Forelesning 14

Rekursjon og induksjon

Dag Normann - 27. februar 2008

Oppsummering

- Mandag repeterte vi en del om relasjoner, da spesielt om ekvivalensrelasjoner og partielle ordninger.
- Vi snakket videre om funksjoner.
- Det er noen grunnleggende begreper i tilknytning til kapitlet om funksjoner man må kjenne til for å kunne gå eksamensdagen i møte med ro i sinnet.
- Det er
  - Injektive funksjoner, også kalt 1-1-funksjoner eller enentydige funksjoner.
  - Surjektive funksjoner (onto).
  - Sammensetning av funksjoner.
  - Omvendte eller inverse funksjoner.

Oppsummering

Det er også viktig å holde orden på hva som menes med:

- *Definisjonsområdet* til en funksjon.
- *Verdiområdet* til en funksjon.
- *Bildemengden* til en funksjon.

I tillegg bør man kunne vite når

- man kan finne en invers til en funksjon
- man kan sette sammen to funksjoner.

Dette avslutter den abstrakte innføringen i funksjoner.

Før vi går over til neste kapittel skal vi imidlertid se litt på hva det vil si at en funksjon er beregnbar.

Beregnbare funksjoner

- IT dreier seg mye om hvordan man løser oppgaver ved hjelp av elektroniske hjelpemidler, fortrinnsvis datamaskiner.
- All IT-aktivitet på maskin-nivå styres av programmer, uansett om vi ser dem eller ikke.
- Hvis man skal kunne forstå informasjonsteknologiens begrensninger, må vi derfor forstå grensene for hva det er mulig å skrive programmer for.
- Alle programmer beskriver egentlig funksjoner, selv om noen argumenter (som maskintid, maskinarkitektur o.a.) ikke er synlig.

- Det er derfor av interesse å studere de funksjonene som lar seg uttrykke ved hjelp av programmer.

Beregnbare funksjoner

- Hvis vi begrenser oss til funksjoner fra $\mathbb{N}_0$ til $\mathbb{N}_0$ har vi gode matematiske karakteriseringer av de beregnbare funksjonene, det vil si de som kan programmeres i et eller annet programmeringsspråk. ($\mathbb{N}_0 = \mathbb{N} \cup \{0\}$)
- Det viser seg at alle programmerbare funksjoner fra $\mathbb{N}_0$ til $\mathbb{N}_0$ kan formuleres som en av våre pseudokoder, hvor vi bare bruker navn på tallene 0 og 1, addisjon og multiplikasjon og Booleske tester uttrykt ved hjelp av $=$ og $<$.

  Det er ikke uvanlig for logikere eller folk som arbeider med teoretisk databehandling å la de naturlige tallene starte med 0.

  Vi skal være snille og holde oss til måten boka gjør det på.

Beregnbare funksjoner

Som en forberedelse til kapittel 7 om induksjon og rekursjon, skal vi se på to pseudokoder hvor vi har pålagt oss å begrense oss til addisjon, multiplikasjon og Booleske tester med $=$ og $<$ (men dermed får lov til å bruke $\leq$).

- I det første eksemplet skal vi beregne $f(x, y) = max\{0, x - y\}$.
- I det andre eksemplet skal vi beregne $g(x, y) = x^y$.

Beregnbare funksjoner

Eksempel (Beregnbare funksjoner)

1. *Input* x $[x \in \mathbb{N}_0]$
2. *Input* y $[y \in \mathbb{N}_0]$
3. $z \leftarrow 0$
4. **While** $y < x$ **do**
   - 4.1 $y \leftarrow y + 1$
   - 4.2 $z \leftarrow z + 1$
5. *Output* z

- Vi har ikke snakket om induksjonsbevis ennå. Det vil være den naturlige metoden for å vise korrekthet av et slikt program.
- I dette tilfellet ser vi at hvis $x \leq y$ starter vi ikke løkka i det hele tatt, mens hvis $y < x$ "teller" vi y opp til x samtidig som vi øker verdien av z tilsvarende mye.

Beregnbare funksjoner

Eksempel (Beregnbare funksjoner)

1. *Input* x $[x \in \mathbb{N}_0]$
2. *Input* y $[y \in \mathbb{N}_0]$
3. $u \leftarrow 0$
4. $z \leftarrow 1$
5. **While** $u < y$ **do**
   - 5.1 $z \leftarrow z \cdot x$
   - 5.2 $u \leftarrow u + 1$
6. *Output* z

Dette resulterer i at vi multipliserer x med seg selv y ganger, altså at vi beregner x^y.

