MAT1030 – Diskret matematikk

Forelesning 17: Generell rekursjon og induksjon

Dag Normann

Matematisk Institutt, Universitetet i Oslo

10. mars 2008

Opphenting

- Forrige uke så vi på rekurrenslikninger.
- En rekurrenslikning er en funksjonslikning på formen

$$at(n) + bt(n-1) + ct(n-2) = 0$$

hvor en løsning er en funksjon
$F : \mathbb{N} \to \mathbb{R}$ slik at

$$aF(n) + bF(n-1) + cF(n-2) = 0$$

for alle $n \geq 2$ (eller for alle n hvor n, $n-1$ og $n-2$ er i definisjonsområdet til F, når vi vil bruke maskineriet vårt i en mer generell situasjon).

- Den karakteristiske likningen er da

$$ax^2 + bx + c = 0$$

Opphenting

- Hvis r og s er to forskjellige løsninger av den karakteristiske likningen, er den generelle løsningen av rekurrenslikningen

$$F(n) = Ar^n + Bs^n$$

hvor A og B er vilkårlige reelle tall.

- Hvis den karakteristiske likningen bare har en løsning r, er den generelle løsningen av rekurrenslikningen

$$(A + Bn)r^n.$$

Opphenting

- Hvis vi i tillegg har krav om at $t(1) = a$ og $t(2) = b$, kan vi bestemme A og B i den generelle løsningen ved å sette inn for $n = 1$ og $n = 2$ i den generelle løsningen, og løse likningene mhp. A og B.
- Dette vil alltid fungere.
- Boka ser bare på tilfellet med initialbetingelser på $t(1)$ og $t(2)$.
- Hadde vi satt krav til $t(17)$ og $t(256)$, eller til t-verdien for to andre, forskjellige tall, kunne vi fortsatt bestemt A og B fra den informasjonen.
- Initialbetingelser for $n = 0$ og $n = 1$ kan gi den enkleste regningen.
- Vi skal nå fortsette innføringen av nytt stoff.

Rekursjon og programmering

- Vi startet innføringen av rekursjon med å gi eksempler på hvordan vi kunne finne pseudokoder som svarer til rekursive konstruksjoner.
- Vi kan minne om at hvis
  - $f(1) = a$
  - $f(n+1) = g(f(n), n)$

  er en rekursiv funksjon, og vi har en pseudokode for g, kan vi erstatte denne pseuokoden (som en del av en større kode) med

$$x \leftarrow g(i, j)$$

  i betydningen at variabelen x får verdien til f når inputvariablene får verdiene til i og j.
- Det er flere måter vi kan lage en pseudokode for g på, vi skal se på to av dem:

Rekursjon og programmering

Eksempel

1 *Input* n $[n \in \mathbb{N}]$
2 $x \leftarrow a$
3 *Output* x
4 **For** $i = 2$ **to** n **do**
  4.1 $x \leftarrow g(x, i)$
  4.2 *Output* x

Merk

Denne pseudokoden vil skrive ut $f(1), f(2), \ldots, f(n)$ i rekkefølge.

Hvis vi bare vil ha ut $f(n)$ blir koden enda enklere:

Rekursjon og programmering

Eksempel

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

Rekursjon og programmering

- Læreboka har et lite avsnitt om plassen til rekursjon i de enkelte programmeringsspråk.
- Dette er lesestoff, som ikke blir utdypet på forelesningene.
- Enkelte programmeringsspråk tillater til og med en sterkere form for rekursjon, selvkallende prosedyrer.
- Rekursjon er et spesialtilfelle av selvkallende prosedyrer, hvor vi definerer en funksjon $f(x)$ ved å bruke verdier $f(y)$ for enkelte y avhengige av x.
- En slik definisjon kan lett lede til løkkeberegninger eller uendelige beregninger, uten at det er lett å se hvorfor det er tilfelle.
- Vi skal ikke komme nærmere inn på det (nå), etter følgende eksempel.

