A Prolog morphological analyzer for Portuguese

Jacques Wainer *Alexandre Farcic*

Relatório Técnico DCC–94-09

Outubro de 1994

A Prolog morphological analyzer for Portuguese

Jacques Wainer Alexandre Farcic

Abstract

This paper describes a morphological analyzer for Portuguese written in Prolog. It understands about the standard declinations for noun, adjectives, and regular verbs and it also understands about prefixes and suffixes. The system is not only able to recognize a word if its root is in the dictionary, but also to infer (or rather guess) the lexical classes of words whose roots are not in the dictionary by using its knowledge of declinations and suffixes.

1 Introduction

Prolog is a programming language that is very well suited for the development of natural language processing programs, specially the syntactic analysis components of these programs [PS87]. But the morphological analysis has never received much attention, mainly because English, which is the language most used word-wide in natural language processing systems, is, from a morphology point of view, very simple.

In a natural language processing system the morphological analysis is the step in which words are analyzed and categorized in terms of different lexical classes, and the particular attributes for that lexical class are determined. Thus, the Portuguese word *construíssemos* should be classified as first person of the plural of the imperfect tense of the subjunctive mode of the verb *construir* (to build), and that determination would be a typical task of a morphological analyzer.

Strictly speaking, for implementing a functioning natural language processing systems there is no need for a morphological analysis component: all declinations of the verb, for example *construir*, could be entered in the lexicon of the system, which would just consult its lexicon to determine the lexical category and the respective attributes of the word *construíssemos*. Of course that would be wasteful since all sixty two declinations of the verb would have to be entered into the lexicon. On the other hand with a morphological analyzer which embodies the rules for the verb's declinations, only the statement that the verb *construir* is regular would suffice.

The goals of this research were:

- to construct a morphological analyzer for the Portuguese language.
- to construct it in standard Prolog so it could be easily incorporated into existing and future Prolog based NLP systems.

- to populate its dictionary with all lexical classes that are closed, that is, classes whose elements are fixed for the Portuguese language. Thus the system's dictionary already includes all pronouns, prepositions, conjunctions, determiners of the Portuguese language, as well as all declinations of important irregular verbs like *ser* (to be), *estar* (to be), and so on.

Because of the second goal above, we had to make some design decisions concerning accents and the special character cedilla ("ç"). Since standard Prologs do not accept extended ASCII code, which includes all accents and non-standard characters in Portuguese, we decided not to include them at all in the words treated by the system: accents should be removed altogether and cedilla should be coded as the letter "c". The two consequences of such decision are that this morphological analyzer cannot be used as a spell checker, and there are an extra level of ambiguity that should be treated by the program that uses this system. For example the word *e* is analyzed as the conjunction *e* (and) and also as the third person, singular of the present tense of the indicative mode of the verb *ser* (to be), which would be graphed as *é*. More severe than that are the words that are disambiguated exactly by the presence of the accent, for example the pair *cantará* (to sing–third person singular, future tense of the indicative mode) and *cantara* (to sing–third person singular *plus-que-parfait* tense of the indicative mode).

On the other hand, the following where **not** part of the system's goals:

- to serve as a computational implementation of some morphology theory (for example [Kar83, RRBP92]). Theories of morphology try to explain all morphology processing, for all languages, in terms of some computational machinery (finite state machines or two-levels finite state machines [Kos83, Kos85, Gaz85]) applied to strings of letters or possibly phonemes. This project does not implement any of those theories. It was meant as a rather straight forward implementation of the morphological rules of Portuguese (as they appear in for example [dC80]) as Prolog clauses.

- to implement a spell checker. This system cannot be considered a spell checker for many reasons: it does not include accents and cedilla, it can makes wrong but reasonable inferences about the spelling of word, its dictionary contains only words that are important from a morphological point of view (the closed lexical classes) and it does not contain any word in the open lexical classes (nouns, adjectives, regular verbs), and finally it is probably not efficient enough as a spell checker.

In its current version the system does not deal with contractions, like *delas*, which is the contraction of the preposition *de* and the pronoun *elas*. Also the system assumes that all letters of the words are lower case. Future versions of the system would probably include: a better interface for the user to enter new words in the dictionary (the details of how new word are entered is not discussed in this paper); and a way of coding accents into words that extends the current scheme (that is, if an accent is entered the system would consider it, but if no accent is present the system assumes the current scheme).