Beregnbare funksjoner

- I programmeringssammenheng er det ikke alltid så lett å vite når et gitt program med et gitt input faktisk gir oss et output i den mengden hvor vi vil ha det.
- I verste fall kan vi skrive programmer for funksjoner hvor det er umulig å bestemme hva definisjonsområdet er.
- Innenfor IT er det derfor naturlig også å studere partielle funksjoner fra en mengde X til en mengde Y.
- Dette vil være funksjoner hvor definisjonsområdet er en delmengde av X og hvor verdiområdet er Y.
- Tolkningen av et program som en funksjon fra et Cartesisk produkt av datatyper til en datatype vil vanligvis være som en partiell funksjon.

OVER TIL KAPITTEL 7

Innledning til rekursjon og induksjon

- Vi skal nå starte på avsnittet om rekursive konstruksjoner og bevis ved induksjon.
- Dette er det første stedet hvor årets MAT1030 vil omfatte mer stoff enn det læreboka omfatter.
- Det betyr at forelesningene er å betrakte som pensum, også der de går ut over rammene til læreboka.
- Alt stoff som er eksamensrelevant vil man finne i læreboka eller i forelesningsnotatene som legges ut på nettet.

Innledning til rekursjon og induksjon

- Læreboka behandler for det meste rekursjon og induksjon over de naturlige tallene $\mathbb{N}$.
- I en IT-sammenheng finnes det andre induktivt konstruerte mengder hvor tilsvarende metoder har mening.

- Vi skal etterhvert se på noen generelle og spesielle eksempler av interesse for IT.
- Vi skal imidlertid først se på rekursjon i en begrenset, men viktig, forstand.

Rekursjon

Eksempel

- Vi definerer en funksjon $f : \mathbb{N} \to \mathbb{N}$ ved
  1. $f(1) = 2$
  2. $f(n+1) = 2^{f(n)}$ for alle n.
- Vi har ikke definert f ved en formel, så er f veldefinert?

Rekursjon

Eksempel (Fortsatt)

- En test kan jo være om vi er i stand til å skrive et program for f!
- Vi kan oppfatte punktene 1. og 2. på forrige side som en spesifikasjon.
- Vi har tidligere sett hvordan vi kan finne en pseudokode for $g(z) = 2^z$
- Det betyr at vi kan bruke en instruksjon på formen

$$z \leftarrow 2^y$$

  med vissheten om at vi kan erstatte den ene linjen med en pseudokode.
- Da er det lett å lage en pseudokode for f:

Rekursjon

Eksempel (Fortsatt)

1. *Input* x $[x \in \mathbb{N}]$
2. $z \leftarrow 2$
3. $i \leftarrow 1$
4. **While** $i < x$ **do**
   4.1 $i \leftarrow i + 1$
   4.2 $z \leftarrow 2^z$
5. *Output* z

Vi kaller $f(x)$ verdien på 2^{er}-tårnet av høyde x.

Rekursjon

Eksempel

- Vårt neste eksempel er en funksjon som brukes mye i matematikk og i sannsynlighetsregning,

$$n \mapsto n!,$$

  eller fakultetsfunksjonen.
- Vi kan bruke omtrent samme formatet som i forrige eksempel:
  1. $1! = 1$
  2. $(n+1)! = n! \cdot (n+1)$ for alle $n \in \mathbb{N}$.
- Vi kan nærmest kopiere pseudokoden fra forrige eksempel, og får følgende algoritme for beregning av $n!$:

Rekursjon

Eksempel (Fortsatt)

1. *Input* x $[x \in \mathbb{N}]$
2. $z \leftarrow 1$
3. $i \leftarrow 1$
4. **While** $i < x$ **do**
   4.1 $i \leftarrow i + 1$
   4.2 $z \leftarrow z \cdot (i)$
5. *Output* z

Rekursjon

- Læreboka tar utgangspunkt i tallfølger, mens vi tar utgangspunkt i funksjoner.
- Det er i prinsippet ingen forskjell mellom en uendelig tallfølge og en funksjon definert på $\mathbb{N}$
- Tallfølgen

$$1, 2, 6, 24, 120, 720, \ldots$$

  er bare en annen måte å skrive fakultetsfunksjonen på.
- Hvorvidt man i konkrete tilfeller bruker tallfølger eller funksjoner, avhenger av hva som er pedagogisk mest forstandig for anledningen.