Rekursjon og programmering

Eksempel

- Da vi ga eksempler på pseudokoder, ga vi et eksempel på en algoritme som ingen ennå vet om vil terminere for alle verdier på input, og vi ga algoritmen i form av en pseudokode.
- Våre pseudokoder fanger ikke opp muligheten for selvkallende prosedyrer, men hadde vi hatt den muligheten, kunne vi betraktet følgende som en meningsfylt algoritme:

Rekursjon og programmering

Eksempel (Fortsatt)

- La $f(1) = 1$.
- La $f(n) = f(\frac{n}{2}) + 1$ hvis n er et partall.
- La $f(n) = f(3n+1) + 1$ hvis $n > 1$ er et oddetall.
- Vi kan da eksempelvis regne ut

$$f(20) = f(10) + 1 = f(5) + 2 = f(16) + 3$$

$$= f(8) + 4 = f(4) + 5 = f(2) + 6 = f(1) + 7 = 8$$

- f er veldefinert som en *partiell* funksjon som vi ikke vet om er *total*.

Generell induksjon og rekursjon

- Vi har argumentert for at konstruksjoner ved rekursjon virker og at induksjon er en gyldig bevisteknikk.
- Vi har begrunnet dette med at vi kan nå alle naturlige tall ved å starte med 1 og så legge til 1 så mange ganger vi trenger.
- En måte å forklare hvorfor induksjonsbevis er matematisk holdbare argumenter på er følgende:
- Vi har at $\mathbb{N}$ er den minste mengden som oppfyller
  - $1 \in \mathbb{N}$
  - Hvis $x \in \mathbb{N}$ vil $x + 1 \in \mathbb{N}$
- Hvis vi da viser en egenskap P ved induksjon, viser vi at
  - $1 \in \{y : P(y)\}$
  - Hvis $x \in \{y : P(y)\}$ vil $x + 1 \in \{y : P(y)\}$.
- Siden $\mathbb{N}$ var den minste mengden med denne egenskapen, må $\mathbb{N} \subseteq \{y : P(y)\}$,

  eller, som vi sier, $P(x)$ holder for alle $x \in \mathbb{N}$.

Generell induksjon og rekursjon

- I logikk og informatikk (og også innen andre deler av matematikken og i andre fag) ser man definisjoner som likner på vår beskrivelse av $\mathbb{N}$; man beskriver en strukturert mengde ved å si hva som kommer inn som start og hvordan man finner mer komplekse elementer av mengden.
- Innen informatikk og logikk beskriver vi gjerne formelle språk på den måten.
- Det er et uttrykt ønske fra informatikerne om at de studentene som skal studere videre hos dem, skal ha en bedre forståelse av induksjon og rekursjon enn hva som har vært tilfelle tidligere år, og at studentene skal ha kjennskap til rekursjon over andre strukturer enn $\mathbb{N}$.

Generell induksjon og rekursjon

- I de følgende eksemplene skal vi anta at vi har et alfabet som består av alle de symbolene vi kan finne på tastaturet til en standard datamaskin (norsk standard om presisjonen er nødvendig), hvor tomrom er å betrakte som et eget symbol. Vi bruker bokstaver i *kursiv* som variable over bokstavene på tastaturet, og vi kan bruke andre bokstaver i kursiv som variable for ord, hvor:

Generell induksjon og rekursjon

- Et ord er en ordnet sekvens av bokstaver, hvor bokstavene er skrevet uten ekstra tegn i mellom.
- Det er vanlig å la e betegne det tomme ordet.
- Vi føyer en bokstav til et ord ved ganske enkelt å skrive det inntil ordet på høyre side.
- Dette kan vi bruke til å gi en induktiv beskrivelse av mengden av ord.

Generell induksjon og rekursjon

Definisjon

Mengden av ord er den minste mengden som oppfyller

- Det tomme ordet e er et ord.
- Hvis w er et ord og b er en bokstav, er wb et ord.

Generell induksjon og rekursjon

Merk

- Dette eksemplet virker en smule kunstig, fordi vi ikke oppfatter ord slik, selv om de fleste av oss starter med tom linje, og så skriver en og en bokstav fra venstre mot høyre.
- Det spiller imidlertid en rolle for hvordan ord representeres som data om de sees på som ordnede sekvenser med en gitt lengde eller som bygget opp ved at nye bokstaver legges til.
- Ord, slik vi har definert dem, er et spesialtilfelle av lister.
- En liste oppfattes som oftes som at enten er den den tomme listen e, eller så er den et ordnet par av siste element og resten av listen.
- Mange tenker seg at ytterste element (*hodet*) i en liste står til venstre for resten (*halen*) av listen.
- Programmeringspråket *LISP* er basert på listerekursjon.

