MAT1030 – Forelesning 19

Generell rekursjon og induksjon

Dag Normann - 6. april 2010

(Sist oppdatert: 2010-04-06 14:45)

Forelesning 19

Repetisjon

- Vi er godt i gang med kapitlet om rekursjon og induksjon.
- Vi har sett på hvordan vi kan definere funksjoner eller tallfølger ved rekursjon over $\mathbb{N}$ og på hva det vil si å bevise en påstand ved induksjon.
- Vi så hvordan vi, på en uniform måte, kan omforme en definisjon ved rekursjon til en pseudokode.
- Deretter brukte vi cirka en uke på å se på rekurrenslikninger og på hvordan man finner den generelle løsningen og spesielle løsninger gitt initialbetingelser.
- I siste time før Påske så vi på et eksempel på en induktiv definisjon, definisjonen av ord over et alfabet som lister.
- Vi definerte mengden av ord som et eksempel på en induktivt definert mengde ved
  - Det tomme ordet e er et ord.
  - Hvis w er et ord og a er en bokstav, vil wa være et ord.
- Når vi lagrer et ord som et dataobjekt, vil et ord beskrevet på denne måten ofte representeres anderledes enn hvis vi representerer et ord som en ordnet sekvens.
- Vi ga eksempler på tre funksjoner på mengden av ord, funksjoner som vi definerte ved rekursjon over oppbygningen av et ord:
  - $\mathrm{PL}(w, a) = aw$
  - $\mathrm{R}(w)$ er speilvendingen av w
  - $v * w = vw$.

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.

Eksempel.

- Mengden av syntaktisk korrekte program i et gitt programmeringsspråk er et formelt språk.
- Mengden av utsagnslogiske formler i utsagnsvariablene $\mathtt{p}$, $\mathtt{q}$ og $\mathtt{r}$ 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.

Merk.

- Mange av de språkene som oppstår i en informatikk-sammenheng er definert induktivt.
- Den begrensningen vi ga til at symbolene skal fins 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.

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 $\mathtt{p}$, $\mathtt{q}$ og $\mathtt{r}$ være utvalgte bokstaver.
- Vi skal gi en induktiv definisjon av mengden av utsagnslogiske formler hvor vi bruker $\mathtt{p}$, $\mathtt{q}$ og $\mathtt{r}$ 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.

Eksempel (Fortsatt).

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

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 $\{\mathbf{T}, \mathbf{F}\}$.
- Ved hjelp av sannhetsverditabeller har vi definert funksjonene $F_\neg$, $F_\wedge$ og $F_\vee$, hvor eksempelvis $F_\neg(\mathbf{F}) = \mathbf{T}$, $F_\wedge(\mathbf{T}, \mathbf{F}) = \mathbf{F}$ og $F_\vee(\mathbf{T}, \mathbf{F}\} = \mathbf{T}$.
- 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:

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.

Eksempel (Fortsatt).

- Vi bestemmer hvordan sannhetsverdifunksjonen skal se ut direkte for de enkleste formlene, nemlig utsagnsvariablene.
- For sammensatte formler 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.

- I vårt neste eksempel skal vi se på en definisjon hvor vi bruker simultan rekursjon.
- Simultan rekursjon innebærer at vi definerer to funksjoner f og g samtidig ved rekursjon.
- For å illustrere hva vi mener med "samtidig" skal vi se på et enkelt eksempel:

Eksempel.

- La $f(1) = 1$
- La $g(1) = 2$
- La $f(n+1) = f(n) \cdot g(n)$ for alle $n \in \mathbb{N}$.
- La $g(n+1) = f(n) + g(n)$ for alle $n \in \mathbb{N}$.
- Vi ser at da kan vi regne ut $f(2) = 1 \cdot 2$ og $g(2) = 1 + 2 = 3$.
- Da er $f(3) = 2 \cdot 3 = 6$ mens $g(3) = 2 + 3 = 5$.
- Slik kan vi fortsette å regne ut resultatene parvis, men vi kommer ingen vei hvis vi bare prøver å regne ut verdiene til f eller bare verdiene til g, vi må regne ut begge funksjonene *simultant*, det vil si *samtidig*.
- Begrunnelsen for at simultan rekursjon er en lovlig definisjonsform er den samme som for vanlig rekursjon.

