Por uma formalização de estruturas de dados

Autor: Marlo Vieira dos Santos e Souza*

Orientador: Armando Luiz Andrade Peixoto**

Resumo

Apresentaremos uma formalização para algumas estruturas de dados utilizando a teoria ingênua dos conjuntos, grafos orientados, e álgebra para representar as estruturas e suas semânticas através de uma abordagem axiomática.

1. Introdução

Estruturas de dados (ou ED) é um conceito central em Ciência da Computação quando tratamos de programação. Uma ED consiste de um conjunto de valores e determinadas operações permitidas sobre estes e possuem uma estrutura interna intrinsecamente ligada a ambos. A formalização desses conceitos faz-se útil, pois nos fornece meios de provar a corretude, complexidade e outras propriedades de algoritmos que os manipulam, ou simplificar estas provas. Devido à pluralidade de estruturas existentes não foi possível ainda uma formalização que englobasse perfeitamente todas elas preservando sua semântica, porém trabalhos como o de EALEY [1971] nos fornece uma base para tentativas posteriores, focando primeiramente na semântica da estrutura e como representá-la.

Pelo fato de estruturas de dados serem conjuntos dinâmicos [CORMEN *et al*, 2002], isto é, em constante mudança, temos algumas dificuldades em representá-las perfeitamente utilizando teoria dos conjuntos, além disso, representar a estrutura interna (definida por qual elemento aponta para o outro) de forma satisfatória é também pouco usual com a abordagem algébrica que tomamos. Para lidar com esse problema de sincronização de

*Bacharelando em Ciência da Computação na Universidade Salvador – UNIFACS cursando o 4º semestre.
**M. Sc. em Matemática pela Universidade Federal da Bahia, professor titular na Universidade Salvador – UNIFACS e professor Adjunto na Universidade do Estado da Bahia – UNEB.

conceitos foi necessário tomarmos um conjunto de primitivas e axiomas. Utilizamos ainda nesse trabalho o modelo de grafo proposto por Furtado [FURTADO 1973] onde um grafo é uma tripla (V,A,*psi*), onde V é um conjunto de vértices, A de arestas e *psi* uma função que a cada aresta de A associa uma dupla (v_1,v_2) pertencente a VxV. Tomamos aqui uma abordagem axiomática para desenvolver o conteúdo proposto.

A seção 2 descreve a semântica de cada estrutura abordada no artigo, a seção 3 apresenta a formalização desses conceitos e na seção 4 estão descritos alguns resultados obtidos a partir desta formalização.

2. Semântica das estruturas

A semântica de uma estrutura de dados nada mais é que a forma como esta pode ser acessada e modificada, consiste basicamente em quais operações são possíveis sobre aquele conjunto de dados. Utilizaremos, na seção 3, conjuntos e transformações sobre estes para representar os dados e as operações e, através de grafos orientados, uma representação adequada para esta estrutura.

Em qualquer estrutura de dados as duas operações básicas necessárias são a inserção de elementos e a remoção destes do conjunto. O que define qual estrutura estamos tratando é a maneira como esta inserção ou esta remoção é feita, assim segue uma lista das estruturas abordadas no trabalho e como estas duas operações são realizadas sobre esta estrutura:

- Lista é um conjunto de elementos L[1], L[2],...,L[n] onde *n* é o número de elementos do conjunto em que a inserção ou a remoção pode ser realizada. Dados dois elementos de uma lista, de um dos dois sempre é possível acessar o outro.
- Lista linear é uma lista onde não podem ocorrer ciclos, ou seja, através da operação *prox* não se passa duas vezes no mesmo elemento da estrutura.
- Lista circular é uma lista onde de qualquer elemento da estrutura é possível acessar qualquer outro.

- Pilha é uma lista linear onde a inserção e remoção de dados acontecem em somente um extremo, ou seja, somente os dados mais recentemente inseridos podem ser removidos.
- Fila é uma lista linear onde a inserção e a remoção só podem ser feitas nos extremos opostos da fila, ou seja, só podemos remover os dados menos recentemente inseridos no conjunto.
- Uma árvore A é um conjunto finito de elementos onde um dos nós é denominado raiz e os demais são particionados em conjuntos disjuntos $A_1,...,A_n$, onde cada A_i é uma árvore.

