# Programação de dicionários

JOSÉ JOÃO DIAS DE ALMEIDA
(Universidade do Minho)

**Resumo:**

Na definição (clássica) das entradas de dicionários há uma enorme quantidade de trabalho que é repetitivo. Esta repetição traduz-se num enorme esforço e numa grande dificuldade de ser coerente. Muita dessa coerência consiste em representar de forma idêntica fenómenos repetidos.

Neste documento apresenta-se uma abordagem à definição de dicionários usando uma linguagem de definição (construída especialmente para o efeito) que possibilita:

- seguir um modelo formal especificado.
- definição de entradas sem preocupação com o aspecto tipográfico.
- definição separada de formatos de saída (tipográficos ou não)
- definições orientadas ao conceito (em vez de orientado à palavra)
- definição de funções (que armazenem partes repetitivas a usar por várias entradas)

Na linguagem usada, é seguida uma orientação geral baseada em *(typed) feature structures*.

**Palavras chave**: TFS, dicionários, semântica

## 1. Introdução

Este trabalho insere-se num projecto de especificação e processamento de dicionários, envolvendo também a parte dinâmica e multi-fonte.

Os principais objectivos deste trabalho correspondem a construir um ambiente para definição de dicionários que seja eficaz (do ponto de vista dos seus criadores) e que torne mais leve o crescimento e manutenção de dicionários não industriais.

Optou-se pela criação de uma linguagem de programação de dicionários com:

- definições orientadas ao conceito
- definição de uma linguagem própria para especificar *termos compostos*
- definição de funções que permitam que não seja necessário repetir trabalho comum
- separação entre definições de conceitos e suas saídas/utilizações
- permitir diferentes saídas para a mesma definição
- verificação formal de consistência (**ainda não implementado**)

A primeira decisão de projecto foi a de que toda a informação existente será vista como feature structures (FS) de forma a constituir um tipo de dados universal e de levar a que toda a informação tenha uma etiqueta (campo) indicando de que tipo de informação se trata. Esta estruturação pode parecer restritiva mas acreditamos que é imprescindível à capacidade processamento subsequente.

As operações básicas são as operações básicas de FS tendo um carácter fundamental a operação de união de FSs (operador "+" infixo na nossa linguagem).

Alguns campos mais habituais têm uma sintaxe própria de modo a tornar mais curta a quantidade de caracteres que se tenha que digitar. Estão nesta classe campos como: *classe, exemplos de uso, paráfrase de exemplo de uso, nível de linguagem...*

A sintaxe concreta usada (definida ao gosto dos seus criadores) é o mais compacta possível mas pode muito facilmente ser alterada de modo a ter uma mais fácil leitura.

## 2. Descrição do sistema de definição de dicionários

### 2.1 Estado actual de desenvolvimento

Presentemente existe construído um protótipo do sistema que permite desenvolver dicionários, não sendo no entanto ainda uma ferramenta industrial. O reconhecedor escrito em YACC e PERL que reconhece o texto do dicionário e pode gerar várias saídas.

As características actuais são as seguintes:

- é muito fácil de alterar (210 linhas de código PERL)
- permite definição de função (com ou sem argumentos)
- saídas:

⇒ ASCII, ver exemplos
⇒ geração automática de HTML (bastante primitivo)
⇒ geração de base de dados PERL - DB_File (usado como base de dados para rápida procura em grandes quantidades de informação)
⇒ previsto: LaTeX,

• Testado com exemplos pequenos-médios (4000 linhas de entrada, cerca de 6000 vedetas)

É provável que o tipo de dicionário usado nos testes (dicionário de calão e idiomáticas) tenha influenciado algumas características da sua versão actual e que seja necessário acrescentar-lhe outras referentes a fenómenos que não surgiram.

Nas secções seguintes serão apresentadas algumas facetas da linguagem de programação de dicionários, sendo também apresentados alguns exemplos.

## 2.2 Alguns campos com notação própria

Classes:

```
termo < cla : sem ;
```

é uma abreviatura de:

```
{name = termo;
isa  = cla;
sem  = sem
}
```

Uso de exemplos/parafrase:

```
puto:nada+"não sabe puto"=nao sabe nada ;
```

é uma abreviatura de

```
{name = puto;
ex = não sabe puto;
paraf = não sabe nada;
sem  = nada
}
```

### 2.3 Definição orientada ao conceito e definição de funções

Os exemplos apresentados não devem ser analisados apenas como meios de exemplificar capacidades de especificação.

Considere-se a seguinte texto de definição de dicionário:

```
1  #
2  sub n2 +nível => 'coloquial'
3  sub sm +género=> 'masculino',
4          gram  => 'só usado no masculino'
5  #
6  gralha < animal : ave preta da famílias dos corvídeos;
7  gralha++n2 ! erro ! gato++sm++n2 : incorrecção num texto;
```

Comentários:

**linha 2--4** – zona das definições de funções; a função n2 será usada para descrever termos que tenham um nível de linguagem do tipo coloquial; a função sm serve para descrever a propriedade de ser um termo masculino em que não se possa usar o feminino correspondente.

**linha 6** – *gralha* pertence à classe dos *animais* com uma semântica associada de *ave preta...*

**linha 7** – descrição do conceito erro como uma lista de sinónimos e sua semântica. Alguns dos sinónimos são acrescidos de funções descrevendo propriedades que só a eles afectam.