The current version of the morphological analyzer is available for anyone interested by contacting the first author. The comments in the code contain a short user's manual which includes topics not covered in the paper, like how to use the code, how to add new words to the dictionary, and what are the syntax of the information the analyzer returns.

2 On declinations and affixes

Declinations are sequence of letters that change the variable attributes of a lexical class. In this paper we will distinguish between fixed and variable attributes (or features) of a lexical class. Fixed attributes are attributes that cannot be modified by declinations of the "original word" or root. Thus the lexical class of nouns has the fixed attribute of **gender** (masculine or feminine), and the variable attributes of **number** (plural or singular) and **degree** (neutral, augmenting degree or diminishing degree). Thus the word *arvore* (tree) is feminine, and that is fixed, but is also singular and neutral. These last two attributes may be changed by declinations: the word *arvorezinhas* (little trees) is feminine, plural and diminishing degree.[1]

The fixed attributes must stored with the root of the word in the dictionary, since there is no way the system can infer them. On the other hand, the system embodies the rules for determining the variable attributes given the declinations present in the word. The concept of root in our system is just the "most common" declination of the word: for adjectives the root is the masculine, singular, neutral declination, for nouns the singular form, and for verbs the impersonal infinitive.

Some lexical classes have only fixed attributes. Prepositions and conjunctions have only one attribute which we will call **type**. Most adverbs also have only one attribute (again called **type**) except for adverbs ending in *-mente*, which can have a variable **degree** attribute (for example, *belamente* (beautifully) and *belissimamente* (very beautifully)).

Nouns have the fixed attribute of **gender**, and the variable attributes of **number** and **degree**. Adjectives have the variable attributes of **gender**, **number** and **degree**. Verbs have the variable attributes of **person**, **number**, **tense** and **mode**.

Different than declinations, affixes, which includes prefixes and suffixes, are not used to alter variable attributes of lexical classes, but to alter the meaning of a word, and possibly its lexical class.

Prefixes are segment of words that are concatenated at the beginning of a word to change its meaning without changing its lexical class. For example, the prefix *contra-*, which have the general meaning of against, can be concatenated with the verb *por* (to put) resulting in the verb *contrapor* (to contrast). Suffixes are word segments that are concatenated to the end of a word in order to change its meaning and possible its lexical class. Thus the suffix *-mente* can be concatenated with adjective *belo* (beautiful) resulting in the adverb *belamente* (beautifully).

The probably longest Portuguese word *anticonstitucionalissimamente* has the prefix *anti-* applied to the adverb ending with the suffix *-mente* in the augmenting degree, which

[1]For some nouns (nouns for animals and professions) the gender could be considered as a variable attribute. One could consider the noun *gata* (cat – fem.) to be derived though a declination from the noun *gato* (cat – masc.), but not in our system. Our system does not consider gender a variable attribute in any case and thus if only *gato* is in the dictionary the system would not recognize *gata*.

in turn was derived from the adjective *constitucional*, which in turn was derived from the noun *constituição* by the suffix *-al*, which in turn was derived from the verb *constituir* by the suffix *-ção*.

3 Levels of Analysis

Conceptually system provides five different levels of morphological analysis, which are discussed below. From the user point of view though, there are only three levels, since lookup and grounded declination are combined into a single level, and the non-grounded declination and non-grounded affixation are also combined.

3.1 Lookup

The first level of analysis, **lookup**, is just a look up function. It searches the dictionary for the word, retrieve the corresponding lexical information, that is, the lexical class and fixed attributes, and returns it. In the present version of the system, the dictionary must be in the program's memory in the form of Prolog clauses. Of course, this is unacceptable but for small dictionaries. A more elaborate solution for the lookup level of analysis would be to search for the word in some external file, probably cashing the results in Prolog's internal clause memory, or to store the words in memory but in a very compact way [LK93].

The lookup level of analysis is performed for the lexical classes of prepositions, determiners, pronouns, conjunction, and so on. Some of these classes, like preposition and conjunction, have only fixed attributes, whereas the others, like determiners, pronouns and so on, are so common in Portuguese sentences that, for efficiency reasons, we decided to store in the dictionary all their possible declinations. Thus the pronouns *ele, ela, eles, elas* (he, she, they–masculine gender, they–feminine gender) are all stored in the dictionary. Furthermore, all declinations of irregular verbs must be present in the system's dictionary.