Eksempel (Fortsatt).

- Hvis vi lar $h(n) = (f(n), g(n))$, det vil si, det ordnede paret av $f(n)$ og $g(n)$, er h en funksjon definert på $\mathbb{N}$ og med verdiområde $\mathbb{N} \times \mathbb{N}$.
- Lar vi $t : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ være definert ved

$$t(a, b) = (a \cdot b, a + b)$$

  ser vi at vi kan erstatte definisjonen av f og g med følgende rekursive definisjon av h:

  - $h(1) = (1, 2)$
  - $h(n+1) = t(h(n))$
- Det er ikke noe spesielt med dette eksemplet, vi kunne gjort en tilsvarende omskrivning hver gang vi definerer funksjoner ved simultan rekursjon.

Eksempel.

Det er ofte mye lettere å studere egenskapene til en utsagnslogisk formel, eksempelvis å undersøke om den er en tautologi, en kontradiksjon eller noe annet, om negasjonstegnet bare står i posisjon rett foran en utsagnsvariabel.

En slik formel sies å være på svak normalform.

Intuitivt kan vi finne en svak normalform ved å bruke deMorgans lover til å skyve negasjonstegnet innover.

Skal vi formulere dette presist, må vi bruke formatet til rekursive definisjoner.

Vi må definere den svake normalformen $SF(A)$ til A og den svake normalformen $SF_\neg(A)$ til $\neg A$ simultant.

Eksempel (Fortsatt).

- La $SF(A) = A$ og $SF_\neg(A) = \neg A$ hvis A er en utsagnsvariabel p, q eller r.
- $SF(\neg A) = SF_\neg(A)$
- $SF_\neg(\neg A) = SF(A)$
- $SF(A \vee B) = SF(A) \vee SF(B)$, $SF(A \wedge B) = SF(A) \wedge SF(B)$.
- $SF_\neg(A \vee B) = SF_\neg(A) \wedge SF_\neg(B)$, $SF_\neg(A \wedge B) = SF_\neg(A) \vee SF_\neg(B)$

- Heller ikke dette eksemplet er valgt bare for å gjøre MAT1030 vanskelig.
- Når man skal skrive programmer for å undersøke om en utsagnslogisk formel er oppfyllbar eller en kontradiksjon, er det til stor hjelp om man kan anta at formelen er på svak normalform.

Eksempel.

Som et eksempel skal vi se på hvordan vi finner den svake normalformen til $(\neg p \wedge q) \vee \neg(p \wedge \neg r)$:

$SF((\neg p \wedge q) \vee \neg(p \wedge \neg r)) =$

$SF(\neg p \wedge q) \vee SF(\neg(p \wedge \neg r)) =$

$(SF(\neg p) \wedge SF(q)) \vee SF_\neg(p \wedge \neg r) =$

$(SF_\neg(p) \wedge q) \vee (SF_\neg(p) \vee SF_\neg(\neg r)) =$

$(\neg p \wedge q) \vee (\neg p \vee SF(r)) =$

$(\neg p \wedge q) \vee (\neg p \vee r)$.

Eksempel.

- Riktig bruk av parenteser er viktig for at programmer skal være syntaktisk korrekte, eller for at matematiske uttrykk skal gi mening.
- I dette eksemplet skal vi se på mengden av korrekte parentesuttrykk, hvor vi bare bruker "("og ")" .
- Vi skal se på hvordan vi kan bygge opp mengden av korrekte parentesuttrykk ved induksjon.
- Deretter skal vi vise at den samme mengden kan beskrives på en annen måte, en måte vi kan bruke for maskinell kontroll av om parenteser er satt riktig eller ikke.
- Vi trenger induksjonsbevis både over oppbyggingen av korrekte parentesuttrykk og over de ikke-negative hele tallene for å vise dette.

