

# Linguistic Knowledge Representation with Deductive Inheritance Network

Li Li                  Barrett R. Bryant

Department of Computer Information and Sciences  
University of Alabama at Birmingham  
1300 University Blvd.  
Birmingham, AL 35294-1170  
{li4, bryant}@cis.uab.edu

## Abstract

This paper presents a knowledge representation model for natural language processing. A multiple default inheritance network implemented as a deductive system is proposed to efficiently encode large and complex unification-based lexicons. Derivational inheritance is offered to achieve maximal reduction of redundancy. The explicit and implicit overwriting mechanism along with the *disjunctive union* operation allow generalization with exceptions to be handled in a perspicuous and economical way. The hierarchical lexicon is compiled into a canonical lexicon to accelerate the parsing algorithm.

## 1 INTRODUCTION

Inheritance networks with such attractions as parsimony, ease of maintenance, uniformity and modularity, have recently received much attention in computational linguistics and natural language processing [1]. Most of the work in this area has used inheritance networks as a means to build large and complex lexicons for unification-based analyses.

Kameyama [2] proposes a multiple inheritance network to support Categorical Unification Grammar descriptions of multilingual nominal translations.

Moens [3] uses *sorts* to build a type hierarchy for Unification Categorical Grammar. All these networks are monotonic in which a class has to inherit all information from its parent.

Francois Andry [4] developed DIALEX, an inheritance-based tool that facilitates rapid construction of linguistic knowledge bases. Domain semantics is encoded as part of the DATR grammar formalism and a compilation procedure is used to generate compact lexicons.

In the framework of PATR-II, Graham Russell [5] proposed a hierarchical lexicon that relies on the total ordering of class lattices to resolve potential contradictions of default inheritance.

Norman Fraser [6] employs propositional logic as a uniform formalism to specify both linguistic and conceptual knowledge for Word Grammar. The default inheritance is achieved through explicit overwriting.

The monotonic inheritance hierarchy proposed by Remi Zajac [7] is based on constraint satisfaction and is able to encode HPSG grammars for both analysis and generation.

Van der Linden [8] exploits the inheritance in a Lambek Categorical Grammar to facilitate lexical disambiguation and preference. The inheritance is maintained between individual words rather than classes and attributes can be “borrowed” among lexical definitions.

Within the framework of unification-based grammar, we have developed a knowledge representation capable of incorporating domain semantics into formal grammars while maintaining high degree of extensibility and transportability. The inheritance network is implemented in the LIFE programming language [9, 10], which is an integration of logic, functional and object-oriented programming paradigms. LIFE supports the manipulation of attribute-value matrices with sort hierarchies in a logic programming environment. Since the inheritance is built on top of LIFE, it is very simple and flexible.

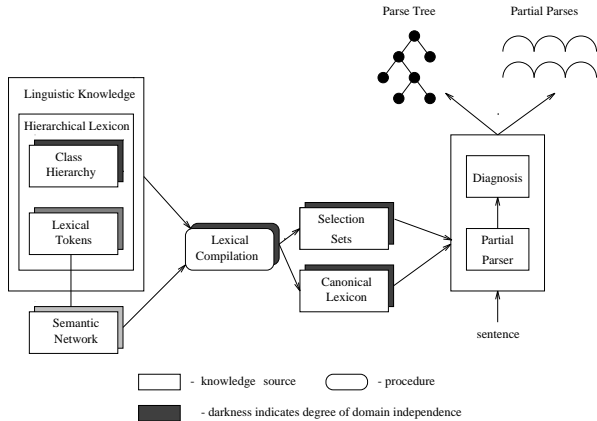


Figure 1: Components of the Parsing Model

## 2 PARSING MODEL

The knowledge base, which consists of a semantic knowledge base and a linguistic inknowledge base, is part of a robust parsing model for understanding casual English sentences. The components of the parsing model are illustrated in Figure 1.

The semantic knowledge base is a sortal hierarchy that encodes the necessary conceptual information for selectional restrictions and semantic interpretation.

