundancies. Furthermore, although in practice macro is a powerful device, it does not have a theoretical status in the HPSG formalism [5, 12].

HPSG-PL is very grammar-specific: it is dedicated to the 1987 version of HPSG. Any changes, such as changes on the attributes of a feature structure or on grammar principles, require modification of the underlying HPSG-PL implementation. ALE, on the other hand, allows users to concentrate on the high level devices that are provided by the system in order to experiment with the different grammars.

Finally, the use of ALE facilitates a comparative study between the two development tools.

The results can be used to provide feedback for the improvement of both tools, as well as SystemX.

3.1 The HPSG Formalism and Its Modification

Head-driven Phrase Structure Grammar (HPSG) is a unification-based formalism [13, 16] which uses feature structure as its primary method for representing knowledge. It is head-driven; it stipulates that each phrase contains a certain word, the *head*, which is centrally important in the sense that the head determines many syntactic properties of the entire phrase.

The primary operation in HPSG is unification of feature structures. Two feature structures, A and B, can be unified into another structure C that contains all information in both A and B, provided that A and B are consistent. If A and B contain conflicting information, the unification fails. Unification is closely related to the notion of subsumption [14, 17], or information containment.

3.1.1 Knowledge Representation

A feature structure, or attribute-value matrix (AVM) is a collection of attribute-value pairs which describes the properties of a linguistic object. The theory requires that each feature structure must have a type which describes the kind of object the structure is modeling, and that the kind of attributes that can appear in a feature structure is determined by its type.

In the present formulation of HPSG [15], a feature structure has at least two attributes, the PHON attribute which is used to represent phonology, and the SYNSEM attribute which encodes the syntactic and semantic properties of an object.

The value of SYNSEM is a feature structure containing local (LOC) and nonlocal (NONLOC) information. NONLOC is concerned with the analysis of unbounded dependency phenomena; its discussion is beyond the scope of this project. LOC information is divided into category (CAT) and content (CONT) attributes.

The CAT attribute contains syntactic properties such as category (whether an object is a verb or a noun), subcategorization requirements, etc. These properties are captured in CAT's two attributes: HEAD and SUBCAT. The value of HEAD is a feature structure whose attributes depend largely on the category of the object represented. For example, a head noun has attribute CASE which is not appropriate for a verb. One attribute common for all head categories is MOD (modified) which is used to analyze head/adjunct structure. Finally, the SUBCAT value is a list of feature structures required for a word or a phrase to become saturated.

The CONT attribute encodes some semantic information about the linguistic object in question. Its content depends on the domain in which it operates. For example, in HPSG [14, 15], the meaning of the word *want* is expressed as a relation with two arguments: one represents the individual who wants something and one represents the object that is wanted. In the AVM representation of *want* shown in Figure 3, such a meaning is encoded in the path SYNSEM|LOC|CONT, specifically in the attributes REL (relation), WANTER and WANTED. Note that the values of WANTER and WANTED are annotated with indices $\boxed{1}$ and $\boxed{2}$, re-

spectively. These indices indicate that their values are the same as the values of the CONT attributes of the noun phrases within the SUBCAT attribute; that is, they denote structure sharing. In other words, the SYNSEM|LOC|CAT|SUBCAT attribute captures the fact that the verb *want* requires an NP subject whose meaning must unify with the value of SYNSEM|LOC|CONT|WANTER and an NP object whose meaning must unify with the value of the attribute SYNSEM|LOC|CONT|WANTED in order to become a complete phrase.

