ANAIS

Volume I

Trabalhos apresentados no
IX Seminário Integrado de
"Software" e "Hardware" - SEMISH

EDITORES - L. J. BRAGA-FILHO, E. G. DE SIMONE E N. MEISEL

SOCIEDADE BRASILEIRA DE COMPUTAÇÃO

DIRETORIA

Presidente: Luiz de Castro Martins
Vice-Presidente: Sílvio Davi Paciornick
Secretário-Geral: Sueli Mendes dos Santos
1º Secretário: Estevam Gilberto de Simone
2º Secretário: Ivan Moura Campos
Tesoureiro; Therezinha da Costa Ferreira Chaves

CONSELHO

Carlos Inácio Zamitti Mammana
Carlos José Pereira de Lucena
Cláudio Zamitti Mammana
Clésio Saraiva dos Santos
Henrique Pacca Luna
Ivan da Costa Marques
João Antônio Zuffo
Luiz Julião Braga-Filho
Mário Dias Ripper
Wilson de Pádua Paula Filho

A SBC tem como finalidade:

a) Incentivar atividades de ensino, pesquisa e desenvolvimento em computação no Brasil; b) zelar pela preservação e aprimoramento do espírito crítico, responsabilidade profissional e personalidade nacional da comunidade técnico-científia que atua no setor de computação no país; c) ficar permanentemente atenta à política governamental que afeta as atividades de computação no Brasil, no sentido de assegurar a emancipação tecnológica do país; d) especificamente, e enquanto for de interesse da sociedade, promover um seminário integrado de "software" e "hardware" nacionais; e) promover, por todos os meios, academicamente legítimos, através de reuniões, congressos, conferências e publicações, o conhecimento, informações e opiniões que tenham por objetivo a divulgação da ciência e os interesses da comunidade de computação.

Maiores informações: Av. Venceslau Brás, 71 Fundos Casa 27
22290 - Rio de Janeiro, RJ

II CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO

UNIVERSIDADE FEDERAL DE OURO PRETO

12 a 16 de julho de 1982

ANAIS

VOLUME I - TRABALHOS APRESENTADOS NO IX SEMINÁRIO INTEGRADO DE "SOFTWARE" E "HARDWARE" - SEMISH

Editores: L.J. Braga Filho, E.G. de Simone e N. Meisel

Pág. 495 a 510

UMA METODOLOGIA PARA VERIFICAR IMPLEMENTAÇÕES DE TIPOS ABSTRATOS DE DADOS

F. E. P. Pessoa*

P. A. S. Veloso

Departamento de Informática- PUC/RJ
Rua Marques de São Vicente, 225
CEP 22.453 -Rio de Janeiro - RJ

SUMÁRIO

Uma metodologia para verificar a correção de uma implementação de um tipo abstrato de dados em outro, ambos especificados algebricamente, é apresentada, justificada e sua aplicação ilustrada. Esta metodologia advém de uma clarificação do conceito de implementação e do desejo de modularizar a verificação de correção. Uma implementação fica especificada por uma função de representação, incluindo a tradução da igualdade, e um invariante de representação. Com isto e as especificação de cada tipo, deve-se mostrar a existência de um isomorfismo apropriado. Os cinco passos da metodologia garantem isso, construindo o isomorfismo por etapas.

ABSTRACT

A methodology for verifying the correctness of an implementation of an abstract data type in another, both specified algebraically, is presented, justified and its application illustrated. This methodology stems from a clarification of the concept of implementation as well as from the desire to modularize the verification of correctness. An implementation is gives by a representation map, including the translation of equality and a representation invariant. Using this and the specification of each type, one must show the existence of a certain isomorphism. The five steps of the methodogy ensure this by means of a stepwise construction of the required isomorphism.

*Da Universidade Federal do Ceará, em licença.

1. INTRODUÇÃO

Abstração é inerente às aplicações dos computadores ao mundo real. Em particular, tipos abstratos de dados (TAD's) têm se revelado uma importante ferramenta para o desenvolvimento de programas. Seu uso permite ao programador escrever programas em termos de tipos de dados que ele considera melhor se adequarem à solução de um dado problema, ao invés de se restringir ao elenco de tipos disponíveis na linguagem de programação usada para codificar o programa. O resultado é a fatoração do programa em duas partes: uma abstrata, que manipula os objetos dos TAD's por meio de suas operações, e uma parte de implementação, que representa os objetos e operações abstratas por meio de outros mais concretos [8]. O emprego, talvez iterado, dessa metodologia dá ao programa uma estrutura elegante e permite a fatoração natural de várias tarefas de programação: especificação, desenvolvimento, documentação, verificação, testes, etc.

Para realmente se tirar proveito da abstração, é necessário que os tipos de dados sejam especificados de maneira formal e independente de qualquer particular representação [9]. Várias maneiras de se apresentar uma especificação formal de um TAD têm sido propostas [9]. Em particular, as chamadas especificações algébricas [4,5,6] têm assumido um papel importante nos estudos de correção da representação de dados. De acordo com esse enfoque um TAD é visto como uma álgebra definida por um conjunto de equações envolvendo suas operações.

A noção de implementação tem sido objeto de freqüentes discussões e vários esforços foram feitos no sentido de precisar-lhe o significado. Dentre os trabalhos com esse objetivo duas abordagens do problema têm merecido destaque: uma considera a implementação como uma função de abstração definida do espaço concreto no espaço abstrato; a outra considera uma função de representação que associa a cada termo do tipo a ser implementado um outro do tipo usado na implementação. A primeira abordagem foi apresentada inicialmente por Hoare [7] e se constitui numa forma mais natural se o tipo de dado é visto como classes de equivalência dos termos, uma vez que um mesmo objeto abstrato pode ter várias representações. A outra abordagem é brevemente apresentada por Goguen, Thatcher e Wagner [6], e de modo mais completo e formal por Ehrig, Kreowski e Padawitz [1]. Tal abordagem parece ser mais conveniente no contexto de sistemas de reescrita ou problemas de tradução.

