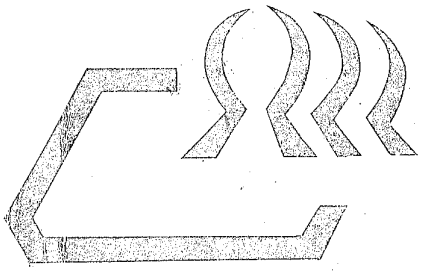
# ANAIS

## VOLUME I



VI CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO

RECIFE - OLINDA 19 A 25 DE JULHO 86

# VI CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇAO

19 A 25 DE JULHO DE 1986

RECIFE - OLINDA

## ANAIS

VOLUME I

## TRABALHOS APRESENTADOS

XIII SEMINARIO INTEGRADO DE SOFTWARE E HARDWARE

EDITOR: Clylton José Galamba Fernandes

PROMOÇAO

SBC - Sociedade Brasileira de Computação
UFPE - Universidade Federal de Pernambuco

PATROCINIO

FINEP, CNPq, CAPES, SEI, COBRA, SERPRO
Fundação Arthur Bernardes

APOIO

Governo do Estado de Pernambuco

XIII SEMISH VI CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO

RECIFE-OLINDA 19 A 25 DE JULHO DE 1986

# PESQUISA ABSTRATA EM TABELAS : UMA VISAO UNIFICADORA

C. F. E. MACIEL WAGA * e PAULO A. S. VELOSO **

SUMARIO

O objetivo deste trabalho è mostrar o poder da abstração na resolução de problemas e no desenvolvimento de programas. O problema de busca em tabelas è formulado de maneira geral e abstrata. Para este problema, um algoritmo geral è desenvolvido e verificado, com base em um tipo abstrato de dados. Os algoritmos usuais são implementações deste algoritmo abstrato, que capta a essência do processo. Algumas vantagens deste enfoque são discutidas.

ABSTRACT

This paper aims at showing the power of abstraction in problem solving and program development. The problem of table searching is formulated in a general and abstract manner. Then, a general algorithm is developed and verified based on an abstract data type. The usual algorithms are implementations of this abstract algorithm, which captures the essence of the procedure. Some advantages of this approach are discussed.

* Mestre em Informàtica ( PUC/RJ, 1984 ); teoria e metodologia de programação; aluna de doutorado na PUC/RJ; Depto. de Informàtica, PUC/RJ, Rua Marquês de São Vicente 225 , 22453 Rio de Janeiro RJ.

** Ph. D. em Ciência da Computação ( Univ. Califòrnia, Berkeley, 1975 ); teoria e metodologia de programação, especificações formais; Professor associado na PUC/RJ; Depto. de Informàtica, PUC/RJ, Rua Marquês de São Vicente 225 , 22453 Rio de Janeiro RJ.

## I. INTRODUÇÃO

Este trabalho pretende mostrar o poder da abstração na resolução de problemas e no desenvolvimento de programas. Abstratamente, temos "o" problema de pesquisa em tabelas, sua solução é "o" algoritmo de pesquisa. Os algoritmos usuais são implementações deste algoritmo abstrato.

Um problema bem frequente em computação é o da recuperação, acesso ou consulta a informações em tabelas ou arquivos. O problema da pesquisa pode ser descrito informalmente como sendo o problema de, dada uma chave a ser pesquisada, encontrar um elemento que tenha o campo chave igual a esta chave de pesquisa. Ao final da pesquisa pode-se ter tido sucesso ou não, conforme se tenha ou não encontrado um elemento satisfazendo a condição.

Vários algoritmos são conhecidos para se pesquisar se uma determinada informação está ou não presente em uma tabela [Knuth]. O objetivo deste trabalho é de capturar e formalizar as idéias comuns, ou seja, a abstração que está embutida em todos estes algoritmos, obtendo assim um algoritmo geral e abstrato que descreve o que é pesquisa. Esta formalização é feita através de tipos abstratos de dados (TAD). Desta forma, os algoritmos conhecidos tornam-se implementações deste programa abstrato (PA).

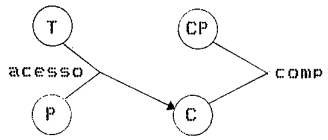
## II. TIPO ABSTRATO DE DADOS

O processo de programação com TAD's possui duas características interessantes. A primeira é a proximidade do domínio do problema, sendo irrelevantes questões como escolha de estruturas de representação. A segunda é que dada uma formulação do problema sobre um TAD escolhido,

basta estendê-lo para suportar um PA e especificar esta extensão para a verificação da corretude do PA.