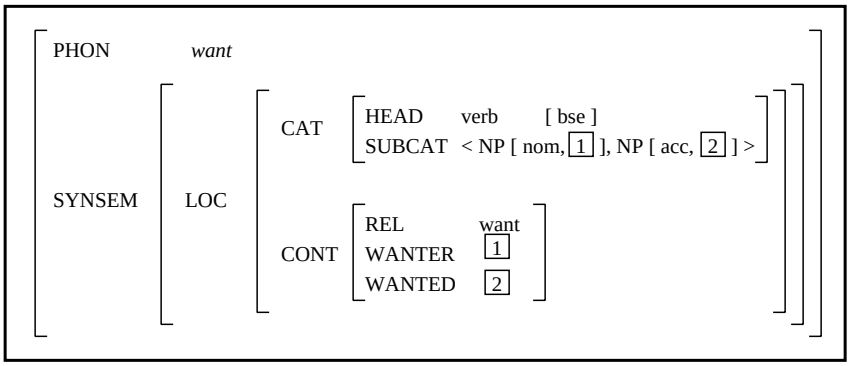

Figure 3: Example of AVM

A semantic representation as given in Figure 3, however, is not entirely suitable for the physical activity domain since words have specific database meanings. Following [4, 11], database information is represented in the CONT feature by using notation that is closely tied to the structure of relational databases. In this project, CONT's value is a feature structure containing two attributes: QUERY and RESTR. QUERY represents the database and contains table (TABLE) and subtables (SUBTABLE) information. TABLE information is partitioned into REL which has an atomic value representing the table name, and COLUMNS feature which takes as its value a feature structure containing database attributes and their values. The SUBTABLE feature is used if information from other tables is required. If the join of tables is necessary, it is indicated by structure-sharing of the variable in the column's attribute in which the join is made. Figure 4 illustrates a simple representation of database semantics for the word *height*.

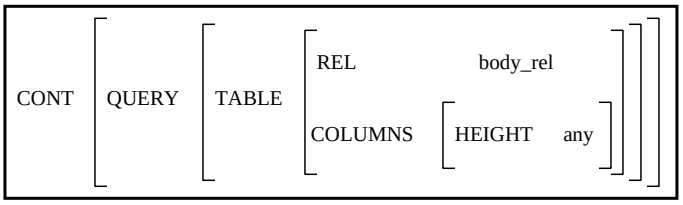

Figure 4: Semantic Representation of *height*

The RESTR (restriction) attribute is introduced inside the CONT feature as a part of the solution to handle the decompositional problems discussed in Section 2.2. The permissible values of this attribute are the following atoms: *join*, *overwrite*, and *none*. This attribute is used by the Semantic Principle which will be described in Section 3.1.2 below.

3.1.2 Grammar Rules and Principles

Grammar rules and principles govern the way constituents are combined to form larger phrases. HPSG has only a few grammar rules that are applied to five different classes of syntactic constituents: *head*, *complement*, *adjunct*, *marker* and *filler*. Most of the English expressions in our sublanguage have either head/complement structures or head/adjunct structures. We therefore focus on the handling of these types of structures. Four rules that govern these types of structures are described below: two rules are applicable to uninverted head/complement structures, one deals with inverted head/complement structures, and one rule licenses the head/adjunct structures. These rules are Schema 1, Schema 2, Schema 3, and Schema 5 of [15] respectively.

1. Schema 1, shown in Figure 5, licenses those phrases which consist of a non-lexical head with its final complement. It corresponds to the rules commonly expressed in the forms:

$$S \rightarrow NP, VP \qquad NP \rightarrow DET, N'$$

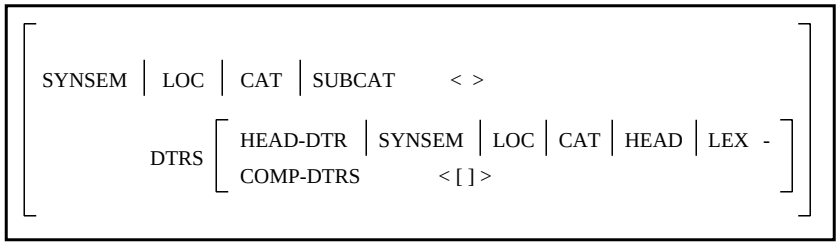

Figure 5: Schema 1

2. Schema 2, shown in Figure 6, licenses phrases which have a lexical head daughter and zero or more complement daughters, and which have satisfied all the subcategorization requirements except its final complement (usually, the subject).

