DD1350 Logik för dataloger

Fö 10 – Strukturell induktion

1

Matematisk induktion

Tidigare: Matematisk induktion tillåter oss att bevisa påståenden om de naturliga talen $\mathbb{N}$, genom att utnyttja att elementen i $\mathbb{N}$ är **ordnade, uppräkningsbara** och har ett **minsta element**.

Induktion kan även utföras på andra ordnade mängder, inte bara $\mathbb{N}$. I denna föreläsning ska vi studera induktion över **rekursivt definierade mängder**.

Rekursiva definitioner

Exemplet från förra föreläsningen

$$\sum_{i=0}^{x} i = \frac{x(x+1)}{2}$$

kan också ges en rekursiv definition:

$f(0) = 0$ (basfall)

$f(i) = i + f(i-1) \quad \text{om } i > 0$ (rekursivt steg)

Rekursion och induktion

Det finns ett nära samband mellan rekursion och induktion.

$f(0) = 0$

$f(i) = i + f(i-1) \quad \text{om } i > 0$

Visa att $f(n) = n(n+1)/2$, för alla n.

<u>Basfall:</u> $f(0) = 0 \cdot (0-1)/2 = 0$

<u>Ind.hypotes:</u> Antag att $f(n) = n(n+1)/2$.

<u>Ind.steg:</u> $f(n+1) = (n+1) + f(n) = (n+1) + n(n+1)/2 = \ldots$
$= (n+1)(n+2)/2 = = (n+1)((n+1)+1)/2$

Rekursiva definitioner

Ibland är **självreferens** det lättaste sättet att definiera en mängd objekt.

”En heltalslista är antingen en tom lista, eller ett heltal följt av en heltalslista.”

”Ett binärt träd är antingen ett löv, eller en nod med vänster och höger binärt träd.”

Självreferens ska dock användas med försiktighet; kom ihåg Russells paradox (fö 6). *”Mängden av alla mängder som inte tillhör sig själva.”*

Listor

”En heltalslista är antingen en tom lista, eller ett heltal följt av en heltalslista.”

Med grammatikregler:

<IntegerList> ::= [ ] | cons(<Integer>, <IntegerList>)

Basfall **Rekursivt fall**

T.ex. **`cons(1,cons(2,[]))`**

Precis som Prolog kommer vi använda syntaktiskt socker, och skriva **`[1,2]`** i stället för **`cons(1,cons(2,[]))`**

Funktioner på listor

Längd av en heltalslista:

```
len([]) = 0
len(cons(A,L)) = 1+len(L)
```

T.ex.

Kom ihåg att `[1,2] = cons(1,cons(2, [] ))`

```
len([1,2]) =
  1+len([2]) =
  1+1+len([]) =
  1+1+0 = 2
```

Funktioner på listor

Lägg ihop heltalslistor:

```
app([],L) = L
app(cons(A,L1),L2) = cons(A,app(L1,L2))
```

T.ex.

```
app([1,2], [3,4]) =
  cons(1, app([2],[3,4])) =
  cons(1, cons( 2, app([],[3,4]))) =
  cons(1, cons(2, [3,4])) =
  [1,2,3,4]
```

Strukturell induktion

Visa att `len(app(A,B))=len(A)+len(B)`
för alla listor A och B.

Bevis genom induktion över första listan:

Basfall `[]`: `len(app([],B)  =  len(B)  =  0+len(B)`
`= len([])+len(B)`

Ind.hypotes: Antag `len(app(A,B))=len(A)+len(B)`

Ind.steg `cons(X,A)`: `len(app(cons(X,A),B)  =`
`len(cons(X,app(A,B)))  =  1+len(app(A,B))  =`
`1+len(A)+len(B)  =  len(cons(X,A))+len(B)`

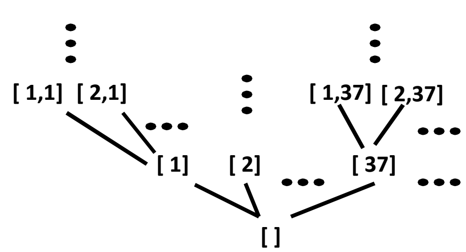
Enl. induktions-hypotesen

Vilken ordning gör vi induktion över?

Listorna är ordnade med **suffix**-relationen, t.ex.
[1] ≤ [2,1] och [4,5,6] ≤ [1,2,3,4,5,6].

dvs L1 ≤ L2 om

<u>antingen</u> L2 = cons(X,L1) för något X,
<u>eller</u> L2 = cons(X,M) och L1 ≤ M.



Partiell ordning med oändlig förgrening i varje nod.

Binära träd

"Ett binärt träd är antingen ett löv, eller en nod med vänster och höger binärt träd."

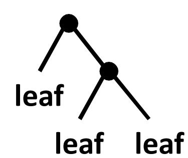
Med grammatikregler:

<Btree> ::= leaf | t(<Btree>,<Btree>)

Basfall **Rekursivt fall**

T.ex.

`t(leaf,t(leaf,leaf))`



Funktioner på träd

Höjden på ett träd:

```
height(leaf) = 0
height(t(A,B)) = 1 + max(height(A),height(B))
```

T.ex.

```
height(t(leaf,t(leaf,leaf))) =
   1 + max(height(leaf),height(t(leaf,leaf))) =
   1 + max(0, height(t(leaf,leaf))) =
   1 + max(0, 1 + max(height(leaf), height(leaf))) =
   1 + max(0, 1 + max(0,0)) =
   1 + max(0, 1) = 1+1 = 2
```

Strukturell induktion

Antalet noder:

`n(leaf) = 1`
`n(t(A,B)) = 1 + n(A)+n(B)`

Bevisa att $\texttt{n(T)} \le 2^{\texttt{height(T)+1}} - 1$ *för alla binära träd* **T.**

<u>Basfall:</u> $\texttt{n(leaf)} = 1 \le 2^1 - 1 = 2^{\texttt{height(leaf)+1}} - 1$

<u>Ind.hypotes:</u> Antag $\texttt{n(A)} \le 2^{\texttt{height(A)+1}} - 1$ och
$\texttt{n(B)} \le 2^{\texttt{height(B)+1}} - 1$

<u>Ind.steg:</u> $\texttt{n(t(A,B))} = 1 + \texttt{n(A)} + \texttt{n(B)} \le$ ← **Enl. induktions-hypotesen**

$\le 1 + 2^{\texttt{height(A)+1}} - 1 + 2^{\texttt{height(B)+1}} - 1 \le$

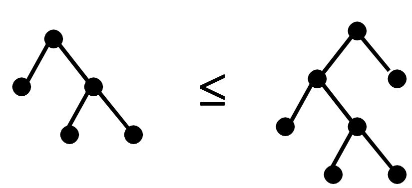
$\le 2\,(\texttt{max}(2^{\texttt{height(A)+1}}, 2^{\texttt{height(B)+1}})) - 1 =$

$= 2 \cdot 2^{\texttt{max(height(A),height(B))+1}} - 1 =$

$= 2 \cdot 2^{\texttt{height(t(A,B))}} - 1 = 2^{\texttt{height(t(A,B))+1}} - 1$

Vilken ordning gör vi induktion över?

Träden är ordnade med **delträds**relationen, t.ex.



dvs T1 ≤ T2 om

<u>antingen</u> T2 = t(T1,X) eller t(X,T1) för något X,
<u>eller</u> <u>antingen</u> T2=t(U,X) för något X och T1 ≤ U,
<u>eller</u> T2 = t(X,U) för något X och T1 ≤ U