3. Formalização

Usamos as seguintes primitivas para o nosso trabalho:

1. Um elemento 'a' seguir um determinado tipo de dado;
2. Um elemento 'a' ser raiz de um conjunto;
3. A relação de aponta, representada por ->;

Consideremos durante o trabalho D como o conjunto dos elementos que seguem um determinado tipo de dados e U como a família de todos os subconjuntos de D. Consideremos também G, um subconjunto finito e não vazio de D, contido em U.

A1) Axioma: Seja *x* pertencente a D. Se *x* é raiz de G então *x* pertence a G.

A2) Axioma: Todo G tem uma raiz.

A3) Axioma: Para todo elemento pertencente a G, raiz(G) aponta para ele.

D1) Definição: Seja raiz(G) uma função t: de U em D , onde t(*x*) = *y* se, e somente se, *y* pertence a *x* e *y* é raiz de *x*.

D2) Definição: Seja *insere* uma função definida por i: de UxU em U, onde i(*x*,*y*)=*x*U*y*.

D3) Definição: Seja *remove* uma função definida por r: de UxU em U, onde $r(x,y)=x- y$.

A4) Axioma: Seja x pertencente a G, se x -> y então y pertence a G.

A5) Axioma: Existe *fi* uma aplicação que define um isomorfismo entre U e F, a família dos grafos orientados, de forma que:

Seja G pertencente a U e G' (V,A, *psi)* pertencente a F, o isomorfo de G por *fi*, *fi* define uma bijeção h entre os elementos de G e os elementos de V. Sejam x,y pertencentes a G com x e y, se x -> y então existe *ai* pertencente a A, tal que *psi*(*ai)*=(h(x),h(y)).

Observação 1) Note que apesar de *fi* ser um isomorfismo entre um conjunto e um grafo orientado, usaremos *fi(x)*, com x pertencente a G, para designar a bijeção definida por *fi* entre os elementos do conjunto e os vértices do grafo.

D4) Definição: Sejam x e y pertencentes a G. Dizemos por x >> y que no grafo G', isomorfo a G, *fi*(y) é atingível de *fi*(x). Denotamos por x>\>y que *fi*(y) não é atingível de *fi*(x).

Observação 2) Por atingível entendemos que no grafo existe uma sucessão de vértices adjacentes, que tem como extremidades o vértice a e o vértice b, seguindo a orientação.

D5) Definição: Seja E um subconjunto finito de D. Definimos dimensão de E ou dim(E) como a quantidade de elementos contidos em G.

Observação 3) Note que por hipótese E é um conjunto finito, logo dim(E) é um número natural, sendo nulo somente se E for o conjunto vazio.

D6) Definição: Chamamos Lista um conjunto G em que: para todo x, y e z

pertencente a G => *x*>>*y* ou *y*>>*x* ou *x*=*y* e, se *x*-> *y* e *z*-> *y* então *x*=z.

D7) Definição: Chamamos Lista Linear o conjunto G onde: G é uma Lista e para todo *x* pertencente a G, se *x*>>*y* => y>\>x.

D8) Definição: Chamamos Enfileira (X,Y) a função: Ef: UxU-> U, com (*x*,*y*) relacionando-se com *z* = *insere* (*x*,*y*), onde raiz(*z*) = raiz (*x*) e para todo *a* pertencente a *x*, se não existir um *b* de *x* tal que *b* -\> *a* em *x*, então *a* -> raiz(*y*) em *z*.

D9) Definição: Chamamos Empilha (X,Y) a função: Ep: UxU->U, em que (*x*,*y*) relacionam-se com *z* = *insere* (*x*,*y*), onde raiz(*z*) = raiz (*y*), se a>>b em *x* (ou *y*) então a>>b em *z* e para todo *a* pertencente a *x*, se não existir um *b* de x tal que *b* -\> *a* em x, então *a* -> raiz(*y*) em *z*.