Eksempel (Fortsatt).

- Til sist skal vi vise hvordan den alternative skrivemåten kan brukes til å finne en pseudokode som avgjør om et uttrykk med parenteser er korrekt eller ikke.

  De korrekte parentesuttrykkene er den minste mengden av ord som oppfyller

1. Det tomme ordet e er et korrekt parentesuttrykk.
2. Hvis v er et korrekt parentesuttrykk, er $w = (v)$ et korrekt parentesuttrykk.
3. Hvis u og v er korrekte parentesuttrykk, er *sammensetningen* $w = uv$ et korrekt parentesuttrykk.

   Vi skal studere dette viktige formelle språket litt nærmere.

Eksempel (Fortsatt).

Påstand 1

Hvis w er et korrekt parentesuttrykk, vil antall høyre og venstreparenteser i w være det samme.

Bevis

Vi bruker induksjon på oppbyggingen av et parentesuttrykk.

I dette tilfellet er påstanden egentlig opplagt, men vi beviser den ved induksjon likevel, som et eksempel.

Hvis $w = e$ er antall høyre og venstreparenteser begge lik 0, og derfor like.

La $w = (v)$ og anta at påstanden holder for v.

Eksempel (Fortsatt).

Da er både antall venstre- og høyreparenteser i w én større enn tilsvarende tall for v, som er like.

Derfor er de like for w også.

La så $w = uv$ og anta at påstanden holder for både u og for v.

Antall venstreparenteser i w er summen av antall venstreparenteser i u og antall venstreparenteser i v.

Ved induksjonsantagelsen er dette det samme som antall høyreparenteser i u og i v.

Da må summen også være den samme.

Eksempel (Fortsatt).

- Vi definerte mengden av de korrekte parentesuttrykkene som den minste mengden X slik at
  - $e \in X$
  - $v \in X \Rightarrow (v) \in X$
  - $u \in X \wedge v \in X \Rightarrow uv \in X$
- Vi har bevist at mengden Y av ord som har like mange venstre- og høyreparenteser, og ingen andre symboler, oppfyller
  - $e \in Y$
  - $v \in Y \Rightarrow (v) \in Y$
  - $u \in Y \wedge v \in Y \Rightarrow uv \in Y$
- Det betyr at $X \subseteq Y$, og det er en reformulering av Påstand 1.

Eksempel (Fortsatt).

Påstand 2

Hvis w er et korrekt parentesuttrykk, og vi leser w fra venstre mot høyre, finner vi ikke noe sted hvor det har vært flere høyre- enn venstreparenteser.

Bevis

Igjen bruker vi induksjon på oppbyggingen av w som et korrekt parentesuttrykk.

Hvis $w = e$ er dette opplagt riktig.

Anta at påstanden holder for v, og la $w = (v)$.

Når vi leser w fra venstre mot høyre har vi alltid én venstreparentes mer når vi leser w enn når vi er på tilsvarende sted i v.

Eksempel (Fortsatt).

Det betyr at vi har et positivt overskudd av (hele tiden mens vi leser v-delen av $(v$, og vi får ikke noe overskudd av) til slutt heller.

Anta at påstanden holder for u og for v, og at $w = uv$.

Når vi leser w fra venstre mot høyre kan vi først ikke opparbeide noe overskudd av) mens vi leser u-delen, og deretter heller ikke mens vi leser v-delen.

Dermed holder påstanden for w også.

Eksempel (Fortsatt).

Påstand 3

La w være et ord hvor vi bare har brukt symbolene "(" og ")".

Hvis w oppfyller konklusjonene i Påstand 1 og Påstand 2, vil w være et korrekt parentesuttrykk.

Bevis

Her lønner det seg å bruke induksjon på lengden av ordet w.

I motsetning til vanlig induksjon, starter beviset her med lengde 0, som er lengden til det tomme ordet.