3.2 Grounded declination

The **grounded declination** level of analysis understands about the standard declinations of lexical classes like regular verbs, nouns, adjectives and adverbs. Given a word that is not in the dictionary, the system will check if its ending corresponds to some known declination, determine the lexical class and the values of the variable attributes that corresponds to the declination, remove the declination in order to obtain the root of the word, check if the root is in the system's dictionary, and retrieve the fixed attributes information.

For example, if the word *amarelo* (yellow) is in the dictionary as an adjective[2] , a grounded declination analysis would recognize the words *amarelo, amarela, amarelos, amarelas* (yellow–masc, yellow-fem, yellow-masc. pl., yellow–fem. pl.).

The morphological analysis of a noun would determine the word's number and degree, and retrieve the root and it's gender from the dictionary. Thus if the noun *livro* is stored in the dictionary as a masculine noun, the system would recognize words like *livrões* (big books) as the augmenting degree, plural of *livro*. But the system would also recognize the

[2] *Amarelo* is also a masculine noun and that must be stored in the dictionary also.

incorrect Portuguese words *livãos* and *livrães* as the augmenting degree, plural of *livro*. The problem is that the plural construction rule for words ending in *-ão*, as in *livrão* (big book), specifies that the three terminations *-ãos, -ões* and *-ães* are possible. In most of the cases only one of the terminations is really correct, but of course the system has no way of knowing which (but some words like *aldeão* (villager) allow for all the three terminations for their plural).

As we mentioned above, the system considers gender a fixed attribute of nouns, and therefore it would not recognize *gata* as a valid declination of *gato*.

The morphological analysis of adjectives would determine number, degree and gender, and retrieve the root from the dictionary. Thus, if the word *amarelo* is declared in the dictionary as an adjective, the system would recognize words like *amarelas* (yellow–fem. pl.), or *amarelissimos* (very yellow–masc. pl.).

Finally, adverbs ending in *-mente* allow for a variable degree attribute (for example *belamente* and *belissimamente*), which is also recognized.

3.3 Grounded affixation

The **grounded affixation** (suffixation and prefixation) level of analysis would also attempt to remove prefixes and suffixes of the word in order to obtain a root that is stored in the dictionary. As we mentioned above, prefixes are segment of words that are appended to the beginning of the word and do not change the word's lexical category. Suffixes are segment of words that are appended to the end of a word, and usually change the original word's lexical category. Thus if a word, after the removal of the known declinations, is not found in the dictionary, the system would try to further remove prefixes and suffixes in order to obtain a known root. Of course the system would not just return the inferred lexical class of the original root, but also report on the process of affixation that transformed the root into the original word.

The grounded affixation analysis works as follows: if the word has not been recognized by the grounded declination analysis, the system would try to match the beginning of the word with a know prefix, remove it an apply the grounded declination on the remaining word (which would be of the same lexical class as the original word). If that fails, the system would try to match the termination of the remaining word with a known suffix. Suffixes can be classified based on the lexical class transformation they result. For example the suffix *-mente* can only transform an adjective into an adverb. Thus if the *-mente* was found at the end of the word, the system infers that that word was an adverb, remove the suffix, and try to find the remainder morpheme in the dictionary (in fact for words ending in *-mente* the system will first try to remove degree declinations since such adverbs have a variable degree attribute). This process can repeated at most three times since we could not find a word that is the result of more than three suffixations. For example, the analysis of the word *afirmativamente* (affirmatively), would return:

```
adverb(suffixation(mente,
                   adjective(suffixation(tivo,
                                         verb(afirmar)))))
```

The prefixes known to the system are:

> *a-, anti-, hiper-, hipo-, meta-, pro-, ante-, com-/con-, contra-, de-, des-, extra-, in-/im-/i-, inter-/entre-, per-, pre-, re-, retro-, sub-/sus-, super-/sobre-, trans-*

Suffixes can be classified in terms of what lexical class transformation they perform. For example, the following suffixes transform adjectives into nouns.

> *-itude, -dade, -idão, -eza/-ez, -ura, -ismo*

For example, the adjective *alto* (high) can be transformed by the use of the suffix *-itude* into the noun *altitude* (height).

Suffixes that transform verbs into nouns are:

> *-mento, -ância/-ência, -ança/-ença, -ante/-ente/-inte, -dor, -ação*

For example, the pairs *jogar* (to play) and *jogador* (player); or *crer* (to believe) and *crença* (belief).