## II.1 ESPECIFICAÇÃO DO PROBLEMA

Para o problema da pesquisa, consideremos o TAD com os sortes: T cujos objetos são estruturas finitas sobre as quais a pesquisa será efetuada, CP de objetos que são chaves a serem pesquisadas, P cujos objetos são indicadores com os quais se faz acesso a uma estrutura t de T, C de objetos que são valores de chaves acessadas em t e R de resultados para o problema, isto é, se determinada chave foi ou não encontrada. Para descrevermos o problema, temos que considerar um função **acesso**: T x P --> C, que efetivamente "recupera" informação em um certo t e um predicado binário **comp**: CP x C que observa "igualdade" entre chaves.




Como descrição das propriedades deste problema, podemos apresentar a especificação q (t,r) : [ ∃j:P comp (cp,acesso (t,j)) & r=OUI] v

[¬ ∃j:P comp (cp,acesso (t,j)) & r=NON]

OUI e NON são funções cujo entendimento é bem intuitivo. Podem significar que uma chave ocorre em uma estrutura ou que ocorre na posição k e que não ocorre, respectivamente.

## II.2 PROGRAMA ABSTRATO

O algoritmo para fazer pesquisa é bem simples: dadas uma estrutura t e uma chave a ser pesquisada cp, deve-se acessar t dado um indicador p e verificar se o elemento acessado c é "igual" a cp. Este processo deve

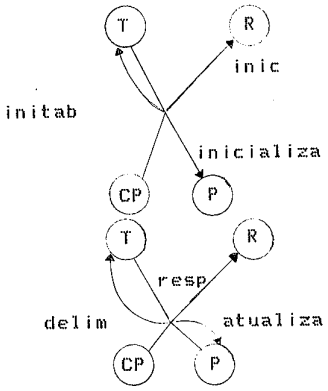
ser repetido até que t esteja esgotado, cp ocorra em t ou se perceba que se deve parar a busca já que não se vai encontrar cp em t daí para a frente.

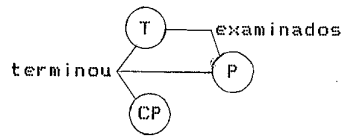
Esta repetição estabelece que se deve ter um resultado positivo para o problema se e somente se pelo menos um dos elementos já acessados em t satisfizer o predicado **comp**. Isto é o invariante do PA, que está descrito através da fórmula:

$$\exists j{:}P\ [examinados\ (t,j)\ \&\ comp\ (cp, acesso\ (t,j))] \longleftrightarrow r=OUI$$

que vamos abreviar por inv (t,r).

Devemos estender o TAD original acrescentando algumas operações. As funções **inic**: T x CP --> R e **inicializa**: T x CP --> P inicializam um certo elemento r resultado e um indicador p, respectivamente. A função **initab**: T x CP --> T cria uma cópia da estrutura de entrada. As funções **delim**: T x P x CP --> T e **atualiza**: T x P x CP --> P são de atualização e **resp**: T x P x CP --> R dá o valor do resultado da pesquisa. Temos ainda dois predicados **terminou**: T x P x CP, que descreve a condição de fim da repetição indicada acima e **examinados**: T x P, que indica quais elementos já foram observados.






Abaixo, temos o programa abstrato anotado com assertiva e o invariante.

```
pgm=abst pesquisa (tab:T,cp:CP)
(* tab, cp : parâmetros de entrada *)
(* r       : parâmetro  de saída  *)
var:  t:T, p:P;
   r := inic(tab,cp)
   t := initab(tab,cp)
   p := inicializa(t,cp)
{inv (t,r): ∃j:P [examinados(t,j) & comp(cp,acesso(t,j))] <--> r=OUI}
   enquanto  - terminou(t,p,cp) faça
      se   comp(cp,acesso(t,p))
      então
           r := resp(t,p,cp)
      senão
           t := delim(t,p,cp)
           p := atualiza(t,p,cp)
      fim=se
   fim=enquanto.
{q(t,r):[  ∃j:P comp(cp,acesso(t,j)) & r=OUI] v
        [¬ ∃j:P comp(cp,acesso(t,j)) & r=NON]}
```

II.3 ESPECIFICAÇÃO DO TIPO ABSTRATO DE DADOS

Até aqui apresentamos o TAD de forma bem intuitiva. Devemos estabele-

cer precisamente o comportamento das operações, ou seja, a especificação do TAD , que será dada através de axiomas. Estes axiomas pretendem capturar a intuição por trás do PA e auxiliar na sua prova de correção.