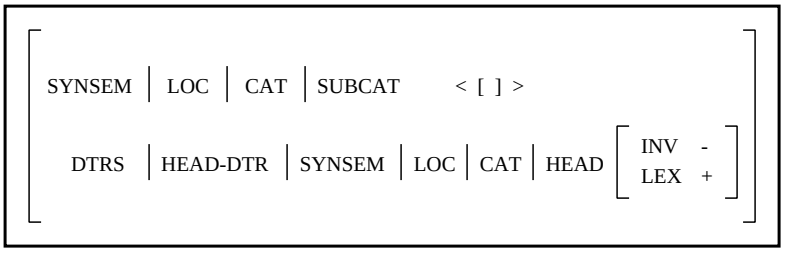

Figure 6: Schema 2

Schema 2 subsumes a vast array of traditional rules such as:

$$VP \rightarrow V, NP \qquad PP \rightarrow P, NP$$
$$VP \rightarrow V, PP \qquad VP \rightarrow V, NP, NP$$

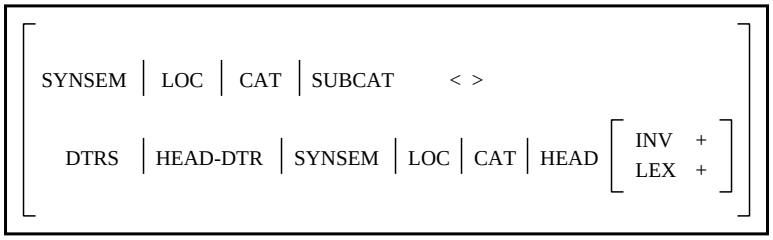

Figure 7: Schema 3

3. Figure 7 shows the rule for inverted clauses such as given in the following sentence *Can I see the result?* .

4. The rule illustrated in Figure 8 is used to form a phrase from an adjunct and a head that it selects. Traditional rules captured by Schema 5 include:

$$\text{N} \rightarrow \text{ADJ, N} \qquad \text{N'} \rightarrow \text{NP, N'}$$
$$\text{N'} \rightarrow \text{N', PP} \qquad \text{VP} \rightarrow \text{VP, PP}$$

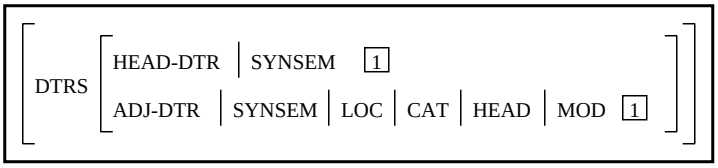

Figure 8: Schema 5

Each of the rules above is processed in conjunction with a set of grammar principles such as *Head Feature Principle*, *Subcategorization Principle*, and *Semantic Principle*. The first two principles of HPSG are adopted from HPSG without any modification. The Semantic Principle must be modified to handle the database semantics of words and phrases.

Head Feature Principle

The HPSG Head Feature Principle states that the value of a HEAD feature of a headed phrase is structure-shared with the HEAD value of its head daughter.

Subcategorization Principle

In any headed phrase, the SUBCAT value of its head daughter is the concatenation of the phrase's SUBCAT list with its complement daughters.

Semantic Principle

The Semantic Principle of HPSG [15] states that the CONT of a phrase is inherited from the semantic head of the phrase. In the case of a head/complement structure, the semantic head coincides with the syntactic head, whereby in a head/adjunct structure, the semantic head is that of the adjunct. However, for our database query expressions, this principle needs to be further specified. This is described below.

In a complex constituent, each daughter may have a different role in determining the database semantics of the mother. This role restricts the daughter's contribution toward the overall meaning and is encoded in the RESTR attribute. These restrictions include

- *none*:
  Words that do not contribute to the database semantics have this restriction. Examples of such words are verbs such as *want*, *generate*, etc.

- *overwrite*:
  This restriction is for words whose meanings may overwrite the meanings of the other words. For instance, the word *weight* takes precedence over the word *body*. Thus, in our domain, the phrase *body weight* has the same meaning as the word *weight*.

- *join*:
  This restriction applies for a class of words which, when combined with other words, cause the join of two or more tables. Examples of such words in our domain are proper names.

The algorithm to compute the database semantics is as follows:

1. If adjunct/complement daughter has restriction of type *join*, then ensure that its TABLE and the head's TABLE share some column name. Mother inherits its SUBTABLE from the adjunct daughter's TABLE and SUBTABLE and its TABLE from the head daughter's TABLE. Mother also inherits its restriction type from its adjunct/complement.

2. If adjunct/complement daughter is of type *overwrite*, check the restriction of the head daughter.

   - If the head daughter has *overwrite* restriction, then mother inherits its QUERY information (i.e., table, subtable, and restriction) from the head daughter.
   - If the head daughter has the *none* restriction, then mother inherits its QUERY from the adjunct/complement.

