Teoria da Computação

Aula Teórica 2: Conjuntos, Relações e Funções.
Definições indutivas de conjuntos e funções
Provas de propriedades dos conjuntos e funções por indução

António Ravara

Departamento de Informática

6 de Março de 2018

Menor conjunto gerado por regras

Forma "construtiva" de definir conjuntos

- Diz-se primeiro ("axiomatiza-se") quais são os elementos "básicos" do conjunto (em número *finito*).
- Dão-se depois "regras" (em número *finito*) para obter novos elementos a partir dos que já estão no conjunto.
- Com um número finito de axiomas e regras é possível definir conjuntos com um número *infinito* de elementos.
- Para provar propriedades dos (elementos dos) conjuntos definidos, basta inspeccionar os casos usados na definição — faz-se uma prova por *indução*.

Menor conjunto gerado por regras

- O conjunto resultante contém *todos* os elementos que se podem gerar com as regras, e *apenas* esses.
- Em geral, o método gera conjuntos *infinitos*, ditos *indutivos* ou definidos por indução.
- A aplicação das regras permite obter um número infinito de elementos (a partir de um número finito de axiomas e regras).
- Cada elemento do conjunto tem no entanto uma justificação, prova ou *derivação finita*: a sequencia das regras aplicadas para o obter (começando por um axioma).
- Se dado elemento pertence a um conjunto, encontra-se facilmente uma prova de tal facto; senão, tem que se ver que regras foram violadas.

Os naturais - definição indutiva sintática

NAT é um conjunto indutivo

$$\begin{array}{ll} \mathrm{ZERO}: & \mathit{zero} \in \mathit{NAT} \\ \mathrm{SUCC}: & (n \in \mathit{NAT}) \longrightarrow \mathit{suc}(n) \in \mathit{NAT} \end{array}$$

- A regra "semente" (o axioma) ZERO introduz no conjunto *NAT* o símbolo *zero*
  Denota o valor – abstracto – 0, do conjunto "semântico" $\mathbb{N}_0$.
  Define a "matéria prima" para construir naturais .
- A regra "de construção" SUCC introduz um novo termo no conjunto *NAT*, juntando a um natural arbitrário a palavra *suc*.
  O termo (sintáctico) $\mathit{suc}(n)$ denota o valor $n + 1$ de $\mathbb{N}_0$.
  A regra define a "máquina" de produção de naturais.

Um tipo de dados indutivo

O conjunto de todas as sequências finitas de naturais, *SEQ*

Este conjunto não se consegue definir por enumeração ou compreensão/separação.

$$\begin{array}{ll} \text{EMPTY}: & () \in SEQ \\ \text{ONEMORE}: & (s \in SEQ \wedge x \in \mathbb{N}_0) \longrightarrow (x, s) \in SEQ \end{array}$$

- A regra "semente" (o axioma) EMPTY introduz no conjunto *SEQ* a sequência vazia (representada pelo conjunto vazio).
- A regra "de construção" ONEMORE introduz uma nova sequência no conjunto *SEQ*, juntando um natural arbitrário como primeiro elemento numa sequência que já está no conjunto.

Outro tipo de dados indutivo

As pilhas de naturais

- O conjunto tem como elementos sequências de valores naturais sobrepostos.
- Fixa-se primeiro que *vazia* é pilha; diz-se depois como obter uma nova pilha a partir de um natural e de uma pilha já gerada.

$$\begin{array}{ll} \text{VAZIA}: & \mathit{vazia} \in \mathit{PILHANAT} \\ \text{PUSH}: & (p \in \mathit{PILHANAT} \wedge x \in \mathbb{N}_0) \longrightarrow \mathit{push}(x, p) \in \mathit{PILHANAT} \end{array}$$

Termos e valores

Sintaxe *vs.* semântica

A definição indutiva das pilhas de naturais introduziu uma (nova) linguagem (formal) para falar de valores abstractos: gera linguagem infinita de palavras finitas.

Examplo

O termo $push(1, push(0, vazia))$ denota o valor $\begin{array}{|c|} 1 \\ 0 \\ \hline \end{array}$

Como ver se dado elemento está num conjunto?

Justificações de pertença a conjunto

São cadeias (finitas) de aplicações de regras que mostram como se gerou o elemento.