Generell induksjon og rekursjon

Eksempel

Som et eksempel på en rekursivt definisjon av en funksjon på mengden av ord kan vi betrakte PL (Push Left) som virker på et ord w og en bokstav a ved:

- $PL(e, a) = a$
- $PL(wb, a) = PL(w, a)b$

Det som her skjer er at PL tar for seg et ord w og en bokstav a, og skriver bokstaven *foran* ordet, slik at vi får aw.

Egentlig burde vi vist denne egenskapen ved PL ved induksjon, men vi avstår i denne omgangen.

Generell induksjon og rekursjon

Eksempel (Fortsatt)

Følgende regne-eksempel viser hvordan PL virker:

$PL(\mathrm{aba}, \mathrm{c}) =$

$PL(\mathrm{ab}, \mathrm{c})\mathrm{a} =$

$PL(\mathrm{a}, \mathrm{c})\mathrm{ba} =$

$PL(e, \mathrm{c})\mathrm{aba} =$

caba

Generell induksjon og rekursjon

Eksempel

Med utgangspunkt i eksemplet PL fra forrige side skal vi definere en speilingsfunksjon R ved rekursjon. R vil ta et ord som input og output vil være ordet skrevet baklengs:

- Vi definerer R ved:
  - $R(e) = e$
  - $R(wb) = PL(R(w), b)$

Igjen overlater vi bekreftelsen av at funksjonen virker i henhold til spesifikasjonen til den enkelte.

Vi skal se på et eksempel, og i tillegg gi en oppgave, som skal hjelpe til med dette.

Generell induksjon og rekursjon

Eksempel

Vi skal vise ved et regneeksempel i full detalj at $R(\mathrm{abc}) = \mathrm{cba}$ slik R og PL er definerte.

Vi vil bare bruke symbolet e når vi ellers måte skrevet *ingenting*, så $e\mathrm{ab}$ er det samme som ab.

$$
\begin{aligned}
&R(\mathrm{abc}) = PL(R(\mathrm{ab}), c)\\
&= PL(PL(R(\mathrm{a}), \mathrm{b}), \mathrm{c})\\
&= PL(PL(PL(R(e), \mathrm{a}), \mathrm{b}), \mathrm{c})\\
&= PL(PL(PL(e, \mathrm{a}), \mathrm{b}), \mathrm{c}) = PL(PL(\mathrm{a}, \mathrm{b}), \mathrm{c})\\
&= PL(PL(e, \mathrm{b})\mathrm{a}, \mathrm{c}) = PL(\mathrm{ba}, \mathrm{c}))\\
&= PL(\mathrm{b}, \mathrm{c})\mathrm{a} = PL(e, \mathrm{c})\mathrm{ba} = \mathrm{cba}.
\end{aligned}
$$

Generell induksjon og rekursjon

Oppgave

a) Ved å bruke den rekursive definisjonen av PL, vis hvordan vi skritt for skritt kan finne verdiene av
   - $PL(e, \mathrm{d})$
   - $PL(\mathrm{a}, \mathrm{d})$
   - $PL(\mathrm{ab}, \mathrm{d})$
   - $PL(\mathrm{aba}, \mathrm{d})$

   Husk at *variablene* a og b kan stå for hvilken som helst av *bokstavene* a, b og d.

b) Vis, ved å bruke definisjonen av R og egenskapen til PL at $R(\mathrm{abac}) = \mathrm{caba}$.

Generell induksjon og rekursjon

Eksempel

Den siste funksjonen vi skal se på i forbindelse med den induktive definisjonen av mengden av ord er sammensetningsfunksjonen $w * v = wv$.

Denne kan også defineres ved rekursjon som følger:
- $w * e = w$
- $w * (vb) = (w * v)b$

Vi kunne gjort det vanskeligere, nemlig brukt rekursjon på første argument:
- $S(e, v) = v$
- $S(wb, v) = S(w, PL(v, b))$

Hvis vi nå vil vise at $S(w, v) = w * v$ for alle ord w og v, kan vi ha god bruk for induksjon.