D10) Definição: Chamamos Retira (X,Y) a função: R: UxU->U, com (*x*,*y*) relacionando-se com *z* , onde:

- *z* = *x*, se *y* não está contido em *x* ou se existe *a* pertencente a *x* tal que para *b* pertencente a *y*, *b*>\>*a*;

- *z* = *remove*(*x*,*y*), caso contrário.

D11) Definição: Chamamos Pilha a tripla <G,Empilha,retira>, onde G é uma Lista Linear.

D12) Definição: Chamamos Fila a tripla <G, Enfileira, retira>, onde G é Uma Lista Linear.

D13) Definição: Dizemos árvore para o conjunto G, tal que seja raiz(G)=r e x_i, com i um numero natural, de modo que r->x_i e E_i={p pertencente a G | x_i >> p}:

- i é diferente de j => a interseção entre E_i e E_j é vazia;

- Para todo i natural, $\{x_i\}$ U E_i é arvore de raiz x_i;

- Para todo i natural, a interseção entre $\{x_i\}$ e E_i é vazia.

D14) Definição: Chamamos Lista Circular o conjunto G, em que G é Lista e para todo *x*,*y* de G, *x*>>*y* e *y*>>*x*.

D15) Definição: Seja *r* uma relação de ordem sobre G, um subconjunto finito e não vazio de D. Dizemos que G_r é o conjunto G ordenado por *r* de modo que, para todo *x*,*y* de G:

- se *x* precede estritamente *y* por *r* => *x*>>*y*.

D16) Definição: Dizemos que *x* e *y* pertencentes a G são irmãos se, e somente se, existe *p* em G tal que *p*->*x* e *p*->*y*.

4. Resultados

P1) Proposição – ED: Dada U, a família de todos os subconjuntos finitos de D e a operação *insere*. <U,*insere*> é um monóide comutativo.

Demonstração: Sejam *x*,*y*,*z* pertencentes a U.

1) *insere*(*insere*(*x*,*y*),*z*) = *insere*(*x*,*y*) U *z* = (*x*U*y*)Uz = *x*U(*y*Uz) = *x* U Insere(*y*,*z*) = *insere*(*x*,*insere*(*y*,*z*)). Logo, *insere* é associativo.

2) *insere*(*x*,*y*) = *x*U*y* = *y*U*x* = *insere*(*y*,*x*). Daí *insere* é comutativo em G.

3) Tomando *e* o conjunto vazio, *insere*(*x*,e) = *x*U*e* = *x*. Logo *insere* possui elemento neutro.

Observação 4) Como <G,*insere*> é monóide, garantimos que o elemento neutro é único.

P2) Proposição – Listas Lineares: Seja G uma Lista Linear e G' seu grafo isomorfo. G' não possui ciclos.

Demonstração: Tomemos G'=(V,A,*psi*), um grafo orientado isomorfo a G, com G Lista Linear, tendo pelo menos um ciclo. G' tem ciclo => Existe *x* em V, tal que:

psi (ai) = (*x*,p_1), *psi* (a_j) = (p_1,p_2) ,..., *psi* (a_k) = (p_n, *x*).

Daí, *fi*(*x*) >> *fi*(p_1) e *fi*(p_1) >> *x*, porém G é Lista Linear por hipótese, logo isto é um absurdo e G' não possui ciclos.

P3) Teorema – Listas Lineares: Seja G um subconjunto finito de D e *r* uma relação de ordem total sobre G. G_r é Lista Linear .

Demonstração: Seja a hipótese de P3 válida. Como *r* é relação de ordem total, para todo *x*, *y* de G, *x* precede *y*, ou *y* precede *x* ou *x* = *y*. Logo, para todo *x*, *y* de G, *x*>>*y* ou *y*>>*x* ou *x*=*y*. Se *x*->*y* e *z*-> *y*, *x* = *z*, pois:

1. *x*-> *y* => não existe p em G/ *x*>> p >>*y*.