3. Otherwise, the mother's QUERY is the unification of the QUERY of all daughters (recursively combining tables and subtables information).

Figure 9 shows the result of applying grammar rules and principles to the analysis of *Smith's height*. This phrase is analysed as a head/complement structure having two daughters: *Smith's* (the complement daughter) and *height* (the head daughter). The phrase *Smith's* is itself a head/complement daughter in which *s* is the head and *Smith* is the complement daughter. The constituents *Smith*, *height*, and *s* have restrictions *join*, *overwrite*, and *none*, respectively. The RESTR attribute of the phrase *Smith's* is inherited from its complement daughter (which is *join*) according to the first Semantic Principle, whereby the QUERY is obtained by joining the tables of both daughters. Since *s* does not have a database meaning, the resulting QUERY contains the same information as the complement daughter's QUERY. The phrase *Smith's height* is then constructed from the constituents *Smith's* and *height*. Since the complement daughter, *Smith's* has a restriction *join*, the first Semantic Principle is again applied. The

RESTR attribute of the phrase *Smith's height* is *join*, whereby the QUERY is obtained by joining the tables of both daughters. The join of the tables is indicated by structure-sharing [1] in the CLIENT_ID attribute.

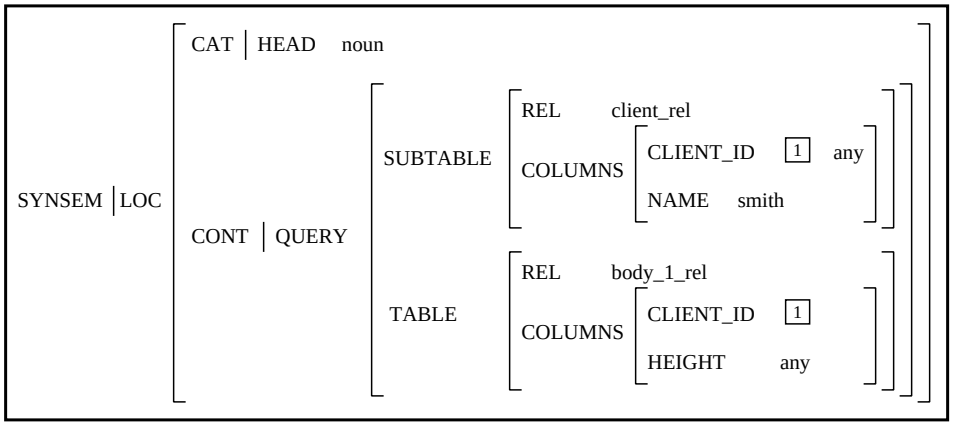

Figure 9: Analysis of *Smith's height*

3.1.3 Lexicon Organization

HPSG relies on a highly structured lexicon to capture linguistic generalizations. It employs two primary methods of organizing the lexicon, namely,

Multiple Inheritance
: Lexical entries are classified on the basis of their common syntactic and/or semantic properties. Each class may have more than one superclass and it inherits all properties of its superclasses.

Lexical Rules
: Lexical rules provide a mechanism for dealing with redundancies such as the kinds of inflexional morphology used for word classes, derivational morphology as found with suffixes and prefixes, and so on.

These methods are applied to the physical activity lexicon by using devices available in the ALE system which will be discussed further in the following section.

3.2 Implementing the Physical Activity Lexicon in ALE

The Attribute Logic Engine (ALE) system consists of a bottom-up chart parser [3, 7, 9] and a compiler. As stated in the beginning of Section 3, the ALE system offers a variety of methods for structuring the lexicon. In this section, we review these methods and show how they can be used to construct the physical activity lexicon.

[1] Structure-sharing implies token-identity and is denoted by the index 1.

3.2.1 Type Hierarchy

As in HPSG, ALE uses *typed feature structure* as its representational scheme. That is, every feature structure must have a type associated with it. Each type must specify which features it can be defined for, and which types of values such features can take. These types are arranged in an inheritance hierarchy, whereby constraints and properties on supertypes are inherited by their subtypes. Every type must be declared as a subtype of ALE's supertype, `bot` which is the unique most general type.