The linguistic knowledge is a hierarchical lexicon maintained by a class definition language with the following features:

1. The inheritance network is translated into a deductive rule system, which allows classes to be declared in any order, including in cycles, and refined incrementally through unification.
2. Inheritance is allowed in two dimensions: from parent to child classes and from child to sibling classes. Roughly speaking, a child class inherits all the properties of a parent, whereas a sibling class derives new properties from its sister class. This expansion not only provides a more economical representation but also coincides with many linguistic phenomena.
3. Multiple default inheritance is achieved with explicit and implicit overwriting mechanisms so that generalization with exceptions can be handled in an elegant and perspicuous way. The disjunctive interpretation of multiple inheritance provides a

practical solution to the potential contradictions in this type of network.

4. In addition to the commonly used equality constraints, procedural constraints are also employed to facilitate semantic assignments.

The category hierarchy and dictionary are compiled into a canonical lexicon to reduce access time.

## 3 HIERARCHICAL LEXICON

The purpose of the hierarchical lexicon is to describe appropriate grammar rules for each type of words economically through inheritance. The basic unit in the inheritance network is *class* of the following format:

```
class Name
  head [H, C1, ..., Ck]
  rule [Rule1, ..., Rulem]
```

where *head* attribute defines the intrinsic feature *H* of the class by a set of *head constraints*  $\{C_i\}$  and *rule* attribute specifies in PATR style [11] the grammar rules shared by this class of words. Grammar rules written in this formalism contains a “context-free” portion and a set of predicates that constrain the constituents described. For instance, the context-free rule postulated for the English sentence  $s \rightarrow np vp$  would be expressed as

$$\begin{aligned}
 X_0 : s &\rightarrow X_1 : np \quad X_2 : vp \\
 &head(X_0, X_2) \\
 &agree(X_1, X_2) \\
 &X_1 = X_2.subj \\
 &X_0.type = X_1.type
 \end{aligned}$$

Unifications are expressed by the equal sign (=). Variables may be tagged with types by the colon (:) notion and attribute paths are formed by the project (.) function of LIFE. Polymorphic predicates like *head*(X, Y) and *agree*(X, Y) denotes sets of equality constraints applicable to different categories. These constraint *macros* make grammar definitions more compact and informative.

Parent-child inheritance and child-sibling derivation are established by the *isa* and *from-with* clauses respectively as follows:

```

class C1
  head Head-list
  rule Rule-list
  isa C2

class C1
  head Head-list
  rule Rule-list
  from C2
  with Rule

```

Multiple inheritance can be asserted using class disjunctions: `class C ... isa {C1; ...; Ck}`.

Conventional inheritance is a special case of derivation. This is motivated by the fact that many distinct classes, such as various types of verbs, share much common information in the rule patterns. Instead of introducing extra superclasses to store the common properties, we simply “borrow” them from other classes, which is more economical and flexible.

Overwriting is realized by the destructive assignment ( $X \leftarrow Y$ ) in LIFE. Unlike the mechanism proposed by Shieber [12], we allow any value to be overridden rather than just atomic ones.

With these facilities, inheritance can be created vertically and horizontally as demonstrated in the definitions of `iv` (intransitive verb) (Figure 2), `tv` (transitive verb) (Figure 3), and `aux` (auxiliary verb) (Figure 4).

Equality constraints are not sufficient to resolve semantic assignments in incremental parsing. For instance, the case role of an adjunctive PP depends on the verb it modifies:

The letter is received [ $PP=agent$  by the school].  
 The car is parked [ $PP=loc$  by the school].

Our solution is to attach to the grammar rules of prepositions a predicate `roleof(X,Y)` that searches the case frame of X for roles that match Y.

```

X1:vp → X1 ____ X2:np
      roleof(X1,X2)

```

While the equality constraints can be evaluated in lexicon compilation, the non-equality restrictions are executed during the parsing.

Lexical entries in the dictionary include only those features specific to the words in the formula `word(Form,Class,Concept)`. For example, the verb *teach*, an instance of `tv` and `dtv` classes, is defined below with its case frame:

```

word ([teach, +=es, taught, taught, +=ing],
      {tv; dtv}, teach)
case-frame (teach, agent⇒[teacher, by],
           patient⇒[course], recipient⇒[student, to])

```

The attributes in a lexical instance may be different from those of its class, thereby causing *implicit* cancellations. An example is the definition of exceptional

```

class iv
  head [ X0
        X0.agree.case = nom
        X0.syn.aux = -
        X0.syn.inv = -
        X0.syn.type = fin
        X0.subj = X0.sem.agent
      ]
  rule [ [ X1:vp → ____
          head (X1, X0)
        ] ]
  isa cat.