2. *z* -> *y* => não existe p em G/ *z*>> p >>*y*.

Como *r* é relação de ordem <u>total</u> *x*>>*z* ou *z*>>*x* ou *z*=*x*. Temos que *z* ->*y* => *z*>>*y* e *x*->*y* => x>>*y*. Daí, e de 1. e 2., inferimos que *z*=*x*. Assim, provamos que G_r é Lista. Temos que provar agora que *x*>>*y* => *y*>\>*x*. Tomando G_r não linear, por absurdo, temos:

Existem *x*,*y* em G_r, tal que: *x*>>*y* e *y*>>*x* => *x* precede estritamente *y* e *y* precede estritamente *x*. Absurdo! Logo *x* >>*y* => *y*>\>*x*. Daí, G_r é Lista Linear.

Observação 5) Note que P3 nos fornece um método de construção de listas lineares (e conseqüentemente de pilhas e filas, já que estas são também listas lineares). Podemos conseguir uma lista aplicando um algoritmo de ordenação sobre uma chave ordenável do dado em qualquer estrutura de dados e é possível torná-la linear.

P4) Proposição – Pilhas: <U, empilha> é monóide.

P5) Proposição – Filas: <U,enfileira> é monóide.

Observação 6) Omitiremos as demonstrações de P4 e P5 por serem demasiado grandes e trabalhosas, usaremos somente o resultado para afirmar que o elemento neutro de ambas as operações são únicos e iguais entre si e ao da operação *insere*, ou seja, o conjunto vazio.

P6) Proposição – Árvores: Seja G uma árvore e G', o grafo isomorfo a G. G' não possui ciclos.

Demonstração: Tomemos por absurdo G' =(V,A,*psi*) contendo um ciclo em *x* pertencente a V. Seja E_i={*fi*(p) | *fi*(*x*) >> *fi*(p)}. G' contém ciclo em *x* => Existe *fi*(*y*) em Ei tal que *fi*(*y*) >> *fi*(*x*) => *fi*(*x*) pertence a Ei, porém a interseção entre {*fi*(*x*)} e Ei é vazia por hipótese. Logo, G' não possui ciclos.

P7) Proposição: Se *x* pertence a G, então {*x*} U{p de G | *x*>>p} é árvore de raiz *x*.

Demonstração: Se *x* pertence a G, raiz(G) >> *x* => raiz(G) -> p1 -> p2 ->...->pn->*x*.
G é árvore e Raiz(G) -> p1 => {p1} U {p de G | p1>>p} é árvore.
Repetindo o pensamento acima n vezes, temos que {*x*} U {p de G | *x*>>p} é árvore.

Observação 7) P7 nos garante que a recursividade em árvores é possível, pois todo nó em sua estrutura é raiz de uma sub-árvore própria.

P8) Proposição – Lista Circular: Seja G uma lista circular e dim(G) = n, existem infinitas funções f_i, com i número natural, que definem uma bijeção sobre G mantendo sua estrutura, isto é, x,y pertencem a G, se x->y então f(x)->f(y).

Demonstração: Seja G = {x_0, x_1,x_2,...,x_{n-1}} disposto de uma forma que x_i->x_{i+1} e f_i uma função sobre G, onde $f_i(x_j)=x_{i+j(modn)}$.

- Fato : x_{n-1} -> x_0
- Prova: Por definição de lista circular para todo x,y x>>y e y>>x.

Como para todo i natural, x_i -> x_{i+1}, x_{n-1} >> x_0 e não existe i > n-1 tal que x_i pertença a G pois dim(G) = n, então x_{n-1} -> x_0.

$f_i(x_j) = f_i(x_k)$ => $x_{i+j \text{ (mod n)}} = x_{i+k\text{(mod n)}}$ => i+j (mod n) = i+k (mod n), como j<n e k<n => j = k. Logo f_i é injetora.

$f_i(x_j) = x_{i+j\text{(mod n)}}$, i+j(mod n) está entre 0 e n-1, logo para qualquer x_j de G, existe x_k tal que i+k(mod n)=j. Logo fi é sobrejetora, e portanto bijetora.