Mais recentemente o problema da implementação foi estudado por Pequeno [10] . Utilizando recursos da lógica matemática ele descreve formalmente o processo de implementação como uma interpretação entre teorias.

No que se segue vamos tratar da implementação e sua correção seguindo o enfoque de [6]. Uma metodologia para verificação da correção de implementações é exibida e sua aplicação é ilustrada por meio de um exemplo. Finalmente algumas

conclusões serão apontadas.

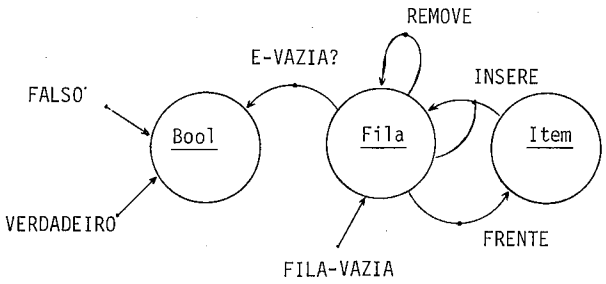
2. IMPLEMENTAÇÃO

A implementação de um TAD A em outro B (abstrato ou não) é levada a efeito em duas fases: primeiro, os objetos de A devem ser representados em função dos objetos de B e, segundo, as operações de A devem ser implementadas através de procedimentos que utilizam as operações de B.

Como exemplo, vamos considerar a implementação do TAD Fila-de-Item no TAD Vetor-de-Item. Antes, porém, apresentaremos as especifcações desses tipos, as quais descrevem o comportamento das operações respectivas, independentemente de qualquer representação.

Uma especificação (algébrica) para o TAD Fila-de-Item é a seguinte:

Sintaxe:



Semântica:

Para q: Fila ; it: Item

f1) REMOVE (INSERE(q,it)) = Se E-VAZIA?(q) então FILA-VAZIA
senão INSERE(REMOVE(q), it)

f2) FRENTE(INSERE(q,it)) = Se E-VAZIA?(q) então it senão FRENTE(q)

f3) E-VAZIA?(FILA-VAZIA) = VERDADEIRO

f4) E-VAZIA?(INSERE(q,it))= FALSO

Restrições:

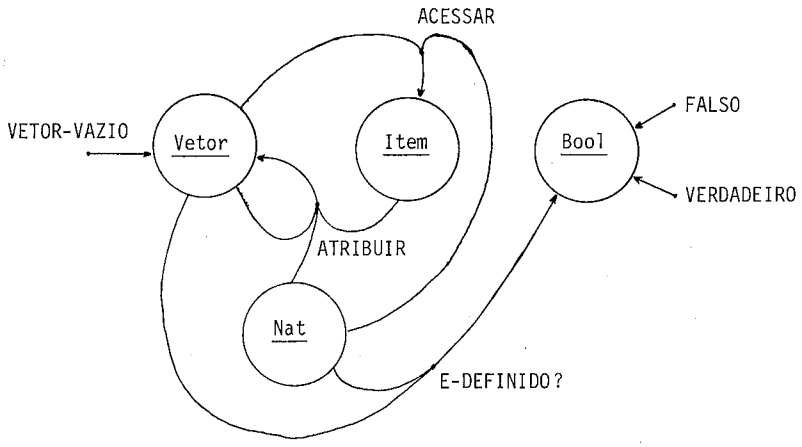
Pre(REMOVE,q) = ¬ E-VAZIA?(q)

Pre(FRENTE,q) = ¬ E-VAZIA?(q)

NOTA: As restrições inseridas na especificação do tipo Fila-de-Item indicam que as operações REMOVE e FRENTE só devem ser aplicadas a filas não vazias. Isto é uma precondição que deve ser verificada pelo usuário do tipo.

Uma especificação (álgebrica) para o TAD Vetor-de-Item é a seguinte:

Sintaxe:



Semântica:

Para v: Vetor; i,j: Nat; it,it': Item

v1) ATRIBUIR(ATRIBUIR(v,i,it),j,it') = Se i=j então ATRIBUIR(v,j,it')
senão ATRIBUIR(ATRIBUIR(v,j,it'),i,it)

v2) ACESSAR(ATRIBUIR(v,i,it),j) = Se i=j então it senão ACESSAR (v,j)

v3) E-DEFINIDO? (VETOR-VAZIO,j)= FALSO

v4) E-DEFINIDO? (ATRIBUIR(v,i,it),j) = Se i=j então VERDADEIRO
senão E-DEFINIDO?(v,j)

Restrições:

¬ E-DEFINIDO?(v,i) ⟹ Falha(ACESSAR,v,i)

NOTA: i) A restrição na especificação do tipo Vetor-de-Item indica que a aplicação da operação ACESSAR(v,i) falha quando a i-ésima coordenada do vetor v não está definida.

ii) Vamos admitir Nat como sendo os naturais com suas operações e especicação usuais.