Suffixes that transform verbs into adjective are:

> *-tório, -avel, -ivel, -tivo*

Some examples: *afirmar* (to affirm) and *afirmativo* (affirmative); and *durar* (to last) and *duravel* (durable).

Suffixes that transform noun into adjective are:

> *-ento, -onho, -tico, -ista, -oso, -udo, -al*

Some examples: *barro* (mud) and *barrento* (muddy); and *riso* (laugh) and *risonho* (smiley).

Finally, there is only one suffix that transform an adjective into an adverb, the suffix *-mente*, as in *facil* (easy) and *facilmente* (easily).

The main problem with the grounded affixation analysis derives from the fact that the morphological rules are underspecified, which is also the explanation for the system acceptance of the incorrect plurals of *pão*, discussed in the previous section. In the case of grounded affixation the problem is more severe, since there are no rules that constrain which suffix ((from a particular class) can be applied to which root. Thus the system would recognize non-existing words like *bonitude* and *estudador*, (if *bonito* and *estudar* are present in the dictionary) because the rules that allow recognizing such words are the same ones that allow the system recognize the correct Portuguese words *infinitude* (infiniteness) and *jogador* (player).

3.4 Non-grounded declination

The **non-grounded declination** analysis is similar to the grounded declination analysis but the requirement that the word's root is stored in the system's dictionary is dropped. A competent Portuguese speaker would recognize the word *assevandijamos* as the second person plural of the present tense of the indicative mode of a possible verb *assevandijar* (which in fact is a Portuguese verb [dH80] albeit an obscure one). Thus just by using the declination rules a competent speaker is able to infer (or rather guess) the lexical class of the words and its attributes. The non-grounded declination analysis would perform this processing.

Of course the use of non-grounded analysis in a natural language processing system is limited. Since the root had to be guessed, there is no semantic information attributed to it and so, if the system that is using the morphological analyzer is trying to understand the meaning of the sentence, the information returned by the non-grounded analysis is useless. But of course, some level of understanding is always possible even without specific semantic information about the root (after all people does understand sentences that contain unknown words). Some other system may realize it does not know the meaning of *assevandijar* and ask the user! A less ambitious use of the non-grounded analysis would be a grammar checker, since it does not have to "understand" the words, but just check if they obey the grammar rules.

3.5 Non-grounded affixation

Non-grounded affixation is the extension of non-grounded declination to suffixes and prefixes. Thus, after the removal of known declinations, prefixes and suffixes, the system would guess the root.

The problem of the underspecification of the morphological rules is even greater in the non-grounded analysis (both declination and affixation) than for grounded analysis since the requirement that the root must be in the dictionary is dropped. Let see what happens when the word *misturaveis* (mixable–pl.) is analyzed at the non-grounded declination level. The system first assumes that the word is a noun, then an adjective, then a verb, and finally an adverb (by design). Thus, the system would analyze *misturaveis* in order as:

- the plural of the (non-existing) nouns *misturavel* and *misturavil* because the rules that responsible for such inferences also allow for the correct plural of the words *missel* and *util*.
- the singular of the noun *misturaveis* since there are singular nouns ending in "s" (*pubis* for example).
- the plural of the adjectives *misturavel* (a correct guess) and *misturavil*.
- as the second person plural of the imperfect part tense of the indicative mode of the verb *misturar* (a correct guess)
- and other combinations of tenses and mode of the non exiting verbs *misturavar*, *misturaver*, *misturavir* and *misturaveir*

Thus eventually the correct guesses are obtained, although the system would also make a lot of reasonable but incorrect guesses.

The non-grounded affixation will also eventually generate the correct guesses among some more bizarre ones. For example the non-grounded affixation analysis of *interminavel* (never ending) generates the correct analysis of an adjective derived from the verb *terminar* (to finish) by prefixation of *in-* and suffixation of *-vel*. But it also analyzes the word as the adjective derived from the verb *minar* (to mine) by prefixation of *inter-* and suffixation of *-vel*, among the other analysis generated by the non-grounded declination level.[3]

4 Details of the Implementation

The interface to our code is done through the predicate `dicionario`, with three arguments:

```
dicionario(Word, Level-of-analysis, Resulting-structure)
```

and the standard modes of operation are: `dicionario(+,+,-)` and `dicionario(+,-,-)` (but see below on bi-directionality). That is, given a word and the desired level of analysis, the predicate returns the resulting structure (the lexical class and its attributes). Or given a word, the system would try to analyze it, return its analysis and the level where the analysis was obtained.