```

Figure 2: `iv` class defines the syntactic and semantic features of a verb.

```

class tv
  head [X0]
  rule [ [ X1 → ____ X2:np
          role (X1,patient,X2)
        ] ]
  from iv
  with X1:vp → ____ .

```

Figure 3: `tv` is derived from `iv` by adding a following NP as the *patient* role.

```

class aux
  head [ X0
        X0.syn.inv ← +
        X0.syn.aux ← +
      ]
  rule [ [X1:vp → ____ X2:vp
        X2.syn.aux = -
        control (X0, X2)
        syn-head (X1, X0)
        sem-head (X1, X2)
      ] ]
  isa tv.

```

Figure 4: `aux` is a `tv` with additional features to form tensed phrases and questions. The default features are overwritten accordingly.

modal verb *might*, which can not be inverted to form questions <sup>1</sup>:

word (might, aux, syn⇒[inv⇒ -])

## 4 DEDUCTIVE INHERITANCE

The inheritance network is implemented as a deductive rule system of Horn clauses. Each class  $C_i$  is represented as a predicate  $class(C_i, [H_i|E_i], R_i)$  and inheritance is realized as the deduction of class predicates based on a set of validness conditions.

An inheritance relation between subclass  $C_1$  and class  $C_2$  is valid if the following conditions hold in the given order:

1. a clause  $class(C_2, [H_2|E_2], R_2)$  exists and its head constraint set  $E_2$  is satisfiable.
2. The head features of two classes are unifiable, i.e.  $H_1 = H_2$ .
3. The head constraint set  $E_1$  is satisfiable.
4.  $R \in R_2$  where  $R$  is the rule in the with clause of a derivational inheritance.

These conditions guarantee that class  $C_1$  inherits proper properties from  $C_2$  should conflicts arise between them. A typical definition `class C1 head [H1|E1] rule R1 isa C2` will be translated into the Horn clause

$$class(C_1, H_1, R_3) \leftarrow class(C_2, H_2, R_2), \\ valid(H_1, H_2, E_2), \quad R_3 = R_1 \cup R_2.$$

where `valid` is the predicate that tests the validness conditions.

The information in the *head* of a class may be instantiated, expanded or altered by a lexical token. This process is carried out by a noncommutative destructive operation called *default union*, which gives priority to one of the mutually conflicting terms.

The default union of two feature structures X and Y, denoted as  $X \sqcup_d Y$ , is the unification of X and Y except that the conflicting attributes of Y (if any) are replaced by the corresponding attributes of X. For instance, we have

$$X : \begin{bmatrix} f \Rightarrow a \\ g \Rightarrow b \\ h \Rightarrow c \end{bmatrix} \sqcup_d Y : \begin{bmatrix} f \Rightarrow a \\ g \Rightarrow c \\ e \Rightarrow d \end{bmatrix} \Rightarrow Y : \begin{bmatrix} f \Rightarrow a \\ g \Rightarrow b \\ h \Rightarrow c \\ e \Rightarrow d \end{bmatrix}$$

Multiple inheritance is obtained by traversing the class lattice using depth-first search, or the standard Prolog backtracking mechanism. Potential contradictions arising from the interaction of overwriting and inheritance are resolved by a operation called *disjunctive union*. In other words, if any attribute is inherited through two paths that lead to mutually contradictory information, two distinct class predicates would be derived and merged by the disjunctive union.

In general, the disjunctive union of two feature structures X and Y, denoted by  $X \uplus Y$ , is the unification of X and Y except that all the conflicting paths (if any) are replaced by the disjunctions of incompatible values. For example, we have

$$\begin{bmatrix} f \Rightarrow a \\ g \Rightarrow b \\ h \Rightarrow c \end{bmatrix} \uplus \begin{bmatrix} f \Rightarrow a \\ g \Rightarrow c \\ e \Rightarrow d \end{bmatrix} \Rightarrow \begin{bmatrix} f \Rightarrow a \\ g \Rightarrow \{b; c\} \\ h \Rightarrow c \\ e \Rightarrow d \end{bmatrix}$$

The operation satisfies the following properties:

1. commutative:  $X \uplus Y \equiv Y \uplus X$
2. associative:  $X \uplus (Y \uplus Z) \equiv (X \uplus Y) \uplus Z$
3. subsumption:  $X \uplus Y \sqsubseteq X, X \uplus Y \sqsubseteq Y$

The following non-linguistic example illustrates the application of disjunctive union to inheritance with contradictions.

```
class ca head [H:x] rule [ [s→ a(H)] ].
class cb head [H:y] rule [ [s→ a(H)] ].
word (ab, {ca;cb}).
```

From the `ca-ab` path, token `ab` inherits the rule `s→ a(x)`, while via the `cb-ab` path, it obtains `s→ a(y)`. The two rules can be packed into `s→ a({x;y})` by disjunctive union.