The physical activity lexicon has three different type declarations: system, syntax, and semantic declarations. The system type declaration defines the hierarchical relation of basic data types. At present, we have two such types: boolean and list. Boolean is declared as a simple type hierarchy consisting of `bool` and its two subtypes: `minus` and `plus`. The list declaration is slightly more complicated. Figure 10 shows the inheritance hierarchy for list.

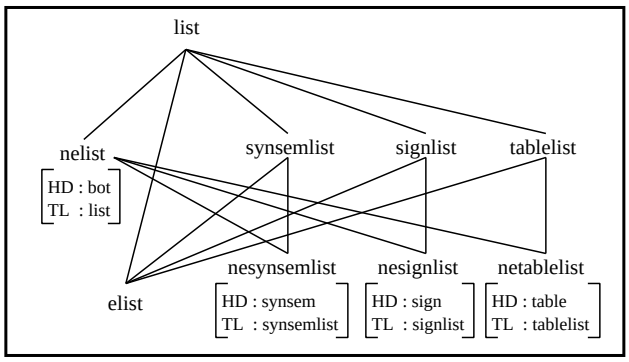

Figure 10: Hierarchy for List Definition

List can be either empty (`elist`) or nonempty (`nelist`). A nonempty list has two features representing the head and the tail of a list. The tail must be a list whereby the head can be anything. However, for our HPSG implementation, the SUBCAT feature has as its value a list of objects of type `synsem`. In such a case, the head of the list must be more restricted to an object of type `synsem`. Similar problems are found for the COMP_DTRS feature whose value is a list of objects of type `sign`, and SUBTABLE feature whose value is a list of objects of type `table`. To impose such restrictions, we introduce different types of list: `synsemlist`, `signlist`, and `tablelist`. An example of how to construct such a hierarchy in ALE is given in Figure 11.

The syntactic declaration consists of the type hierarchy related to the syntactic information of linguistic objects. This declaration is, in general, domain independent and therefore can be ported to different applications. The following list shows part of the basic syntactic types based on HPSG description [15]. Note that the notation $\sqsubseteq$ indicates the subsumption relation ($A \sqsubseteq B$ means that A is more general than B).

- `sign` $\sqsubseteq$ `word`, `phrase`.
  `sign` has an attribute SYNSEM which is inherited by both `word` and `phrase`.

- `synsem`

```
list sub [elist, nelist, synsemlist, signlist, tablelist].
   elist sub [].
   nelist sub [nesynsemlist, nesignlist, netablelist]
        intro [hd: bot,
               tl: list].
   synsemlist sub [elist, nesynsemlist].
      nesynsemlist sub []
           intro [hd: synsem,
                  tl: synsemlist].
   signlist sub [elist, nesignlist].
      nesignlist sub []
           intro [hd: sign,
                  tl: signlist].
   tablelist sub [elist, netablelist].
      netablelist sub []
           intro [hd: table,
                  tl: tablelist].
```

Figure 11: Specification of List Hierarchy in ALE

- `loc`
- `cat`
- `head` $\sqsubseteq$ `subst`, `funct`.

The rest of the hierarchy under `head` is depicted in Figure 12. Further divisions of the syntactic

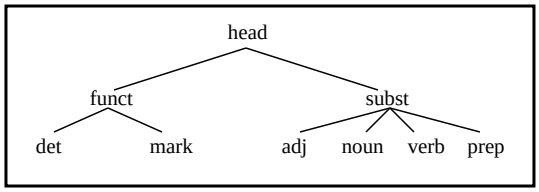

Figure 12: Syntactic Hierarchy under Head

heads are performed in the macro hierarchy.

The semantic declaration contains the hierarchical definition of meanings related to a specific domain. Some semantic types are:

- `cont`
- `query`
- `table`
- `columns` $\sqsubseteq$ `height`, ...
- `restr` $\sqsubseteq$ `none`, `overwrite`, `join`.