Se j+i for menor que n, então $x_{j+i\text{(mod n)}} = x_{j+i}$, senão $x_{j+i} = x_p$, onde j+i= kn +p. Se (j+i)+1 for menor que n, então $x_{j+i+1\text{(mod n)}} = x_{(j+i)+1}$, senão $x_{j+i+1} = x_q$, onde j+i+1= kn + q, como j+i = kn+p, q = p+1. Logo, $f_i(x_i)$ -> $f_i(x_{i+1})$. Falta provar que $f_i(x_{n-1})$ -> $f_i(x_0)$.

$f_i(x_{n-1}) = x_{n-1+i\text{(mod n)}} = x_p$, onde n-1+i = kn +p e $f_i(x_0) = x_{i\text{(mod n)}} = x_q$, onde i = an +q. Se i for maior que 0 e não for múltiplo de n a>0 e q>0, i = an +q e n-1+i = kn +p => n-1+an +q = kn +p => p = (a-k)n + q-1 => $f_i(x_{n-1}) = x_{q-1}$, logo $f_i(x_{n-1})$ -> f(x_0). Caso i seja 0 ou multiplo de n, q=0 e p=n-1. Daí f_i mantém a estrutura de G.

P9) Lema – Listas Circulares: Para todo i natural. $f_i = f_k$ pertencente a {f_0,...,f_{n-1}}.

Demonstração: i+j(mod n) = (i+an)+j (mod n) =p, onde i+j =kn + p, pois (Z_n,+) é um grupo cíclico. Daí, $f_i(x_j)=f_{i+an}(x_j)$. Se i<n então f_i pertence a {f_0,...,f_{n-1}}, senão existe l maior ou igual a 0 e menor que n tal que $f_i = f_l$.

Observação 8) P8 e P9 nos garante liberdade para tratar qualquer elemento de uma lista circular como sua raiz pois qualquer elemento pode ser atingido de um outro.

5. Conclusão

Apesar de não apresentar nenhum resultado expressivo no tratamento das estruturas de dados, o presente trabalho fornece um meio de trabalhar com as estruturas através de argumentos algébricos, além de nos permitir uma abordagem diferente para provar algoritmos que atuem sobre essas estruturas. Apesar de não tratarmos de estruturas mais complexas como árvores AVL ou árvores B, é possível modelarmos essas com base no que já produzimos, acrescentando-se alguns axiomas ou primitivas quando necessário.

6. Referências

BIRKHOFF, A. ; MACLANE, S. Álgebra Moderna Básica. 4ª edição. Editora Guanabara Dois, Rio de Janeiro – RJ, 1980.

BRASSARD, Gilles & BRATLEY, Paul. Fundamentals of Algoritmics. Prentice-Hall. 1996.

CORMEN, T.H.; LEISERSON, R.L. Algoritmos. Editora Campus, Rio de Janeiro – RJ, 2002.

EARLEY, J. Toward an understanding of data structures. *In*: Communications of the ACM Vol. 4 Number 10. 1971.

FURTADO, A. L. Teoria dos Grafos : algoritmos. Editora Livros Técnicos e Científicos – LTC, Rio de Janeiro – RJ, 1973.

IEZZI, G.; DOMINGUES, H. H. Álgebra Moderna. 2ª edição. Editra Atual, São Paulo – SP, 1982.

LUCCHESI, C. L. *et al*. Aspectos Teóricos da Computação.Editora Livros Técnicos e Científicos – LTC, Rio de Janeiro, 1979.

MONTEIRO, L. H. J. Iniciação às estruturas algébricas. Livraria Nobel, São Paulo – SP, 1971.

BOAVENTURA NETTO, P. O. Grafos : Teoria, modelos, algoritmos. 2ª edição. Editora Edgar Blücher, São Paulo – SP, 1996.

SHOKRANIAN, S.; SOARES, M.; GODINHO, H. Teoria dos Números. 2ª edição. Editora UNB, Brasília – DF, 1998.