Conforme mencionado antes, a implementação de um tipo A em outro B (por exemplo, Fila-de-Item em Vetor-de-Item) consiste na representação dos objetos de A em termos dos objetos de B e na descrição das operações de A em termos das operações de B. De maneira mais formal, a implementação de um tipo de dados A em outro B consiste basicamente em se especificar uma função de representação ρ, que associa a cada sorte s de A um sorte $\rho(s)$ pertencente ao tipo B ou construido a partir dos sortes deste, e a cada operação f de A uma operação f^o defi

nida a partir das operações de B e tal que se $f: s_1 \times s_2 \times \ldots \times s_n \to s$, então $f^{\rho}: \rho(s_1) \times \rho(s_2) \times \ldots \times \rho(s_n) \to \rho(s)$. Além disso, ρ deve traduzir na representação a relação de igualdade existente para o tipo a ser implementado. A função ρ pode ser estendida para os termos do tipo a ser implementado, do modo usual:

$$\rho(f(t_1,t_2,\ldots,t_n)) = f^{\rho}(\rho(t_1), \rho(t_2),\ldots\rho(t_n))$$

Para o exemplo proposto, a implementação do TAD Fila-de-Item no TAD Vetor-de-Item, escolhemos representar os objetos do sorte Fila através da tripla <Vetor,Nat,Nat>. O primeiro natural da tripla indica a coordenada do vetor onde começa a fila, enquanto o segundo aponta a primeira posição livre do vetor após a fila. Assim desejamos que $<[a_0,a_1,\ldots,a_{n-1}],0,n>$ represente a fila constituída pelos itens $a_0,a_1,\ldots,a_{n-1}$ nesta ordem.

Por simplicidade, limitações sobre o tamanho da fila ou do vetor não serão consideradas.

Assim, para o exemplo em questão:

Função de Representação:

Associação de Sortes:

ρ(Bool) = Bool

ρ(Item) = Item

ρ(Fila) = Vetor × Nat × Nat

Operações Definidas

Para v,v' : Vetor; i,j,i',j' : Nat; it : Item;

p1) $FALSO^{\rho}$ = FALSO

p2) $VERDADEIRO^{\rho}$ = VERDADEIRO

p3) $FILA\text{-}VAZIA^{\rho}$ = <VETOR-VAZIO,0,0>

p4) $INSERE^{\rho}$ (< v,i,j >, it) = <ATRIBUIR(v,j,it),i,j+1>

p5) $REMOVE^{\rho}$ (< v,i,j >) = <v,i+1,j>

p6) $FRENTE^{\rho}$ (< v,i,j >) = ACESSAR(v,i)

p7) $E\text{-}VAZIA^{\rho}$(< v,i,j >) = $i \stackrel{?}{=} j$

Tradução da Igualdade

r1) $\rho(=_B)$: identidade em Bool

r2) $\rho(=_I)$: identidade em Item

r3) <v,i,j> $\rho(=_F)$ <v',i',j'>:[(j-i)=(j'-i')] ∧ ∀k [0≤k<j-i=>
=>ACESSAR(v,i+k)=ACESSAR(v',i'+k)]

NOTA: i) B,I e F abreviam Bool, Item e Fila, respectivamente.

ii) Duas triplas de Vetor × Nat × Nat são idênticas se suas componentes o são segundo as especificações correspondentes.

Conforme se depreende da função de representação, os sortes Bool e Item da

linguagem de Fila-de-Item foram representados nos próprios na linguagem de Vetor-de-Item, enquanto o sorte Fila foi representado no produto cartesiano Vetor×Nat×Nat.

Cada um dos p1,p2,..,p7 pode ser visto como especificando um programa que descreve uma operação do tipo Fila-de-Item em termos das operações do tipo Vetor-de-Item. Entretanto, observe que o formalismo usado para se especificar esses programas é semelhante àquele adotado na descrição dos axiomas de uma especificação. Procedemos dessa forma para isolar as dificuldades relativas às particularidades e idiossincrasias das diferentes linguagens de programação. Ademais, por não estarem presas a uma linguagem de programação específica, a descrição do processo de implementação e sua correção adquirem um caráter mais geral.

A interpretação da igualdade, especificada por r1,r2 e r3, diz que os sortes Bool e Item têm a mesma representação tanto para Fila-de-Item como para Vetor-de-Item, e que duas triplas, <v,i,j> e <v',i',j'>, representam a mesma fila se as diferenças entre seus subscritos (tamanho da fila) são iguais, e se todos os elementos correspondentes são iguais. Está claro que $\rho(=_B)$, $\rho(=_I)$ e $\rho(=_F)$ são simétricas, reflexivas e transitivas, e dessa forma, relações de equivalência.

3. CORREÇÃO DA IMPLEMENTAÇÃO

Ficou implícito na escolha da representação do exemplo em exame que nem toda tripla < Vetor, Nat, Nat > representa uma fila. Contudo, para sermos precisos devemos explicitar uma propriedade que caracterize o conjunto das triplas que desejamos tomar como representantes para os objetos do tipo Fila-de-Item. Guttag [5] chama essa propriedade de Invariante da Representação. Para o exemplo em questão; temos:

Invariante da Representação (IR):

$$IR(<v,i,j,>) \text{ é } (i \leq j) \wedge \forall k(0 \leq k < j \Rightarrow \text{E-DEFINIDO?}(v,k) = \text{VERDADEIRO} \wedge$$
$$\wedge \forall k\ (k \geq j \Rightarrow \text{E-DEFINIDO?}(v,k) = \text{FALSO})$$

A função de representação ρ é , por construção, um homomorfismo que mapeia os termos do tipo A nos termos do tipo B que satisfazem o invariante da representação.

Como metodologia para demonstrar a correção de uma implementação ρ de um tipo A em outro B, apresentamos a seguinte:

Passo 1: Mostrar que o IR é fechado sob as implementações f^{ρ} das operações f do tipo A.