Ax1. ¬ ∃j:P [examinados (initab (tab,cp),j)]

Ax2. terminou (t,p,cp) <--> [finished (t,p) v ¬ adianta (t,p,cp) v
v r=OUI]

Ax3. finished (t,p) --> ∀j:P examinados (t,j)

Ax4. ¬ adianta (t,p,cp) --> ∀j:P [¬ examinados (t,p) -->
--> ¬ comp (cp, acesso (t,j))]

Ax5. ∀j:P [examinados (delim (t,p),j) <--> j=p v examinados (t,j)]

Ax6. ∀j:P [comp (cp,acesso (delim (t,j,cp)) -->
--> comp (cp,acesso (t,j))]

II.4 VERIFICAÇÃO

A verificação do PA pode ser dividida em corretude parcial e terminação [Manna, Veloso].

II.4.1 CORRETUDE PARCIAL

Para que o PA esteja parcialmente correto, devemos mostrar que as seguintes condições de verificação valem:

Cv1: inv (t,r) (p/[inicializa (t,cp)]) (t/[initab (tab,cp)])
(r/[inic (tab,cp)]), i.e.,
∃j:P [examinados (initab (tab,cp),j) &
& comp (cp,acesso (initab (tab,cp),j))] <--> inic (tab,cp)=OUI

Cv2: inv (t,r) & ¬ terminou (t,p,cp) & comp (cp,acesso (t,p)) -->
--> inv (t,r) (r/[resp (t,p,cp)]), i.e.,

( ∃j:P [examinados (t,j) & comp (cp,acesso (t,j))] <--> r=OUI) &

& ¬ terminou (t,p,cp) & comp (cp,acesso (t,p)) -->

--> ( ∃j:P [examinados (t,j) & comp (cp,acesso (t,j))] <-->

<--> resp (t,p,cp)=OUI)

Cv3: inv (t,r) & ¬ terminou (t,p,cp) & ¬ comp (cp,acesso (t,p)) -->

--> inv (t,r) (p/[atualiza (t,p,cp)]) (t/[delim (t,p,cp)]), i.e.,

( ∃j:P [examinados (t,j) & comp (cp,acesso (t,j))] <--> r=OUI) &

& ¬ terminou (t,p,cp) & ¬ comp (cp,acesso (t,p)) -->

--> ( ∃j:P [examinados (delim (t,p,cp),j) &

& comp (cp,acesso (delim (t,p,cp),j))] <--> r=OUI)

Cv4: inv (t,r) & terminou (t,p,cp) --> q (t,r), i.e.,

( ∃j:P [examinados (t,j) & comp (cp,acesso (t,j))] <--> r=OUI) &

& terminou (t,p,cp) --> ([ ∃j:P comp (cp,acesso (t,j)) & r=OUI]

v [¬ ∃j:P comp (cp,acesso (t,j)) & r=NON])

Não é difícel ver que as condições acima decorrem dos axiomas.

II.4.2 TERMINAÇÃO

Terminação de programas envolve idéias sobre "quantidades" que estão diminuindo ao longo do processamento (função potencial).

No PA para pesquisa, tomaremos uma função total **tam**: T --> IN, que dada uma estrutura, dá como resultado um número natural que indica seu "tamanho". Vamos tomar tam (t) como função potencial, associada a iteração do PA. Consideremos a condição de verificação para terminação:

Ct: [tam (t) >= 0 & ¬ terminou (t,p,cp)] -->

--> tam (t) > tam (delim (t,p,cp))

Ct é a condição de decréscimo estrito da função potencial [Veloso].

Esta condição também é verificada utilizando-se o axioma a seguir, que

completa a especificação do TAD, e o fato de a relação de ordem < de IN ser bem fundada.

Ax7. ¬ terminou (t,p,cp) ⟶ tam (delim (t,p,cp)) < tam (t)

## III. COMPARAÇÃO

A título de exemplificação da aplicabilidade do PA apresentado anteriormente, vamos compará-lo com dois algoritmos de pesquisa bem conhecidos: pesquisa sequencial e binária.

Dados: tabela com registros r1,...,rn com chaves k1,...,kn, respectivamente e uma chave a ser pesquisada k.