Rekursjon

- Kan vi gi en bedre begrunnelse for at de to funksjonene vi har sett på er veldefinerte enn at vi kan finne pseudokoder for dem?
- Svaret er selvfølgelig *JA*.
- Vi kan nå alle naturlige tall ved å
  1. Starte med 1

2. Legge til 1 så mange ganger som nødvendig.

- Hvis vi da definerer en funksjon f ved å bestemme
  1. hva $f(1)$ er
  2. hvordan $f(n+1)$ avhenger av $f(n)$ og n

  har vi bestemt $f(n)$ for alle n.
- Vi kan oppfatte en konkretisering av punktene 1 og 2 over som en spesifikasjon.
- Vi skal se på et eksempel i detalj:

Rekursjon

Eksempel

Vi definerer funksjonen $f(n, m)$ ved rekursjon på n ved

1. $f(1, m) = 2m - 1$
2. $f(n+1, m) = 2f(n, m) - 1$

- Med tilstrekkelig tålmodighet kan vi finne et uttrykk for $f(n, m)$ for hver enkelt n ved:

Rekursjon

Eksempel

1. $f(1, m) = 2m - 1$
2. $f(2, m) = 2f(1, m) - 1 = 2(2m - 1) - 1 = 4m - 3$
3. $f(3, m) = 2f(2, m) - 1 = 2(4m - 3) - 1 = 8m - 7$
4. $f(4, m) = 2f(3, m) - 1 = 2(8m - 7) - 1 = 16m - 15$

   ...

Vi ser at vi kan gjøre listen av utregninger så lang vi vil, så $f(n, m)$ er definert for alle n og m.

En annen sak er om vi kan vise den formelen som ser ut til å peke seg ut.

Da vil vi få bruk for induksjonsbevis.

Rekursjon

- Læreboka har brukt **For**-løkker der vi har brukt **While**-løkker.
- Forskjellen er kosmetisk.
- Det viktige er at vi bruker en løkke til å fange opp formatet
  1. $g(1) = a$
  2. $g(n+1) = f(g(n), n)$
- og at vi har en standard overgang fra en pseudokode for f til en pseudokode for g.
- Vi sier at g er definert fra a og f ved rekursjon.
- Vi beskriver den generelle **For**-løkka på neste side:

Rekursjon

1. *Input* n $[n \in \mathbb{N}]$
2. $x \leftarrow a$
3. **For** $m = 2$ **to** n **do**
   3.1 $x \leftarrow f(x, m)$
4. *Output* x

Merk

Vi sier at klassen av funksjoner programmerbare via en pseudokode er lukket under *definisjoner ved rekursjon*.

Rekursjon

Oppgave

Betrakt følgende pseudokode, hvor det inngår en rekursiv definisjon:

1. *Input* n $[n \in \mathbb{N}]$
2. $x \leftarrow 1$
3. $y \leftarrow 1$
4. $z \leftarrow 1$
5. **For** $m = 2$ **to** n **do**
   5.1 $y \leftarrow y + 1$
   5.2 **For** $k = 1$ **to** y **do**
      5.2.1 $z \leftarrow z + 1$
      5.2.2 $x \leftarrow x + z$
6. *Output* x

Rekursjon

Oppgave (Fortsatt)

- Følg beregningen og finn verdien på output for $n = 1$, $n = 2$, $n = 3$ og $n = 4$.
- Hvordan tror du denne følgen fortsetter?
- Vil beregningen stoppe uansett hvilket naturlig tall n vi starter med?

Rekursjon

- Til nå har vi bare sett på funksjoner fra $\mathbb{N}_0$ til $\mathbb{N}_0$ definert ved rekursjon.
- Filosofien bak hvorfor rekursive definisjoner gir mening gir oss også muligheten til å betrakte andre definisjonsområder:

Rekursjon

Eksempel

La $f(2) = 1$

Hvis $n \geq 2$ definerer vi $f(n+1)$ ved

- $f(n+1) = f(n)$ hvis n ikke er et primtall.
- $f(n+1) = f(n) + 1$ hvis n er et primtall.

Da er $f(n)$ definert for alle tall $n \geq 2$ og forteller oss hvor mange primtall det finnes $\leq n$.