Figure 13 illustrates part of the physical activity hierarchy under columns. In this hierarchy, objects of type `pa` inherit all properties of `body-2`, `muscle`, and `cardiac`. The semantic type

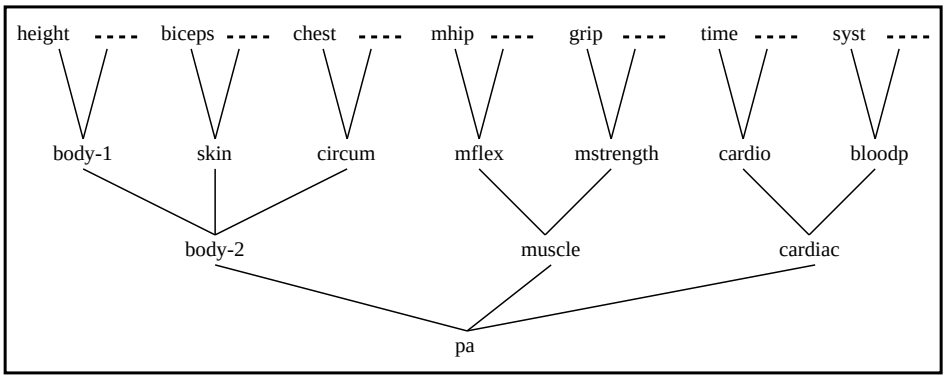

Figure 13: Part of the Semantic Hierarchy

hierarchy is essential for assigning structure to nominal compounds, and to control syntactic and semantic ambiguities. Words are classified according to their semantic properties. The ALE parser uses this type information to determine possible structures for a particular phrase. For example, the semantic classification determines that the phrase *cardiovascular circumference* is meaningless in the current domain.

3.2.2 Macro Hierarchy

The idea of macro originates from the template mechanism in PATR-II [7, 16]. Macro allows users to define a description once and later use the definition to abbreviate the description. An example of macro taken from the lexicon implementation is as follows:

```
head_s(X) macro (loc:cat:head:X).
```

A call to the macro, e.g., `@head_s(verb)` is equivalent to writing: [LOC|CAT| HEAD verb]. Macros are useful to maintain consistency of lexical entries, to provide easy access to information deeply embedded in description, and to facilitate maintenance of lexical entries.

Note that other macros can be called in the definition of a macro, allowing hierarchies of definitions to be formed. In our implementation, we use macro hierarchies for further classifying linguistic objects. For example, verbs have several subclasses such as intransitive, transitive, and ditransitive verbs. These verbs have the same category (i.e., same HEAD feature values) and differ in the number and type of complements they require. As described earlier, this subcategorization requirement is encoded in the path SYNSEM|LOC|CAT|SUBCAT whose value is a list of objects of type `synsem`. The length of the list varies depending on the kind of verbs; e.g., for intransitive verbs, the list has only one element whereas for transitive verbs, the list contains two elements. The difference between these subclasses of verb is implemented by specifying the classes in the macro definition as shown in the following example [2]:

```
intrans macro (@vp, @subcat([@np_s])).
trans   macro (@vp, @subcat([@np_s, @obj_np_s])).
ditrans macro (@vp, @subcat([@np_s, @obj_np_s, @obj_np_s])).
```

The specification of individual verbs such as *walk*, *eat*, *give* can be simply stated as follows:

```
walk ---> word, @intrans.
eat  ---> word, @trans.
give ---> word, @ditrans.
```

All lexical entries in our lexicon are specified using calls to the lexical macros as was previously described. This approach simplifies the specification of lexical entries and reduces the amount of redundancies in the lexicon.

3.2.3 Rules and Principles

In ALE, grammar rules are specified in the form of rewrite rules. Some general constraints on rules (e.g., grammar principles) can be expressed by attaching definite clauses on the right hand side of such rewrite rules.

To illustrate how HPSG schemas can be encoded in ALE's rules, let us examine Schema 2 depicted in Figure 6. This schema licenses head/complement structures consisting of a lexical head daughter and zero or more complement daughters. It must be used in conjunction with grammar principles which constrain the applicability of the rule. Figure 14 shows an implementation of this schema in ALE. We first describe the ALE notation used in such rule. The operator `===>` separates the mother of a phrase structure rule from the daughters. Each daughter is prefixed with `cat>`. A list of complement daughters can be specified with a prefix `cats>`. The `goal>` operator is used to attach definite clauses which constrain the rule applicability.