Generell induksjon og rekursjon

- Automatateori er den matematiske teorien for studiet av formelle regnemaskiner som tar for seg et inputord, og enten prosesserer et outputord eller svarer på et spørsmål om inputordet.
- Flere av disse formelle maskinene leser inputordet fra venstre mot høyre, og utfører en beregning underveis.
- Eksempler på dette er de såkalte endelige tilstandsmaskinene og pushdownautomater eller stackautomater.
- Hvordan en slik automat virker på et ord defineres ved rekursjon på oppbyggingen av ordet.
- Vi skal ikke innføre automatateori, men vi skal i eksempler og oppgaver se på et par tilfeller hvor rekursjon på ord kan brukes til å erstatte slike formelle regnemaskiner.
- De kraftigste formelle maskinene fanges ikke opp av ord-rekursjon.

Generell induksjon og rekursjon

Eksempel

- La w være et ord hvor vi vet at bare bokstavene a og b forekommer.
- Vi vil finne en algoritme som avgjør om det er like mange bokstaver av hvert slag, eller om det er et overskudd av den ene.
- Som en hjelpefunksjon definerer vi en funksjon $f(w)$ slik at $f(w)$ er differansen mellom antall a'er og antall b'er i w.
  - $f(e) = 0$
  - $f(w\mathrm{a}) = f(w) + 1$
  - $f(w\mathrm{b}) = f(w) - 1$

Generell induksjon og rekursjon

Eksempel (Fortsatt)

- Vi kan da regne ut

$$f(\mathrm{aababba}) = f(\mathrm{aababb}) + 1 = f(\mathrm{aabab}) - 1 + 1 = f(\mathrm{aabab})$$

$$= f(\mathrm{aaba}) - 1 = f(\mathrm{aab}) = f(\mathrm{aa}) - 1 = f(\mathrm{a}) = \mathrm{f(e)} + 1 = 1.$$

Generell induksjon og rekursjon

Eksempel (Fortsatt)

- Siden ord er ordnede sekvenser av symboler, kan vi finne pseudokoder som erstatter ord-rekursjon.
- Vi skal gi en pseudokode for beregning av f fra dette eksemplet.
- Her er det viktig at *input* er et ord hvor bokstavene a og b forekommer, og at *output* er et helt tall.
- Vanligvis må man deklarere typene til variablene som en del av programmet, men det formaliserer vi ikke her.

Generell induksjon og rekursjon

Eksempel (Fortsatt)

1 *Input* n $[n \in \mathbb{N}_0]$
2 *Input* w $[w = v_1 \cdots v_n$ er et ord av lengde $n]$
3 $x \leftarrow 0$
4 **For** $i = 1$ **to** n **do**
  4.1 **If** $v_i = \mathrm{a}$ **then**
    4.1.1 $x \leftarrow x + 1$
    else
    4.1.2 $x \leftarrow x - 1$
5 *Output* x

Generell induksjon og rekursjon

Merk

- Hvis vi arbeider med algoritmer som skal virke på ord i et alfabet eller lister av data, vil det være aktuelt å innføre egne kontrollstrukturer for listerekursjon.
- Det finnes programmeringsspråk av teoretisk interesse, og også av praktisk betydning, hvor det er enkelt å uttrykke listerekursjon og andre former for generell rekursjon.

Generell induksjon og rekursjon

Fra nå av skal vi anta at vi har definert mengen av ord, og at vi kan foreta oss de operasjoner på ord og bokstaver som vi finner nødvendige.

Hvorvidt det er naturlig å bruke *ordrekursjon* eller ikke, avhenger av hvordan vi representerer ord i en datamaskin.

Definisjon

Et formelt språk er en mengde av ord.

Generell induksjon og rekursjon

Eksempel

- Mengden av syntaktisk korrekte program i et gitt programmeringsspråk er et formelt språk.
- Mengden av utsagnslogiske formler i utsagnsvariablene p, q og r er et formelt språk, hvis vi regner $\neg$, $\wedge$, $\vee$, $\rightarrow$ og $\leftrightarrow$ med blant bokstavene.
- Mengden av ord fra *delalfabetet* $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ som uttrykker kubikktall er et formelt språk.
- Mengden av ord som betyr noe både på tysk og spansk er et formelt språk.