We adopted this practical approach because the disambiguation among grammar rules can be and should be resolved in the parsing phase.

## 5 LEXICAL COMPILATION

In general, inheritance will increase the access time of a lexicon as extra searching is added to each

<sup>1</sup>We do not accept: “Might I go?” or “What might he do?”

lookup step. The problem is tackled by the *lexical compilation* process which compiles the hierarchical lexicon into a *canonical lexicon* of the form `entry(Token,Classes,Rules)`. For instance, the lexical entry without attributes for `teach` is

```
entry(teach, [tv, dtv], [vp→_____ np,
                    vp→_____ np np, vp→ np pp])
```

During the compilation, the alternative inheritance paths of a token are exhausted by backtracking and disjunctive union is employed to collect the rules with the same context-free portions across different classes. The time complexity of the compilation is thus  $O(K * C^2 * R^2)$ , where  $K$  is the total number of tokens in the lexicon,  $C$  is the maximal number of classes traversed for a token and  $R$  is the maximal number of rules a token may possess. Consider the average number of classes and rules are rather small, this process is efficient.

The proposed inheritance network has been used in building a lexicon for some common English constructs. The current linguistic coverage includes wh-questions, yes/no-questions, passive sentences, relative clauses, and various verb phrases

For a small lexicon, the canonical lexicon is three times larger than the hierarchical lexicon. This difference on one hand demonstrates the significant reduction of redundancy by inheritance and on the other hand shows the necessity of more sophisticated compilation techniques, such as bit-coding [4].

## 6 CONCLUSION

We have described a knowledge representation model that makes extensive use of inheritance networks to encode linguistic information and domain semantics in an efficient way. The deductive inheritance and predicate constraints offer a simple and flexible formalism for specifying large and complex unification-based lexicons. Our future work is to extend the linguistic coverage and explore more advanced compilation methods. The task of lexical extension is made easy due to highly modularized knowledge representation.

## References

- [1] Walter Daelemans and Koenraad De Smedt. Inheritance in natural language processing. *Computational Linguistics*, 18(2):205–218, 1992.
- [2] Megumi Kameyama. Atomization in grammar sharing. *Proceedings of the Association for Computational Linguistics Conference*, pages 194–203, 1988.
- [3] Marc Moens et al. Expressing generalization in unification-based grammar formalism. *Proceedings of 4th Conference of the European Chapter of ACL*, pages 194–203, 1989.
- [4] Francois Andry et al. Making DATR work for speech: Lexical compilation in sundial. *Computational Linguistics*, 18(3):245–267, 1992.
- [5] Graham Russel et al. A practical approach to multiple inheritance for unification-based lexicons. *Computational Linguistics*, 18(3):311–337, 1992.
- [6] Norman M. Fraser and Richard A. Hudson. Inheritance in word grammar. *Computational Linguistics*, 18(2):1–158, June 1992.
- [7] Remi Zajac. Inheritance and constraints-based grammar formalism. *Computational Linguistics*, 18(2):159–182, June 1992.
- [8] Erik-Jan van der Linden. Incremental processing and the hierarchical lexicon. *Computational Linguistics*, 18(2):219–238, 1992.
- [9] Hassan Ait-Kaci and Patrick Lincoln. LIFE: A natural language for natural languages. Technical Report ACA-ST-074-88, Micro-electronics and Computer Technology Corporation, 1989.
- [10] Hassan Ait-Kaci et al. *The Wild LIFE Handbook (prepublication edition)*. Digital Equipment Corporation, Paris Research Laboratory, March 1994.
- [11] Stuart M. Shieber. *Constraint-Based Grammar Formalisms*. The MIT Press, 1992.
- [12] Stuart M. Shieber. *An Introduction to Unification-Based Approaches to Grammar*. CSLI Lecture Note 4, 1986.