In the specification of Schema 2 given in Figure 14, the mother category is a phrase which is still missing its subject. The head daughter is a word which subcategorizes for its subject and other complements. These complement daughters are not known until run-time. Since various lexical items may have different number of complement daughters, e.g., zero for proper nouns, one for intransitive verb, etc, `cats>` is required in order to match the actual number of complements at run-time. Furthermore, the complement daughters information are determined by

[2] Notice that the definitions are expressed in terms of calls to other macros, e.g., @vp.

```
schema2 rule
(Mother, phrase, @subcat([SubjSynsem]))
===>
cat> (HeadDtr, word, @subcat([SubjSynsem|CompSynsem])),
goal> (synsem_to_phrase(CompSynsem, Comp),
       is_not_subject(Comp, SubjSynsem)),
cats> Comp,
goal> (dtrs(Mother, @head_comp(HeadDtr, Comp)),
       principles(Mother, HeadDtr, Comp, CompSynsem)).
```

Figure 14: Implementation of Schema 2 in ALE's Phrase Structure Rule

imposing constraints on their types (by the goal `synsem_to_phrase/2`) and their subject condition (by calling `is_not_subject/2`). The first constraint is necessary because the complements specified on the head's SUBCAT are of types `synsem`, whereby complement daughters should be of type `phrase`. The second constraint ensures that the complement daughter is not the missing subject. The call to `dtrs/2` in the last goal serves to build up a parse tree. The applicability of the rule is further constrained by calling the grammatical principles `principles/4`. Figure 15 shows the definition of the definite clause `principles/4`.

```
principles(Mother, Head, Comps, CompsSynsem) if
        (head_feature_principle(Mother, Head),
         subcat_principle(Mother, Head, CompsSynsem),
         semantic_principle(Mother, Head, Comps)).
```

Figure 15: Definition of `principles/4`

As shown, the `principles/4` is a conjunction of the Head Feature Principle, Subcategorization Principle, and Semantic Principle. It can be called from all rules by specifying the correct arguments which would be required by the principles.

ALE offers a method to implement HPSG lexical rules which can be used to capture patterns of redundancies that obtain among classes of lexical entries [16], including inflectional morphology, derivational morphology, passivization, nominalization, etc. This method is useful for eliminating horizontal redundancy. For example, instead of exhaustively listing every inflectional form of every verb, one can just specify its base form. As in grammar rules, definite clauses can be used to express constraints on lexical rules. At present, only one lexical rule is implemented in our lexicon. The rule transforms a base verb into a finite verb.

In summary, the physical activity lexicon has four HPSG grammar rules, three principles, and one lexical rule. Analyses of some sentences produced by the grammar and the lexicon can be found in Appendix B.

4 Concluding Remarks

We described the design and implementation of an HPSG lexicon for physical activity database. The prototype lexicon is implemented in ALE because ALE offers powerful mechanisms such as macro, type inheritance, lexical rule, and definite clause attachment. These mechanisms are very useful for organizing an HPSG lexicon which often contains large and complex information. We used macro hierarchy extensively in order to keep consistency of lexical entries and to provide ease of maintenance and modification.

In the physical activity sublanguage defined, some words and phrases have database meanings. In some cases, the meaning of a constructed phrase is non-compositional. To overcome these problems, we proposed a feature structure representation of semantics similar to the representation employed in [4, 11] which is closely tied to the structure of the database and we introduced a Semantic Principle which is used for analysing the meaning of a constructed phrase.

The prototype lexicon has four HPSG grammar rules and employs three principles, the Head Feature Principle, the Subcategorization Principle, and the Semantic Principle. The rules and principles are used to license the admissibility of phrases in the physical activity sublanguage which are classified as head/complement or head/adjunct structures.

The HPSG rules are implemented using ALE's phrase structure rules. The prefix `cat>` used in ALE's rules enforce the ordering of the head/complement (head/adjunct) daughters. Consequently, Schema 5 of HPSG which is intended to capture the generalize head/adjunct structure must be implemented into two rules: one rule licenses the case where adjunct precedes the head, and another rule licenses the case where adjunct follows the head. Grammar principles are implemented as definite clause attachment to the phrase structure rules. This approach has a clear advantage in that it provides ease of modification of any of the principles while keeping the principles consistent across the rules.

The work presented herein leaves many areas open for future study. Some of them have been identified:

- The prototype lexicon was developed as part of the development of a natural language interface to the physical activity database. It is capable of analysing the admissibility of an input string with respect to the chosen domain. Further works need to be done to integrate the lexicon with the remainder of the interface modules such that the resulting system will generate answers to queries.