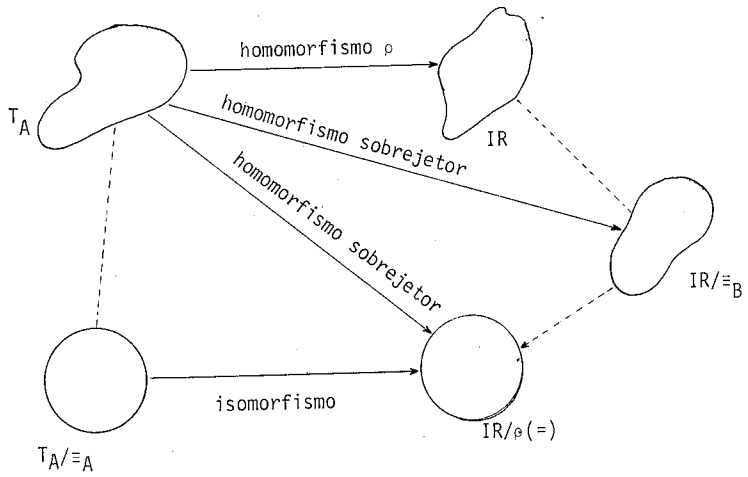
Passo 2: Mostrar que todo termo de IR é gerado ou é idêntico,pelos axiomas de B, a algum gerado pelas implementações f^{ρ} das operações de A.

Passo 3: Mostrar que a relação de equivalência induzida pelos axiomas de B restrita a IR está contida na tradução da igualdade $\rho(=)$ de A em B.

Passo 4: Mostrar que cada classe de equivalência gerada em A por seus axiomas é mapeada em uma classe de B segundo $\rho(=)$, ou seja, que os axiomas de A são preservados em B.

Passo 5: Mostrar que classes de equivalência distintas geradas em A por seus axiomas são mapeadas em classe distintas de B segundo $\rho(=)$, ou seja, $\rho(=)$ está de acordo com os axiomas de A.

A verificação desses passos determina a correção da implementação por garantir a existência de um isomorfismo apropriado, conforme ilustra gráfico abaixo:



Pelo passo 1 da metodologia as constantes (operações nulárias) de A têm representação em IR. Como qualquer objeto de A é obtido a partir destas pela aplicação das outras operações, e como o IR é fechado sob a implementação das operações de A, segue-se que todo objeto de A tem representação em IR. Por outro lado, pelo passo 2, temos que qualquer objeto do IR é gerado ou é identificado pelos axiomas de B a algum gerado pela implementação f^{ρ}. Isto significa que ρ induz um homomorfismo sobrejetor em $IR/\equiv_B$ (quociente de IR pela relação de equivalência induzida pelos axiomas de B).

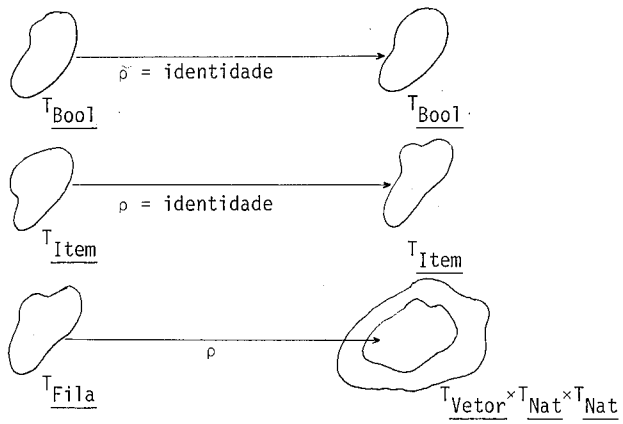
O passo 3 da metodologia diz que $\rho(=)$ está de acordo com os axiomas de B, no sentido de que os objetos de B que são idênticos segundo seus axiomas, também o são segundo $\rho(=)$. Decorre daí que ρ induz um homomorfismo sobrejetor em

$IR/\rho(=)$.

Pelo passo 4 da metodologia, temos que termos de uma mesma classe de equivalência gerada em A por seus axiomas são mapeados em termos de uma mesma classe de equivalência de IR segundo $\rho(=)$, enquanto pelo passo 5, termos pertencentes a classe distintas de $T_A/\equiv_A$ são mapeados em classes distintas de $IR/\rho(=)$. Assim, a função de representação ρ induz um isomorfismo de $T_A/\equiv_A$ em $IR/\rho(=)$.

4. APLICAÇÃO DA METODOLOGIA

A funçao de representaçao ρ especificada na seção 2 para o exemplo em análise mapeia os termos dos sortes Bool, Item e Fila nos termos dos sortes Bool, Item e Vetor×Nat×Nat, respectivamente:



É óbvio que a representação identidade de um sorte nele próprio é correta. Então, a verificação de que ρ implementa corretamente o tipo Fila-de-Item no tipo Vetor-de-Item se reduz à verificação de que os termos do sorte Fila são coerentemente representados no espaço dos termos de Vetor×Nat×Nat, isto é, os passos 1,2,3,4 e 5 da metodologia apresentada na seção anterior se verificam.

No que se segue vamos desenvolver a prova de cada um desses passos.

Passo 1: IR é fechado sob a implementação das operações de Fila-de-Item

Prova: Por indução:

1) Para FILA-VAZIA, temos que $\text{FILA-VAZIA}^{\rho} = \langle \text{VETOR-VAZIO},0,0\rangle$

Assim sendo, IR(<VETOR-VAZIO,0,0>) é (0≤0) ∧ ∀k (0≤k<0 ==>
==> E-DEFINIDO?(VETOR-VAZIO,k) = VERDADEIRO) ∧ ∀k (k≥0 ==>
==> E-DEFINIDO?(VETOR-VAZIO,k) = FALSO).
IR (<VETOR-VAZIO,0,0>) se verifica, trivialmente por axiomas dos naturais e pelo axioma v3 de Vetor-de-Item. Portanto <VETOR-VAZIO,0,0> ∈ IR.