Esta definição produz a seguinte saída em modo ASCII:

```
[{name => gralha
    sem => ave preta da famílias dos corvídeos
    isa => animal}
 {name => gralha
    syn => [
      erro
      gato]
    sem => incorrecção num texto
    nível => coloquial}
 {name => erro
```

```
    syn => [
      gralha
      gato]
    sem => incorrecção num texto}
{name => gato
    género => masculino
    syn => [
      gralha
      erro]
    sem => incorrecção num texto
    nível => coloquial
    gram => só usado no masculino}
```

Como pode ser verificado, uma única linha (orientada ao conceito) pode produzir vários termos do dicionário (orientados à palavra, ou ao termo).

### 2.4 Sub-linguagem para termos compostos

Nos exemplos analisados transpareceu a necessidade de definir um modo de descrever termos compostos (termos multi-palavra).

Essa sub linguagem permite descrever variantes de expressões em que se descreve qual/quais as palavras que podem ser flexionadas (usando "*") alternativas (usando "(" ")" e "|") e outras características (para mais detalhes ver[7]).

Seguem-se alguns exemplos

```
...
ter* (boml) ouvido :
    ter facilidade de aprendizagem musical;

(aprender*|tocar*) de ouvido :
    __ música sem partituras nem professores;

(ser* um homem (com|de)|ter*) tomates++n3
| ter*-os no sítio++n2:
    sem = ser valente;
     uso = só se aplica a homens! ;

(ir*|mandar*) à merda++n4
| (mandar*|ir*) abaixo de Braga++n2:
     sem = +diabo;
      nota = ...refere à antiga lixeira (Frossos) ;
-------------------------
```

```
{name => ter*-os no sítio
   syn => [ (ser* um homem (com|de) |ter*) tomates ]
   sem => ser valente
   nível => calão
   uso => só se aplica a homens! }

{name => ter* (bom|) ouvido
   sem => ter facilidade de aprendizagem musical}

{name => (aprender*|tocar*) de ouvido
   sem => __ música sem partituras nem professores}

{name => (mandar*|ir*) abaixo de Braga
   syn => [ (ir*|mandar*) à merda ]
   nota => expressão antiga ... Frossos)
   isa => interjeição
   sem => ordem de não aborrecer e de se ir embora
   nível => coloquial}
```

### 2.5 Definição colectivas

Existem definidos duas maneiras alternativas para definições colectivas: o operador setofelement e o operados begin ... end

#### 2.5.1 SetOfElement

```
+setofele([ Janeiro , Dezembro], { isa = mês; sem = é uma subdivisão do ano )}

--------------------
 {name => Janeiro
   isa => mês
   sem => é uma subdivisão do ano}
 {name => Dezembro
   isa => mês
   sem => é uma subdivisão do ano}]
```

#### 2.5.2 begin ... end

Apresenta-se seguidamente-se um exemplo do uso deste operador:

```
begin {isa = mamífero; género = masculino; área = zoologia};
```

```
 cão : melhor amigo do homem;
 gato : animal que tem sete vidas;
 elefante : ;
end;
```

### 2.5.3 Funções para definição de entradas associadas

```
+adnc(democrata, {name = democracia;
                  isa = regime político ;
                  sem = regime que ... });
----------
 {cat => nome comum
    name => democracia
    seealso => {
      name => democrata
      sem => aquele que defende e pratica democracia
    isa => regime político
    sem => regime que ... }

 {name => democrata
    cat => adjectivo ou nome comum
    sem => |
       { cat => nome comum
         prop => aquele que defende e pratica democracia
         isa => [ pessoa] }
        {cat => adjectivo
         prop => referente a democracia
         isa => [ propriedade]}]}
```

## 2.6 Conclusões

A experiência mostrou que embora a construção de dicionários seja uma tarefa muito complexa e demorada, que há toda a utilidade em dispor de ferramentas como a que aqui se apresenta que permitam ir usando imediatamente os dicionários inacabados e que permitam também que uma mesma definição possa servir para vários usos.

Experiências realizadas mostraram ser fácil gerar dicionários automaticamente consultáveis pela internet com base nesta ferramenta.

Planeia-se fazer brevemente a adaptação da linguagem de modo a permitir uma cómoda utilização em dicionários multilingue.

Planeia-se também proceder à criação de um nível de validação formal das definições, tomando como base uma especificação dos campos e suas propriedades.

BIBLIOGRAFIA:

J.J. ALMEIDA. Especificação e tratamento de dicionários. In *Actas do XI Congresso da Associação Portuguesa de Linguística, Lisboa 1995*, volume 2, 1996.

JORGE M. BAPTISTA. Estabelecimento e formalização de classes de nomes compostos. Master's thesis, Fac. Letras, Universidade de Lisboa, 1994.

MARGARITA CORREIA. Bases digitais lexicias na união europeia. In *Simpósio de Lexicologia, Lexicografia e Terminologia*, 1994. Araraquara.

NANCY IDE AND JEAN VÉRONIS. Encoding dictionaries. In *Text encoding iniciative*, pages 167-179. Kluwer Academic publisher, 1995.

HANS-ULRICH KRIEGER. Typed feature formalisms as a common basis for linguistic specification. Research Report RR-94-39, Deutsches Forschungszentrum für Künstliche Intelligenz, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany, 1995. Springer, Berlin, 1995.

Projecto Natura. Processamento de linguagem natural, 1995. http://www.di.uminho.pt/~jj/pln/pln.html.

ULISSES PINTO AND J.J. ALMEIDA. Tratamento automático de termos compostos. In *Actas do XI Congresso da Associação Portuguesa de Linguística, Lisboa 1995*, volume 2, 1996.

C.M. SPERBERG-MCQUEEN AND LOU BURNARD. *Guidelines for Electronic Text Encoding and Interchange (TEI P3)*. Chicago: ACH/ACL/ALLC, 1994.

LARRY WALL AND RANDAL L. SCHWARTZ. *Programming PERL*. O'Reilly and Associates, 1991.