The code was conceived to be efficient, bi-directional and to implement as straight forward as possible the morphological rules of Portuguese.

We believe that the most common use of our system would be the grounded declination analysis, and thus, efficiency was tuned for that. There are two aspects on efficiency: how long the program takes to compute a solution, and, specifically for Prolog, how many backtracking does it take for the predicate to return the "correct" answer. In terms of the first aspect of efficiency, we implemented some "tricks" like representing a word as a list of its letters in inverse order. This way, the declinations would appear in the beginning of the list, and thus their removal would be cheaper than if it were at the end of the list [O'K90]. Also in this realm of efficiency, we tried to explore Prolog internal hashing scheme (which we assumed is the standard scheme of hashing clauses by the functor and its first argument [Ari88, SB-88]) so that searches on the dictionary are fast.

In terms of efficiency in generating the "correct" answer as soon as possible, we rearranged the order of predicates in the program so the first solutions generated are what we consider the most common ones. Thus, given the word *eles*, the system would first answer that it is a pronoun, and only upon backtracking it will analyze it as the plural of the word *ele* (the letter ell).

The system was also designed to be bidirectional, that is, to serve both as an analyzer and a generator. As a generator, the predicate works in the mode `dicionario(-,+,+)`, that is, given a structure and the level of analysis, the predicate constructs the word. So, one can ask the system for the plural of *pão* (bread) and it will return *pões*, and then on backtracking *pães*, and finally *pãos*. The first and last words are not the correct plural of *pão*, but they are reasonable tries given the rules of plural formation of words ending in *-ão*.

Finally, the last requirement of the system was that the morphological rules were implemented as straight forward as possible. Morphological rules are usually expressed [dC80] in

[3]This is not just problem of non-groundedness. If both verbs where in the dictionary, a grounded affixation analysis would make the same mistakes. Non-groundedness only makes more mistakes.

terms of what one wants to accomplish (for example, to obtain the plural), the sequence of letters at the end of original word (for example, words ending in *-ão*), and the consequent of the rule are usually removal of letters form the original word, follow by addition of sequence of letters to the end of the word, yielding the final declination (for example, remove *-ão* and add *-ões*). In our system each rule corresponds to a Prolog clause, which in some way decreases the efficiency of the program (since two rules could be combined into a single one like the compound rule to generate the feminine plural declination of a adjective), but it increases in readability, modularity and maintainability. Since we did not implement all morphological rules of Portuguese (nor do we know where to find such a compilation of rules), and so the system will suffer modifications in the future, maintainability is an important concern.

5 Conclusions

This paper does not report a a ground breaking result, but it reports on a nice implementation of a morphological analyzer in Prolog. TO our knowledge no description of such system has been published in he literature. The system presented here does more that what is usually expected from a morphological analyzer, which corresponds to our grounded declination level of analysis. The system also understands about affixation (grounded affixation analysis), and is even able to guess the lexical class and root of unknown words (the non-grounded analysis).

The is, of course, much work yet to be done, especially on the level of reducing the incorrect analysis and guesses the system produce. As we mentioned these incorrectness are the result of the underspecificity of the morphological rule *as they appear in Portuguese language grammars*. There is a interesting line of research to be followed which would try to make those morphological rules more specific. For example, we treated prefixes and suffixes independently, but there we found evidence that not all combinations of prefix/suffix are possible. Thus one could propose a matched treatment of suffixes and prefixes. Also we assumed that any prefixed could be applied to any root, but there is also evidence that prefixes may be typed (like suffixes). For example the prefix *extra-* is applied mostly to adjectives, and only rarely to the other classes. There is only one verb with the prefix *hiper-* (*hiperestesiar*, and no verb with prefix *hipo-*, except *hipotecar* which only by chance starts with *hipo*.

References

[Ari88] *The Arity/Prolog Language Reference Manual*, 1988.

[dC80] Celso Ferreira da Cunha. *Gramatica da Lingua Portuguesa*. FENAME, seventh edition, 1980.

[dH80] Aurelio Buarque de Hollanda. *Dicionario da Lingua Portuguesa*. Editora Nova Fronteira, 1980.

[Gaz85] Gerald Gazdar. Finite state morphology: a review of koskenniemi (1983). *Linguitics*, 23(4):597–607, 1985.