2) Seja agora uma tripla qualquer <v,i,j> ∈ IR.
Isso significa que (i≤j) ∧ ∀k (0≤k<j ==> E-DEFINIDO?(v,k) = VERDADEIRO) ∧
∧ ∀k (k≥j ==> E-DEFINIDO?(v,k) = FALSO).

a) Para operação INSERE, temos

$$INSERE^{\rho}(<v,i,j>, it) = <ATRIBUIR(v,j,it),i,j+1>$$

Assim, o IR para tripla resultante é (i≤j+1) e
∀k(0≤k<j+1 ==> E-DEFINIDO?(ATRIBUIR(v,j,it),k) = VERDADEIRO) ∧
∧ ∀k (k≥j+1 ==> E-DEFINIDO?(ATRIBUIR(v,j,it),k)= FALSO)
Por hipótese de indução i≤j e assim é verdade que i≤j+1. De igual modo e por v4, E-DEFINIDO?(ATRIBUIR(v,j,it),k)= VERDADEIRO se verifica para 0≤k<j. Isso é válido também para k=j pelo axioma v4 do tipo Vetor-de-Item. Por outro lado, como E-DEFINIDO?(v,k) = FALSO se verifica para k≥j, por hipótese de indução, por v4 vê-se que E-DEFINIDO?(ATRIBUIR(v,j,it),k) = FALSO, para k≥j+1. Desse modo IR(INSERE$^{\rho}$(<v,i,j>, it)) se verifica, e então
INSERE$^{\rho}$ (<v,i,j,>,it) ∈ IR.

b) Para operação REMOVE, temos REMOVE$^{\rho}$(<v,i,j >) = <v,i+1,j>
Assim, IR(<v,i+1,j >) é (i+1≤j)∧ ∀k(0≤k<j ==> E-DEFINIDO?(v,k)=
= VERDADEIRO) ∧ ∀k(k≥j) ==> E-DEFINIDO?(v,k) = FALSO).
Por hipótese de indução E-DEFINIDO?(v,k) = VERDADEIRO se verifica para 0≤k<j, e de igual modo E-DEFINIDO?(v,k) = FALSO, para k≥j. Também por hipotese de indução i≤j. Porém como <v,i,j> não pode ser FILA-VAZIA$^{\rho}$ pela restrição Pre(REMOVE,q)= ¬ E-VAZIA?(q) da especificação do tipo Fila-de-Item, temos que i<j o que implica que i+1≤j. Desse modo, IR(<v,i+1,j>) se verifica e portanto REMOVE$^{\rho}$ (<v,i,j>) ∈ IR

□

O que acabamos de mostrar é que a representação de FILA-VAZIA satisfaz o invariante IR, e que se uma fila q tem sua representação <v,i,j> satisfazendo IR, a aplicação a q das operações que dão resultados no sorte Fila resulta numa nova q' cuja representação <v',i',j'> também satisfaz IR. Como qualquer fila é obtida à partir da FILA-VAZIA pela aplicação das operações INSERE e REMOVE, IR conterá a representação de todas as filas.

Passo 2: Toda tripla de IR é gerada ou é idêntica, pelos axiomas de

Vetor×Nat×Nat, a alguma gerada pela implementação das operações de Fila-de-Item.

Prova: Vamos tomar uma tripla qualquer $\langle v,i,j\rangle \in IR$. Isso significa que $(i\leq j) \wedge \forall k(0\leq k<j \Longrightarrow$ E-DEFINIDO? (v,k) = VERDADEIRO$) \wedge \forall k(k\geq j \Longrightarrow$ $\Longrightarrow$ E-DEFINIDO?(v,k) = FALSO).

Se o vetor v está definido exatamente para $0\leq k<j$ pelos axiomas v1, v3 e v4 de Vetor-de-Item, temos que:

v = ATRIBUIR(...ATRIBUIR(VETOR-VAZIO,k_0,it_{k_0})... , k_{j-1},it_{j-1}) sendo $k_0,k_1,\ldots,k_{j-1}$ uma permutação de 0,1,2,....,j-1.

Consideremos agora

$w = \rho($REMOVE(...REMOVE(INSERE(...INSERE(FILA-VAZIA,it_0)...,
(i vezes, j vezes) ,it_{j-1}))...))

Pela definição de ρ temos:

w = REMOVE$^{\rho}$(...REMOVE$^{\rho}$(INSERE$^{\rho}$(...INSERE$^{\rho}$(FILA-VAZIA$^{\rho}$,it_0)...,
,it_{j-1}))...)

Usando sucessivamente p_3, j vezes p_4 e i vezes p_5, obtemos

$w = \langle$ATRIBUIR(...ATRIBUIR(VETOR-VAZIO,0,it_0) ...,j-1,it_{j-1}),i,j$\rangle$

Agora, pela aplicação dos axiomas de Vetor-de-Item vemos que v é idêntico ao vetor que aparece na tripla w.

Conclusão: Dada uma tripla qualquer $\langle v,i,j\rangle \in IR$ ela é igual, pelos axiomas das triplas <Vetor, Nat, Nat>, à representação de alguma fila. □

Pelo passo 1 vimos que para uma fila q qualquer, $q^{\rho} \in IR$. Pelo passo 2 temos que para uma tripla $\langle v,i,j\rangle \in IR$, existe uma fila-de-item q', tal que $(q')^{\rho} \equiv_{V\times N\times N} \langle v,i,j\rangle$ ($\equiv_{V\times N\times N}$ é a relação de equivalência gerada pelos axiomas de Vetor×Nat×Nat em IR. Podemos concluir que a função de representação ρ induz um homomorfismo sobrejetor de T_F em $IR/\equiv_{V\times N\times N}$.