Justificação "Top-down"

- $\mathit{push}(1, \mathit{push}(0, \mathit{push}(1, \mathit{vazia}))) \in \mathit{PILHANAT}$,
  pois $1 \in \mathbb{N}_0$, e $\mathit{push}(0, \mathit{push}(1, \mathit{vazia})) \in \mathit{PILHANAT}$ (regra *PUSH*);
- $\mathit{push}(0, \mathit{push}(1, \mathit{vazia})) \in \mathit{PILHANAT}$,
  uma vez que $0 \in \mathbb{N}_0$ e $\mathit{push}(1, \mathit{vazia}) \in \mathit{PILHANAT}$ (regra *PUSH*);
- $\mathit{push}(1, \mathit{vazia}) \in \mathit{PILHANAT}$,
  já que $1 \in \mathbb{N}_0$ e $\mathit{vazia} \in \mathit{PILHANAT}$ (axioma *VAZIA*).

Partir ou terminar nos axiomas

Justificações de pertença de um valor a um conjunto

- Bottom-up: parte-se dum axioma e mostra-se como chegar ao valor, usando regras;
- Top-down: parte-se do valor e mostra-se como foi construindo usando regras, até chegar a um axioma.

$(2, 4) \in SEQ$

- $() \in SEQ$ pela regra (axioma) EMPTY;
- $(4, ()) \in SEQ$ aplicando a regra ONEMORE a $() \in SEQ$ e $4 \in \mathbb{N}_0$. Note-se que assumimos a "simplificação" $(4, ()) = (4)$;
- $(2, (4, ())) \in SEQ$ aplicando a regra ONEMORE a $(4, ()) \in SEQ$ e $2 \in \mathbb{N}_0$. Note-se que $(2, (4, ())) = (2, 4)$.

Sistemas dedutivos

As regras de dada definição indutiva podem ser apresentadas como regras de prova (também ditas de derivação ou de inferência) de um sistema dedutivo.

árvores de prova

Sendo $P, P_1, \ldots, P_n$ asserções, para um natural $n \geq 0$, uma regra escreve-se na forma

$$\frac{P_1 \;\; \cdots \;\; P_n}{P}$$

As asserções $P_1, \ldots, P_n$ dizem-se as *hipóteses*, e a asserção P a *tese* ou conclusão, da regra.
Se dada regra não tem hipóteses (caso $n = 0$), diz-se um *axioma*.

Provas como árvores etiquetadas

Derivação

- Cada árvore é construida a partir de árvores singulares (ou folhas) resultantes de axiomas da definição indutiva.
- Obtém-se um novo nível da árvore por aplicação de uma regra de inferência.
- A etiquetas dos nós são elementos do conjunto, sendo o elemento na raíz a conclusão da prova.

Diz-se que a árvore é uma derivação desse elemento.

Exemplo de árvore de prova

$suc(suc(0)) \in NAT$

$$\cfrac{\cfrac{\cfrac{}{zero \in NAT}\ (ZERO)}{suc(zero) \in NAT}\ (SUCC)}{suc(suc(zero)) \in NAT}\ (SUCC)$$

Outro exemplo de árvore de prova

$push(0, push(1, vazia)) \in PILHANAT$

$$
\dfrac{\dfrac{}{0 \in \mathbb{N}_0}\ (ZERO) \qquad \dfrac{\dfrac{\dfrac{}{0 \in \mathbb{N}_0}\ (ZERO)}{1 \in \mathbb{N}_0}\ (SUCC) \qquad \dfrac{}{vazia \in PILHANAT}\ (VAZIA)}{push(1, vazia) \in PILHANAT}\ (PUSH)}{push(0, push(1, vazia)) \in PILHANAT}\ (PUSH)
$$

Definição indutiva de funções

A ideia

- Uma função é um conjunto de pares ordenados, logo pode ser definida indutivamente.
- Usam-se os casos que serviram para definir o conjunto de valores que constitui o domínio da função:
  Utiliza-se a *recursão*:
  - Os axiomas da definição indutiva são os *casos base* (axiomas);
  - As regras são os *passos*.

Número de elementos de uma pilha de naturais

Função tamanho

Tam $\in$ *PILHANAT* $\rightarrow \mathbb{N}_0$

- Tam$(vazia) = 0$
- Tam$(push(n, p)) = 1+$Tam(p)

$$
\begin{array}{rcl}
Tam(push(7, push(4, vazia))) & = & 1 + Tam(push(4, vazia)) \\
 & = & 1 + 1 + Tam(vazia) \\
 & = & 1 + 1 + 0 \\
 & = & 2
\end{array}
$$

Sopreposição de duas pilhas

O topo da da esquerda é o topo da resultante.

$Sobrepoe \in PILHANAT^2 \rightarrow PILHANAT$

Sejam $n \in \mathbb{N}_0$ e $p, q \in PILHANAT$.