- Further development of the lexicon to allow morphological information to be represented effectively.

- Extending the coverage of the grammar and lexicon for more complex structures such as conjunction.

We are currently in the process of developing the natural language interface to a physical activity database which uses the prototype lexicon described in this paper. The initial prototype is expected to complete in early Fall 1995.

ACKNOWLEDGEMENT

The author is a member of the Institute for Robotics and Intelligent Systems (IRIS) and wishes to acknowledge the support of the Networks of Centres of Excellence Program of the Government of Canada, the Natural Sciences and Engineering Research Council, and the participation of PRECARN Associates Inc. The author is grateful to Dr. Nick Cercone for valuable comments, suggestions and support. Thanks are also due to Dr. F. Popowich and Dr. D. Fass for providing information on resources and literatures.

References

[1] Calder, J. and Humphreys, K., *Pleuk User Manual*, University of Edinburgh, Edinburgh, Scotland, 1994.

[2] Carpenter, B. and Penn, G., *The Attribute Logic Engine User's Guide version 2.0*, Carnegie Mellon University, Pittsburgh, PA, 1994.

[3] Earley, J., "An Efficient Context-free Parsing Algorithm", *Communications of the ACM*, vol. 13, pp. 94-102, 1970. Reprinted in Grosz et al. (1986) pp. 25-33.

[4] Cercone, N., McFetridge, P., Popowich, F., Fass, D., Groeneboer, C., and Hall, G., *The SystemX Natural Language Interface: Design, Implementation and Evaluation*, Technical Report CSS-IS TR 93-03, Simon Fraser University, Burnaby, BC, 1993.

[5] Daelemans, W., Gazdar, G., and De Smedt, K., "Inheritance in Natural Language Processing", *Computational Linguistics*, vol. 18, no. 2, 1992.

[6] Flickinger, D. and Nerbonne, J., "Inheritance and Complementation: A Case Study of Easy Adjectives and Related Nouns", *Computational Linguistics*, vol. 18, no. 3, 1992.

[7] Gazdar, G., and Mellish, C., *Natural Language Processing in Prolog: An Introduction to Computational Linguistics*, Addison-Wesley Publishing Co., Menlo Park, CA, 1989.

[8] Hoeger, W. W. K., *Lifetime Physical Fitness and Wellness*, Morton Publishing Company, Englewood, Colorado, 1986.

[9] Kay, M., *Algorithm Schemata and Data Structures in Syntactic Processing*, Research Report, Xerox PARC, 1980. Reprinted in Grosz et al. (1986) pp. 35-70.

[10] Kodrič, S., Popowich, F., and Vogel, C., *The HPSG-PL System Version 1.2*, Technical Report CSS-IS TR 92-05, Simon Fraser University, Burnaby, BC, 1992.

[11] McFetridge, P., Popowich, F., and Fass, D., " An Analysis of Compounds in HPSG for Database Queries", to appear in *First International Workshop on Applications of Natural Language to Databases*, France, June 1995.

[12] Meurers, W. D., "On Implementing an HPSG Theory", In E. W. Hinrichs, W. D. Meurers, and T. Nakazawa: *Partial-VP and Split-NP Topicalization in German - An HPSG Analysis and its Implementation*, Arbeitspapiere des SFB 340 Nr 58, Universität Tübingen, Tübingen, Germany, 1994.

[13] Pereira, F. C. N. and Shieber, S. M., *Prolog and Natural Language Analysis*, CSLI, Stanford University, CA, 1987.

[14] Pollard, C., and Sag, I., *Information-based Syntax and Semantics, Volume 1: Fundamentals*, CSLI, Stanford University, CA, 1987.

[15] Pollard, C., and Sag, I., *Head-driven Phrase Structure Grammar*, CSLI, Stanford University, CA, 1992.

[16] Shieber, S. M., *An Introduction to Unification-based Approaches to Grammar*, CSLI, Stanford University, CA, 1986.

[17] Vogel, C., Popowich, F., and Cercone, N., *Inheritance Reasoning for Head-driven Phrase Structure Grammar*, Simon Fraser University, Burnaby, BC, 1991.

Appendix A

Program Listing

Appendix B

Sample Run

Program listing and sample run can be obtained by sending electronic mail to
mayasari@cs.uregina.ca