Passo 3: Em IR $\rho(=_F)$ está de acordo com os axiomas de Vetor-de-Item

Prova: Considere duas triplas de IR, $\langle v,i,j\rangle$ e $\langle v',i',j'\rangle$, tais que $\langle v,i,j\rangle \equiv_{V\times N\times N} \langle v',i',j'\rangle$. Devemos mostrar que $\langle v,i,j\rangle\ \rho(=_F)\ \langle v',i',j'\rangle$. Mas, $\langle v,i,j\rangle \equiv_{V\times N\times N} \langle v',i',j'\rangle$ significa $v\equiv_V v'$, $i \equiv_N i'$ e $j \equiv_N j'$, os dois últimos, pela nossa suposição sobre a especificação de Nat, equivalendo a i=i' e j=j'. Então, temos $\langle v,i,j\rangle$ e $\langle v'i,j\rangle$ em IR tais que $v\equiv_V v'$ e queremos mostrar que $\langle v,i,j\rangle\ \rho(=_F)\ \langle v',i,j\rangle$. Como $v'\equiv_V v'$ se e só se esta igualdade decorre dos axiomas de Vetor-de-Item, basta mostrar que, para cada axioma u=w de Vetor-de-Item, com u e w de sorte Vetor, se tem $\langle u,i,j\rangle \rho(=_F) \langle w,i,j\rangle$, ou seja, j-i=j-i e para p=i,...,j-1, ACESSAR(u,p) = ACESSAR(w,p). Nestas condições, temos o axioma (v1) ,

quando u é ATRIBUIR(ATRIBUIR(v,k,it),ℓ,it') e w é ATRIBUIR (v,k,it'), caso k=ℓ, ou ATRIBUIR(ATRIBUIR(v,ℓ,it'), k,it) caso contrário. Usando o axioma (v2), obtemos

	ACESSAR(u,p)	ACESSAR(w,p)	
p = ℓ	it'	it'	p = k = ℓ
		it'	p = ℓ k ≠ ℓ
p ≠ ℓ p = k	it	it	k ≠ ℓ p = k
p ≠ ℓ	ACESSAR (v,p)	ACESSAR (v,p)	p = ℓ p ≠ k
≠ k		ACESSAR (v,p)	k ≠ ℓ p ≠ k p ≠ ℓ

□

Passo 4: Os axiomas de Fila-de-Item são preservados pela representação

Prova: Considere uma fila q, com representação q^{ρ}=<v,i,j>

a) Seja o axioma f1: REMOVE(INSERE(q,it))=
Se E-VAZIA?(q)
então FILA-VAZIA
senão INSERE(REMOVE(q),it).

Pelo lado esquerdo do axioma, temos

ρ(REMOVE(INSERE(q,it))) =
= $REMOVE^{\rho}(INSERE^{\rho}(q^{\rho},it))$, pela def. de ρ
= $REMOVE^{\rho}(INSERE^{\rho}(<v,i,j>, it)$
= $REMOVE^{\rho}$(<ATRIBUIR(v,j,it),i,j+1>) , por p4
= <ATRIBUIR(v,j,it),i+1,j+1> , por p5

Pelo lado direito, temos:

Se ρ(E-VAZIA?(q)) então ρ(FILA-VAZIA)
senão ρ(INSERE(REMOVE(q),it) =
= Se $E\text{-}VAZIA?^{\rho}(q^{\rho})$ então $FILA\text{-}VAZIA^{\rho}$
senão $INSERE^{\rho}(REMOVE^{\rho}(q^{\rho}),it)$
= Se $i \stackrel{?}{=} j$ então <VETOR-VAZIO,0,0>
senão $INSERE^{\rho}$(<v,i+1,j>,it), por p7 e p5
= Se $i \stackrel{?}{=} j$ então <VETOR-VAZIO,0,0>
senão <ATRIBUIR(v,j,it),i+1,j+1> por p4

Temos dois casos a analisar:

i) Caso i=j

Lado esquerdo: <ATRIBUIR(v,j,it),i+1,j+1>
Lado direito : <VETOR-VAZIO,0,0>
Estes dois termos são idênticos segundo $\rho(=_F)$

ii) Caso i≠j
Lado esquerdo: <ATRIBUIR(v,j,it),i+1,j+1>
Lado direito : <ATRIBUIR(v,j,it),i+1,j+1>
Estes dois termos são idênticos segundo $\rho(=_F)$
Com isso concluímos que o axioma f1 é preservado.

b) Seja o axioma f2: FRENTE(INSERE(q,it)) =
= Se E-VAZIA?(q)
então it senão FRENTE(q)

Pelo lado esquerdo do axioma, temos, pela definição de ρ, por p4 e p6:
ρ(FRENTE(INSERE(q,it)) =
= ACESSAR(ATRIBUIR(v,j,it),i)
Pelo lado direito, temos, por p7 e p6:
Se ρ(E-VAZIA?(q)) então it senão ρ(FRENTE(q)) =
= Se $i \stackrel{?}{=} j$ então it senão ACESSAR(v,i)
Temos dois casos a analisar:
i) Caso i=j
Lado esquerdo: ACESSAR(ATRIBUIR(v,j,it),i) = it, por v2
Lado direito: it
Os dois termos são idênticos segundo $\rho(=_I)$.