Rekursjon

- Foreløpig gir det ikke mening å bruke rekursjon til å definere funksjoner med definisjonsområder som ikke er $\mathbb{N}$, $\mathbb{N}_0$ eller $\{n \in \mathbb{N} \,:\, n \geq k\}$ for en k.
- Det er imidlertid ingen grunn til at verdiområdet skal bestå av tall, noe vårt neste eksempel vil vise:

Rekursjon

Eksempel

- Vi har en klassisk definisjon av regningsart R_n nummer n:
- $R_1(x, y) = x + y$
- R_2 defineres rekursivt ved
  - $R_2(0, y) = 0$
  - $R_2(x+1, y) = R_1(R_2(x, y), y)$
- Hvis $n \geq 2$ og R_n er definert, definerer vi R_{n+1} rekursivt ved
  - $R_{n+1}(0, y) = 1$
  - $R_{n+1}(x+1, y) = R_n(R_{n+1}(x, y), y)$.

Vi ser at R_1 er addisjon, R_2 er multiplikasjon, R_3 er eksponensiering osv.

Induksjonsbevis

Eksempel

- La oss gå tilbake til den rekursive definisjonen
  1. $f(1, m) = 2m - 1$
  2. $f(n+1, m) = 2f(n, m) - 1$

  hvor det er naturlig å gjette på at

$$f(n, m) = 2^n \cdot m - (2^n - 1).$$

- Vi har sett at denne formelen stemmer for $n = 1$, $n = 2$, $n = 3$ og $n = 4$ da vi regnet ut

Induksjonsbevis

Eksempel (Fortsatt)

1. $f(1, m) = 2m - 1$
2. $f(2, m) = 2f(1, m) - 1 = 2(2m - 1) - 1 = 4m - 3$
3. $f(3, m) = 2f(2, m) - 1 = 2(4m - 3) - 1 = 8m - 7$
4. $f(4, m) = 2f(3, m) - 1 = 2(8m - 7) - 1 = 16m - 15$

- Hvis vi nu prøver å se om formelen stemmer for $n = 5$ på en slik måte at vi forhåpentligvis finner en forklaring, kan vi regne som følger:

Induksjonsbevis

Eksempel (Fortsatt)

$$f(5, m) = 2f(4, m) - 1 = 2(2^4 m - (2^4 - 1)) - 1 =$$

$$2 \cdot 2^4 m - 2(2^4 - 1) - 1 =$$

$$= 2^5 m - 2^5 + 2 - 1 = 2^5 m - (2^5 - 1).$$

Induksjonsbevis

Eksempel (Fortsatt)

- I denne utregningen har vi bare brukt at $5 = 4 + 1$
- Vi kunne erstattet 4 med en vilkårlig n og 5 med $n + 1$, og fått utregningen

$$f(n + 1, m) = 2f(n, m) - 1 = 2(2^n m - (2^n - 1)) - 1 =$$

$$= 2 \cdot 2^n m - 2(2^n - 1) - 1 =$$

$$= 2^{n+1} m - 2^{n+1} + 2 - 1 = 2^{n+1} m - (2^{n+1} - 1).$$

- Dermed har vi gitt det som kalles et induksjonsbevis for at formelen vår er riktig.

Induksjonsbevis

- Hvorfor kan vi betrakte argumentet over som et bevis for at formelen holder for alle n?

- Årsaken er at vi nå vet at vi ved direkte utregning kan bevise formelen hver gang noen gir oss en verdi for n, for eksempel $n = 8$.
- Vi vet nå at formelen holder for $n = 5$ og vi vet at siden den holder for $n = 5$ må den holde for $n = 6$.
- Utregningen vår gir imidlertid at formelen også må holde for $n = 7$ og deretter for $n = 8$.
- Siden vi vet at vi med utholdenhet kan fortsette å tenke slik så langt noen kunne ønske, vet vi at formelen vår må holde for alle n.

Induksjonsbevis

- Den metoden vi har brukt til å bevise en påstand for alle naturlige tall på kalles som sagt induksjonsbevis.
- I sin enkleste form kan induksjonsbevis formuleres som:

Definisjon

La $P(n)$ være et predikat med en variabel n for et element i $\mathbb{N}$.

Anta at vi kan bevise

1. $P(1)$
2. $\forall n(P(n) \rightarrow P(n+1))$

Da kan vi konkludere $\forall n P(n)$.

Denne måten å bevise $\forall n P(n)$ på kalles induksjon.