Generell induksjon og rekursjon

Merk

- Mange av de språkene som oppstår i en informatikk-sammenheng er definert induktivt.
- Den begrensningen vi ga til at symbolene skal finnes på tastaturet, er ikke vanlig.
- Det vanlige er at man for ethvert formelt språk også presiserer hvilket alfabet som skal brukes.
- Vår opprinnelige definisjon av formelle språk er brukbar for studiet av programmeringsspråk, men det er viktige eksempler som går ut over begrensningen til tastaturet.

Generell induksjon og rekursjon

Eksempel

- Vi skal nå se hvordan vi kan gi en helt presis definisjon av de utsagnslogiske formlene.
- For at ikke eksemplet skal bli for omfattende, dropper vi bindeordene $\rightarrow$ og $\leftrightarrow$ i denne definisjonen.
- Vi lar p, q og r være utvalgte bokstaver.
- Vi skal gi en induktiv definisjon av mengden av utsagnslogiske formler hvor vi bruker p, q og r som utsagnsvariable, bindeordene $\neg$ $\wedge$ og $\vee$, og hvor vi har formalisert bruken av parenteser.
- Hva vi mener med *induktiv* definisjon vil vi la komme frem gjennom eksemplene.

Generell induksjon og rekursjon

Eksempel (Fortsatt)

1. p, q og r er formler.
2. Hvis A er en formel, er $\neg A$ en formel.
3. Hvis A og B er formler, er $(A \wedge B)$ og $(A \vee B)$ formler.
4. Mengden av formler er den minste mengden som oppfyller 1. - 3. over.

- Når man blir vant til å arbeide med induktive definisjoner, vil vanligvis punktet tilsvarende 4. være underforstått.
- Definisjonen av *ord* hadde langt på vei det samme formatet.

Rekursjon og programmering

Eksempel (Fortsatt)

- Vi har allerede gitt en viktig definisjon ved rekursjon på oppbyggingen av en formel, uten at vi har sagt direkte at det var det vi gjorde.
- En sannhetsverdifunksjon er en funksjon hvor definisjonsområdet er mengden av fordelinger av sannhetsverdier til variablene p, q og r, og verdiområdet er $\{\mathrm{T}, \mathrm{F}\}$.
- Ved hjelp av sannhetsverditabeller har vi definert funksjonene $F_\neg$, $F_\wedge$ og $F_\vee$, hvor eksempelvis $F_\neg(\mathrm{F}) = \mathrm{T}$, $F_\wedge(\mathrm{T}, \mathrm{F}) = \mathrm{F}$ og $F_\vee(\mathrm{T}, \mathrm{F}\} = \mathrm{T}$.
- Hvis (s_p, s_q, s_r) er en ordnet sekvens av tre sannhetsverdier, definerer vi

  $$F_A(s_p, s_q, s_r)$$

  ved rekursjon på oppbyggingen av A som følger:

Generell induksjon og rekursjon

Eksempel (Fortsatt)

- $F_p(s_p, s_q, s_r) = s_p$
- $F_q(s_p, s_q, s_r) = s_q$
- $F_r(s_p, s_q, s_r) = s_r$
- $F_{\neg A}(s_p, s_q, s_r) = F_\neg(F_A(s_p, s_q, s_r))$
- $F_{(A \vee B)}(s_p, s_q, s_r) = F_\vee(F_A(s_p, s_q, s_r), F_B(s_p, s_q, s_r))$
- $F_{(A \wedge B)}(s_p, s_q, s_r) = F_\wedge(F_A(s_p, s_q, s_r), F_B(s_p, s_q, s_r))$
- For å kunne forstå denne definisjonen trenger vi mye av det vi har lært, blant annet:
  - Generelle funksjoner
  - Sammensetning av funksjoner
  - Rekursive definisjoner i en generell situasjon.

Generell induksjon og rekursjon

Eksempel (Fortsatt)

- Vi bestemmer hvordan sannhetsverdifunksjonen skal se ut direkte for de enkleste formlene, nemlig utsagnsvariablene.
- For sammensatte fomler vil sannhetsverdifunksjonen avhenge av hvilke sannhetsverdifunksjoner vi har for delformler.
- Det er akkurat denne delen av rekursjonen vi følger når vi skriver ut en sannhetsverditabell.
- Hvis vi skal bruke utsagnslogikk i en programmeringssammenheng, vil det være programmer basert på en slik rekursiv definisjon som ligger til grunn når maskinen beregner verdien av en Boolsk test.