ii) Caso i≠j
Lado esquerdo: ACESSAR(ATRIBUIR(v,j,it), i) = ACESSAR(v,i), por v2
Lado direito: ACESSAR(v,i)
Estes termos são idênticos segundo $\rho(=_I)$
Com isso concluímos que o axioma f2 é preservado.

c) Seja o axioma f3: E-VAZIA?(FILA-VAZIA) = VERDADEIRO
Pelo lado esquerdo, temos, pela definição de ρ, por p3 e p7:
ρ(E-VAZIA?(FILA-VAZIA)) = $0 \stackrel{?}{=} 0$
Pelo lado direito, temos, por p2:
ρ(VERDADEIRO) = VERDADEIRO$^{\rho}$= VERDADEIRO
O axioma se verifica, uma vez que $0 \stackrel{?}{=} 0$ é idêntico a VERDADEIRO, segundo $\rho(=_B)$.
NOTA: Estamos supondo uma especificação usual dos naturais.

d) Seja o axioma f4: E-VAZIA?(INSERE(q,it)) = FALSO
Pelo lado esquerdo, temos, pela definição de ρ, por p4 e p7:
ρ(E-VAZIA?(INSERE(q,it)) = $i \stackrel{?}{=} j+1$

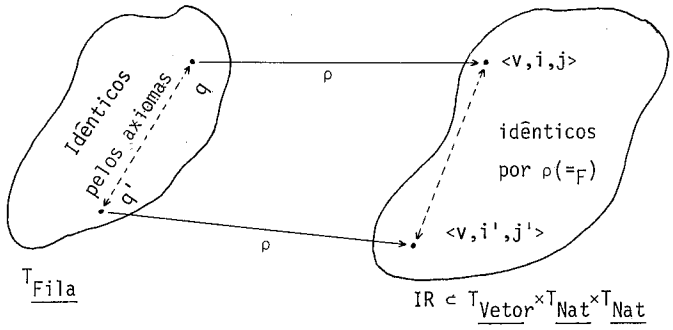
Pelo lado direito, temos, por p1:

$\rho(\text{FALSO}) = \text{FALSO}^{\rho} = \text{FALSO}$

O axioma se verifica, uma vez que $q^{\rho} \in IR$ e portanto $i \leq j$. Assim $i \stackrel{?}{=} j+1$ é idêntico a FALSO segundo $\rho(=_B)$.

□

O que acabamos de demonstrar, utilizando para isto os axiomas de Vetor-de Item, os axiomas dos naturais e a tradução das igualdades $\rho(=_F)$, $\rho(=_I)$ e $\rho(=_B)$ é que se os axiomas de Fila-de-Item identificam dois termos q e q', então q^{ρ} e $(q')^{\rho}$ são identificados em Vetor×Nat×Nat:



Passo 5: $\rho(=_F)$ está de acordo com os axiomas de Fila-de-Item

Prova: Consideremos duas filas q_1 e q_2 cujas representações sejam identificadas por $\rho(=_F)$, isto é:

$$q_1^{\rho} \quad \rho(=_F) \quad q_2^{\rho}$$

Sejam $\overline{q}_1$ e $\overline{q}_2$ representantes canônicos [5,p.123] das classes de equivalencia que contêm q_1 e q_2, respectivamente. Portanto, $q_1 \equiv_F \overline{q}_1$, $q_2 \equiv_F \overline{q}_2$. Assim, pelo demonstrado no passo 4, $q_1^{\rho} \rho(=_F) \overline{q}_1^{\rho}$ e $q_2^{\rho} \rho(=_F) \overline{q}_2^{\rho}$, donde se conclui que $\overline{q}_1^{\rho} \rho(=_F) \overline{q}_2^{\rho}$

Resumindo:

por escolha					
q_1	q_2	q_1^{ρ}	$\rho(=_F)$	q_2^{ρ}	pelo passo 4
$\equiv_F$	$\equiv_F$	$\rho(=_F)$		$\rho(=_F)$	
$\overline{q}_1$	$\overline{q}_2$	$\overline{q}_1^{\rho}$	$\rho(=_F)$	$\overline{q}_2^{\rho}$	

Os termos canônicos de Fila-de-Item são da forma:

$\overline{q}_1$ = INSERE(INSERE ... INSERE(FILA-VAZIA, a_0),...,a_n)

$\overline{q}_2$ = INSERE(INSERE ... INSERE(FILA-VAZIA, b_0),...,b_m)

Pela definição de ρ, temos

$\rho(\overline{q}_1)$ = <ATRIBUIR(...(ATRIBUIR(ATRIBUIR(VETOR-VAZIO,0,a_0), ,1,a_1)..., n-1,a_n), 0,n >

$\rho(\overline{q}_2)$ = <ATRIBUIR (...(ATRIBUIR(ATRIBUIR(VETOR-VAZIO,0,b_0), ,1,b_1) ...,m-1, b_m),0,m >

Pela definição de $\rho(=_F)$, se $\overline{q}_1 \ \rho(=_F) \ \overline{q}_2^{\rho}$, necessariamente temos que

$(n=m) \wedge \forall k \ (0 \leq k < n \Rightarrow ACESSAR(v,k) = ACESSAR(w,k))$, onde

v = ATRIBUIR(...(ATRIBUIR(ATRIBUIR(VETOR-VAZIO,0,a_0), 1,a_1)...,n-1,a_n)

w = ATRIBUIR(...(ATRIBUIR(ATRIBUIR(VETOR-VAZIO,0,b_0), 1,b_1)...,m-1,b_m)