### III.1 PESQUISA SEQUENCIAL

```
i:=1                          ⟶ inicializa
WHILE i <= n DO               ⟶ terminou
   IF  k = ki                 ⟶ comp
   THEN
       EXIT;  (sucesso)       ⟶ resp
   i:=i+1;                    ⟶ (delim, atualiza)
```

### III.2 PESQUISA BINÁRIA

```
l:=1                          ⎫
u:=n                          ⎬ inicializa
i:=l+u/2                      ⎭
WHILE u < l  DO               ⟶ terminou
   IF  k = ki                 ⟶ comp
   THEN
       EXIT;  (sucesso)       ⟶ resp
   IF  k < ki
   THEN
```

```
        u:=i-1
    ELSE                        } (delim, atualiza)
        l:=i+1;
    i:=l+u/2;
```

Podemos observar que estes dois algoritmos familiares são implementações do PA, onde escolheu-se uma determinada representação para os objetos dos sortes e para as operações e predicados do TAD.

## IV. RETROSPECTO

As seções anteriores mostraram como se pode desenvolver um algoritmo geral para o problema de pesquisa em tabelas, por meio do método dos tipos abstratos de dados.

Agora, vamos passar em revista as principais características gerais deste método, dos pontos de vista de desenvolvimento de programas e de resolução de problemas:

1. A formulação do problema, que é a especificação do programa, é feita por meio de uma fórmula que descreve o comportamento entrada-saída desejado. Esta descrição é facilitada pela abstração de detalhes quanto à particular maneira de representar os objetos envolvidos. Assim, o problema é formulado sobre um TAD.
2. Postula-se uma extensão do TAD do problema por meio de operações intuitivamente próximas ao domínio do problema e que pareçam apropriadas para resolvê-lo.
3. Com base na visão intuitiva do TAD, escreve-se um programa abstrato PA para resolver o problema abstrato em questão.
4. O comportamento das operações adicionadas ao TAD original é especificado de modo a se poder garantir a corretude do PA, usando as

condições de verificação como guia para obter a especificação deste TAD estendido.

5. Levando em conta agora, a particular representação desejada para os dados e resultados, escolhem-se representações para os outros objetos abstratos do TAD. Com base nestas representações são programadas as operações abstratas utilizando-se operações concretas realmente disponíveis. Em outras palavras o TAD é implentado sobre um tipo concreto.

Algumas vantagens do desenvolvimento de programas por TAD's, conforme esboçado acima, são [MVS]:

- fatoração do programa e tarefas associadas como documentação, verificação, etc., em duas partes independentes: programa abstrato e módulo de implementação;
- adiamento de decisões sobre representação;
- obtenção de um algoritmo geral abstrato, que pode ser especializado a várias versões concretas particulares;
- viabilidade da iteração do método, levando a refinamentos sucessivos.

## V. CONCLUSÕES

O emprego de abstrações nos leva a formular o problema de pesquisa de chave em tabela de forma geral e abstrata. Para este problema foi projetado um algoritmo igualmente geral e abstrato.

Algumas observações sobre este enfoque merecem atenção.

1. O algoritmo geral foi verificado, de uma vez por todas, com base na especificação do TAD.
2. Os vários algoritmos usuais, como os ilustrados em III, árvore bi-

nária de pesquisa, "hashing", interpolação, etc. correspondem a particulares implementações do algoritmo geral.

3. Para se verificar a corretude de um destes algoritmos familiares, assim enfocado, basta verificar que o módulo de implementação está correto: garante os axiomas da especificação do TAD.

Esta abordagem separa claramente dois aspectos do programa final resultante: os aspectos gerais de busca em tabela, incorporados no programa abstrato e os aspectos ligados à particular representação da tabela ou a decisões de projeto, embutidos no módulo de implementação.

A separação nítida mencionada acima permite uma melhor compreensão dos vários métodos da busca em tabela, possibilitando classificá-los em famílias. Tal classificação, além do interesse intrínseco, parece ser bastante útil para propósitos didáticos.

## VI. BIBLIOGRAFIA


[Knuth] Knuth,D.E. The art of computer programming vol 3: sorting and searching. Addison-Wesley, Reading, 1973.

[Manna] Manna,Z. The mathematical theory of computation. McGraw-Hill, New York, 1984.

[MVS] Mainbaum,T.S.E., Veloso,P.A.S., Sadler,M.R. A theory of abstract data types for program development: bridging the gap? Joint Conference on Theory and Practice of Software Development, Berlim, 1984.

[Veloso] Veloso,P.A.S. Verificação e construção de programas. I Escola Brasileiro- Argentina de Informática, Campinas, 1986.

[Waga] Waga,C.F.E.M. Métodos para resolução de problemas. Rio de Janeiro, PUC, D.I., 1984, Dissertação de Mestrado.