[Kar83] L. Karttunen. Kimmo: a general morphological processor. In *Texas Linguistics Forum*, volume 22, pages 165–186, 1983.

[Kos83] K. Koskenniemi. A two-level model for morphological analysis. In *Proceeding of the 8th International Joint Conference on Artificial Intelligence*, pages 683–685, Karlsruhe, 1983.

[Kos85] K. Koskenniemi. A general computational model for word-form recognition and production. In *22nd Annual conference of the association of computational linguistics*, Stanford University, 1985.

[LK93] C. L. Lucchesi and T. Kowaltowski. Applications of finite automata representing large vocabularies. *Software - Practice and Experience*, 1(23):15–30, 1993.

[O'K90] Richard O'Keefe. *The Craft of Prolog.* MIT Press, 1990.

[PS87] Fernando C. N. Pereira and Stuart M. Shieber. *Prolog and Natural-Language Analysis.* Center for the Study of Language and Information, Stanford,CA, 1987. CSLI Lecture Notes Number 10.

[RRBP92] G.D. Ritchie, G.J. Russell, A.W. Black, and S.G. Pulman. *Computational Morphology.* MIT Press, 1992.

[SB-88] *SB-Prolog 3.1*, 1988.

Relatórios Técnicos – 1992

92-01 **Applications of Finite Automata Representing Large Vocabularies,** *C. L. Lucchesi, T. Kowaltowski*

92-02 **Point Set Pattern Matching in d-Dimensions,** *P. J. de Rezende, D. T. Lee*

92-03 **On the Irrelevance of Edge Orientations on the Acyclic Directed Two Disjoint Paths Problem,** *C. L. Lucchesi, M. C. M. T. Giglio*

92-04 **A Note on Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams,** *W. Jacometti*

92-05 **An (l, u)-Transversal Theorem for Bipartite Graphs,** *C. L. Lucchesi, D. H. Younger*

92-06 **Implementing Integrity Control in Active Databases,** *C. B. Medeiros, M. J. Andrade*

92-07 **New Experimental Results For Bipartite Matching,** *J. C. Setubal*

92-08 **Maintaining Integrity Constraints across Versions in a Database,** *C. B. Medeiros, G. Jomier, W. Cellary*

92-09 **On Clique-Complete Graphs,** *C. L. Lucchesi, C. P. Mello, J. L. Szwarcfiter*

92-10 **Examples of Informal but Rigorous Correctness Proofs for Tree Traversing Algorithms,** *T. Kowaltowski*

92-11 **Debugging Aids for Statechart-Based Systems,** *V. G. S. Elias, H. Liesenberg*

92-12 **Browsing and Querying in Object-Oriented Databases,** *J. L. de Oliveira, R. de O. Anido*

Relatórios Técnicos – 1993

93-01 **Transforming Statecharts into Reactive Systems,** *Antonio G. Figueiredo Filho, Hans K. E. Liesenberg*

93-02 **The Hierarchical Ring Protocol: An Efficient Scheme for Reading Replicated Data,** *Nabor das C. Mendonça, Ricardo de O. Anido*

93-03 **Matching Algorithms for Bipartite Graphs,** *Herbert A. Baier Saip, Cláudio L. Lucchesi*

93-04 **A lexBFS Algorithm for Proper Interval Graph Recognition,** *Celina M. H. de Figueiredo, João Meidanis, Célia P. de Mello*

93-05 **Sistema Gerenciador de Processamento Cooperativo,** *Ivonne. M. Carrazana, Nelson. C. Machado, Célio. C. Guimarães*

93-06 **Implementação de um Banco de Dados Relacional Dotado de uma Interface Cooperativa,** *Nascif A. Abousalh Neto, Ariadne M. B. R. Carvalho*

93-07 **Estadogramas no Desenvolvimento de Interfaces,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

93-08 **Introspection and Projection in Reasoning about Other Agents,** *Jacques Wainer*

93-09 **Codificação de Seqüências de Imagens com Quantização Vetorial,** *Carlos Antonio Reinaldo Costa, Paulo Lício de Geus*

93-10 **Minimização do Consumo de Energia em um Sistema para Aquisição de Dados Controlado por Microcomputador,** *Paulo Cesar Centoducatte, Nelson Castro Machado*