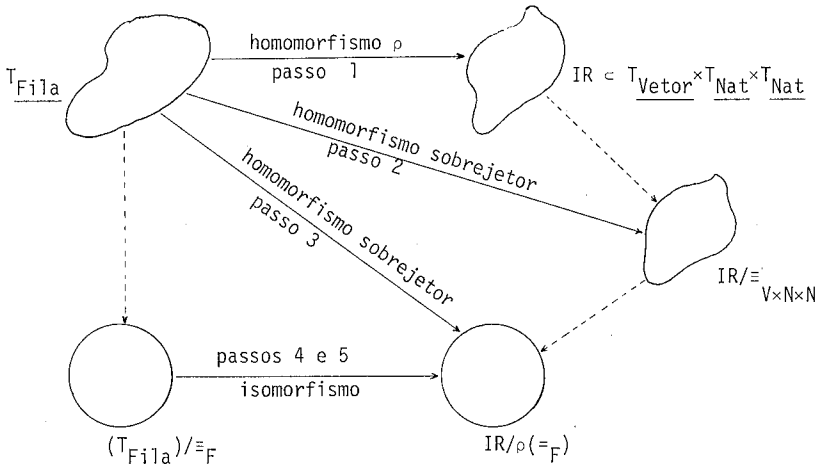
Com m=n e $a_k = b_k$, para k=0,1,...,n, temos que $\overline{q}_1$ e $\overline{q}_2$ são a mesma fila e assim $\overline{q}_1 \equiv_F \overline{q}_2$, e portanto $q_1 \equiv_F q_2$.

Conclusão: $\rho(=_F)$ não identifica termos que não sejam representações de filas identificadas por $\equiv_F$

□

Como já notamos na seção 3, esses cinco passos garantem a existência de um isomorfismo entre $(T_{Fila})/\equiv_F$ e $IR/\rho(=_F)$, e desse modo, que a implementação do tipo Fila-de-item em Vetor-de-Item é correta.

Resumindo graficamente, o efeito de cada passo nessa demonstração, temos:



5. CONCLUSÕES

Visando clarificar aspectos de correção, reexaminamos a noção de implementação de um TAD A em outro B. Em particular apresentamos e justificamos

uma metodologia para verificar a correção de implementações e a ilustramos por meio de um exemplo.

Tentamos, na metodologia apresentada, sistematizar os vários passos da verificação de maneira mais ou menos modular em relação ao que se usa em cada passo. Assim é que no passo 1 são usados apenas os axiomas do tipo B e as pre condições do tipo A, além da definição do invariante. Por outro lado, os axiomas de A somente são usados no passo 5, juntamente com os B e a definição de $\rho(=)$.

Outra observação pertinente se refere às exigências dos vários passos. Por exemplo, no passo 2 da metodologia apresentada, exigimos que os termos de IR sejam gerados ou identificados pelos axiomas de B a algum termo gerado pela implementação das operações de A. Isso, contudo, não é estritamente necessário. O que realmente importa é cada uma das classes de equivalência de $IR/\rho(=)$ ter pelos menos um termo gerado pela implementação das operações de A. Por exemplo, a demonstração apresentada na seção 4 não seria muito diferente se o IR escolhido fosse um pouco mais amplo, como o seguinte:

$$(i \leq j) \wedge \forall k\ (i \leq k < j) \Longrightarrow \text{E-DEFINIDO?}(v,k) = \text{VERDADEIRO})$$

que deixa em aberto E-DEFINIDO? (v,k) para k fora do intervalo [i,j-1].

Finalmente devemos ressaltar que esta metodologia foi desenvolvida tendo em vista a conceito de implementação baseada em função de representação. Não parece muito difícil adaptá-la para o caso de implementação baseada em função de abstração. Porém é importante ter-se em mente que tudo o que foi dito se refere a TAD'S vistos como álgebras iniciais. A noção de implementação para outros enfoques de TAD'S será objeto de outros estudos.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] H. Ehrig, H.,J. Kreowski, P. Padawitz - Some Remarks Concerning Correct Specification and Implementation of Abstract Data Types; *Informatik Research Report* nº 77 - 13, Technical University Berlin, 1977.

[2] F. E. P. Pessoa - Programação com Tipos de Dados Abstratos; Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Dissertação de Mestrado, 1981.

[3] M.C. Gaudel - Algebraic Specification of Abstract Data Types; IRIA Rapport de Recherche nº 360, Rocquencourt, 1979.

[4] J.V. Guttag, J. J. Horning - The Algebraic Specification of Abstract Data Types; *Acta Informatica*, vol. 10, nº 1, p. 27-52, 1978.

[5] J.V. Guttag, E.Horowitz, D.R. Musser-Abstract Data Types and Software Validation; *Comm. ACM*, vol. 21, nº 12, 1978.

[6] J. A. Goguen, J. W. Thatcher, E.G. Wagner - An Initial Algebra Approach to the Specification, Correctness, and implementation of Abstract Data Types; em R.T. Yeh (ed.) *Current Trends in Programming Methodology*, vol. IV, Prentice-Hall, Englewood Cliffs, 1978.

[7] C.A.R. Hoare - Proof of Correctness of Data Representations; *Acta Informatica*, 1, p. 271-281, 1972.

[8] B. Liskov, S. Zilles - Programming With Abstract Data Types; Proc. of a Symp. on Very High Level Languages, *SIGPLAN*, Notices Vol. 9, nº 4, p. 50-59, 1974.

[9] B. Liskov, S. Zilles - Specification Techniques for Data Abstraction; *IEEE Transactions on Software Engineering*, vol. 1, nº 1, p. 7-19 , 1975.

[10] T.H.C. Pequeno - Uma Descrição Formal dos Processos de Especifica - ção e Implementação de Tipos Abstratos de Dados; Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro , Tese de Doutorado, 1981.