- $Sobrepoe(vazia, p) = p$
- $Sobrepoe(push(n, p), q) = push(n, Sobrepoe(p, q))$

$$\begin{aligned}
Sobrepoe(push(7, push(4, vazia)), push(6, push(2, push(5, vazia)))) &= \\
push(7, Sobrepoe(push(4, vazia), push(6, push(2, push(5, vazia))))) &= \\
push(7, push(4, Sobrepoe(vazia, push(6, push(2, push(5, vazia)))))) &= \\
push(7, push(4, push(6, push(2, push(5, vazia)))))
\end{aligned}$$

Inversão de uma pilha

Passa-se o topo para a base, inverte-se o resto, e sobrepõe-se a inversão na base.

Inverte $\in$ *PILHANAT* $\rightarrow$ *PILHANAT*

Sejam $n \in \mathbb{N}_0$ e $p \in PILHANAT$.

- Inverte$(vazia) = vazia$
- Inverte$(push(n, p)) =$ Sobrepoe(Inverte(p),$push(n, vazia)$)

A ideia

O facto de dado conjunto estar definido indutivamente fornece uma estratégia geral para provar propriedades gozadas por elementos desse conjunto: o princípio de indução.

O princípio de indução

Uma regra de uma definição indutiva de um conjunto S, introduzindo termos construidos com dado operador n-ário op, tem a forma geral seguinte.

> *Se para qualquer i tal que $1 \leq i \leq n$, com $n \geq 0$, se tem que $e_n \in S_i$, então $\mathrm{op}(e_1, \ldots, e_n) \in S$, sendo cada S_i ou o conjunto S um outro conjunto já previamente definido.*

Generalização

- Tal como se faz indução sobre os naturais, pode-se fazer indução sobre (os construtores de) qualquer conjunto definido indutivamente.
- A indução natural é um caso particular desta indução *estrutural*.

O princípio de indução estrutural

- Seja S um conjunto definido indutivamente e P um predicado sobre elementos de S.
- Se para cada construtor (k-ário) op da definição indutiva de S, se tem $P(\text{op}(e_1, \ldots, e_k))$, se $P(e_i)$, para cada $e_n \in S$ (com $1 \leq i \leq k$),
  então tem-se $P(e)$ para qualquer $e \in S$.

A sobreposição de dada pilha na vazia é essa pilha

$P \in PILHANAT \rightarrow BOOL$
$P(p) = Sobrepoe(p, vazia)) = p$

Prova por indução estrutural

- Caso Base: $p = vazia$.
  $Sobrepoe(vazia, vazia) = vazia$
- Passo: $p = push(n, r)$, com $n \in \mathbb{N}_0$ e $r \in PILHANAT$.

$$\begin{aligned} Sobrepoe(push(n, r), vazia)) &= push(n, Sobrepoe(r, vazia)) \\ &= push(n, r) \end{aligned}$$

Tamanho da sobreposição de pilhas

$P \in PILHANAT^2 \to BOOL$
$P(p, q) = Tam(Sobrepoe(p, q)) = Tam(p) + Tam(q)$

Prova por indução estrutural

- Caso Base: $p = vazia$.
  $Tam(Sobrepoe(vazia, q)) = Tam(q) = 0 + Tam(q) = Tam(vazia) + Tam(q)$
- Passo: $p = push(n, r)$, com $n \in \mathbb{N}_0$ e $r \in PILHANAT$.

$$\begin{aligned} Tam(Sobrepoe(push(n, r), q)) &= Tam(push(n, Sobrepoe(r, q)) \\ &= 1 + Tam(Sobrepoe(r, q)) \\ &= 1 + Tam(r) + Tam(q) \\ &= Tam(p) + Tam(q) \end{aligned}$$

A sobreposição de pilhas é associativa

$P \in PILHANAT^3 \rightarrow BOOL$
$P(p, q, r) =$
$\quad Sobrepoe(Sobrepoe(p, q), r) = Sobrepoe(p, Sobrepoe(q, r))$

Prova por indução estrutural

Caso Base: $p = vazia$.
$Sobrepoe(Sobrepoe(vazia, q), r) = Sobrepoe(q, r) =$
$Sobrepoe(vazia, Sobrepoe(q, r))$
Passo: $p = push(n, r)$, com $n \in \mathbb{N}_0$ e $r \in PILHANAT$.
$Sobrepoe(Sobrepoe(push(n, s), q), r) =$
$Sobrepoe(push(n, Sobrepoe(s, q)), r) =$
$push(n, Sobrepoe(Sobrepoe(s, q), r)) =$
$push(n, Sobrepoe(s, Sobrepoe(q, r))) =$
$Sobrepoe(push(n, s), Sobrepoe(q, r))$