93-11 **An Implementation Structure for RM-OSI/ISO Transaction Processing Application Contexts,** *Flávio Morais de Assis Silva, Edmundo Roberto Mauro Madeira*

93-12 **Boole's conditions of possible experience and reasoning under uncertainty,** *Pierre Hansen, Brigitte Jaumard, Marcus Poggi de Aragão*

93-13 **Modelling Geographic Information Systems using an Object Oriented Framework,** *Fatima Pires, Claudia Bauzer Medeiros, Ardemiris Barros Silva*

93-14 **Managing Time in Object-Oriented Databases,** *Lincoln M. Oliveira, Claudia Bauzer Medeiros*

93-15 **Using Extended Hierarchical Quorum Consensus to Control Replicated Data: from Traditional Voting to Logical Structures,** *Nabor das Chagas Mendonça, Ricardo de Oliveira Anido*

93-16 $\mathcal{LL}$ – **An Object Oriented Library Language Reference Manual,** *Tomasz Kowaltowski, Evandro Bacarin*

93-17 **Metodologias para Conversão de Esquemas em Sistemas de Bancos de Dados Heterogêneos,** *Ronaldo Lopes de Oliveira, Geovane Cayres Magalhães*

93-18 **Rule Application in GIS – a Case Study,** *Claudia Bauzer Medeiros, Geovane Cayres Magalhães*

93-19 **Modelamento, Simulação e Síntese com VHDL,** *Carlos Geraldo Krüger e Mário Lúcio Côrtes*

93-20 **Reflections on Using Statecharts to Capture Human-Computer Interface Behaviour,** *Fábio Nogueira de Lucena e Hans Liesenberg*

93-21 **Applications of Finite Automata in Debugging Natural Language Vocabularies,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

93-22 **Minimization of Binary Automata,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

93-23 **Rethinking the** DNA **Fragment Assembly Problem,** *João Meidanis*

93-24 EGOLib — **Uma Biblioteca Orientada a Objetos Gráficos,** *Eduardo Aguiar Patrocínio, Pedro Jussieu de Rezende*

93-25 **Compreensão de Algoritmos através de Ambientes Dedicados a Animação,** *Rackel Valadares Amorim, Pedro Jussieu de Rezende*

93-26 GeoLab**: An Environment for Development of Algorithms in Computational Geometry,** *Pedro Jussieu de Rezende, Welson R. Jacometti*

93-27 **A Unified Characterization of Chordal, Interval, Indifference and Other Classes of Graphs,** *João Meidanis*

93-28 **Programming Dialogue Control of User Interfaces Using Statecharts,** *Fábio Nogueira de Lucena e Hans Liesenberg*

93-29 EGOLib – **Manual de Referência,** *Eduardo Aguiar Patrocínio e Pedro Jussieu de Rezende*

Relatórios Técnicos – 1994

94-01 **A Statechart Engine to Support Implementations of Complex Behaviour,** *Fábio Nogueira de Lucena, Hans K. E. Liesenberg*

94-02 **Incorporação do Tempo em um** SGBD **Orientado a Objetos,** *Ângelo Roncalli Alencar Brayner, Claudia Bauzer Medeiros*

94-03 **O Algoritmo KMP através de Autômatos,** *Marcus Vinícius A. Andrade e Cláudio L. Lucchesi*

94-04 **On Edge-Colouring Indifference Graphs,** *Celina M. H. de Figueiredo, João Meidanis, Célia Picinin de Mello*

94-05 **Using Versions in** GIS, *Claudia Bauzer Medeiros and Geneviève Jomier*

94-06 **Times Assíncronos: Uma Nova Técnica para o** Flow Shop Problem, *Hélvio Pereira Peixoto e Pedro Sérgio de Souza*

94-07 **Interfaces Homem-Computador: Uma Primeira Introdução,** *Fábio Nogueira de Lucena e Hans K. E. Liesenberg*

94-08 **Reasoning about another agent through empathy,** *Jacques Wainer*

94-09 **A Prolog morphological analyser for Portuguese,** *Jacques Wainer, Alexandre Farcic*

94-10 **Introdução aos Estadogramas,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

94-11 **Matching Covered Graphs and Subdivisions of K_4 and $\overline{C_6}$,** *Marcelo H. de Carvalho and Cláudio L. Lucchesi*

94-12 **Uma Metodologia de Especificação de Times Assíncronos,** *Hélvio Pereira Peixoto, Pedro Sérgio de Souza*