Hvis lengden av w er 0, det vil si hvis $w = e$, er w et korrekt parentesuttrykk pr. definisjon.

La $w \neq e$ og anta at påstanden holder for alle kortere ord v (Kommentar kommer senere).

Eksempel (Fortsatt).

Vi må dele beviset opp i to tilfeller:

Tilfelle 1: Når vi leser w fra venstre mot høyre finner vi ikke noe sted før til slutt at w har like mange høyre- som venstreparenteser.

Siden w oppfyller påstandene, må w være på formen (v), og siden vi er i tilfelle 1 vil også v oppfylle Påstand 1 og Påstand 2.

Ved induksjonsantagelsen er v korrekt, og da er $w = (v)$ korrekt.

Eksempel (Fortsatt).

Tilfelle 2: Vi kan dele w opp i to ekte delord, $w = uv$, slik at u har like mange venstre- som høyreparenteser.

Da må det samme gjelde for v, siden det gjelder for w.

Videre får vi ikke noe overskudd av høyreparenteser mens vi leser u, siden det er det samme som å lese w. Siden vi ikke starter med noe overskudd av venstreparenteser etter å ha lest u, vil situasjonen være lik om vi leser v direkte eller etter å ha lest u.

Det betyr at Påstand 1 og Påstand 2 holder for både u og v.

Eksempel (Fortsatt).

Ved induksjonsantagelsen er da u og v korrekte.

Da er $w = uv$ også korrekt.

Siden dette argumentet dekker alle mulighetene, er Påstand 3 vist ved induksjon over $\mathbb{N}_0$

Eksempel (Fortsatt).

- La oss oppsummere dette eksemplet.
- For å vise Påstand 1 brukte vi induksjon over oppbyggingen av et uttrykk som et korrekt parentesuttrykk.
- Vi brukte induksjon over den samme induktivt definerte mengden for å bevise Påstand 2.
- For å bevise Påstand 3 brukte vi imidlertid induksjon over $\mathbb{N}_0$.
- Det er mulig å skrive om bevisene for Påstand 1 og for Påstand 2 slik at de blir beviser ved induksjon over $\mathbb{N}_0$, eksempelvis bruke induksjon på antall parenteser i uttrykket.
- Dette er en omskrivning som i innlæringsfasen kan gjøre det lettere å forstå generell induksjon, men som på sikt gjør det mer tungvint å formulere bevisene.

Eksempel (Fortsatt).

- Noen vil stusse over at vi ikke brukte den vanlige formuleringen med å vise $P(1)$ og at $P(n) \Rightarrow P(n+1)$ for alle n.
- Det vi brukte her var et annet induksjonsprinsipp:

  Anta at vi kan vise for alle n at

  $$\forall m < n P(m) \rightarrow P(n)$$

- Da kan det ikke fins noe minste tall n slik at $P(n)$ er usann.
- Dermed vet vi at $P(n)$ er sann for alle n.
- Det er fullt ut akseptabelt å bruke denne alternative formen i bevis.
- Vi skal komme tilbake med et eksempel til på denne formen for induksjon.

Eksempel (Fortsatt).

- Hvordan kan vi så finne en pseudokode som avgjør om et parentesuttrykk er korrekt eller ikke?
- Vi skriver en kode for funksjonen som teller overskudd av "(" i forhold til ")".
- Underveis kontrolleres det at vi ikke får for mange ")".
- Til sist kontrolleres det at vi hadde like mange av hver.
- n hentes fra $\mathbb{N}_0$, alle v_i'ene er parenteser, x varierer over $\mathbb{J}$ og y tar JA/NEI som mulige verdier.
- Pseudokoden, som egentlig er en repetisjon av hva vi har sett på før, kommer på neste side.

1 *Input* n
2 *Input* $w = v_1 \cdots v_n$
3 $x \leftarrow 0$
4 $y \leftarrow JA$
5 **For** $i = 1$ **to** n **do**
  5.1 **If** $v_i = ($ **then**
    5.1.1 $x \leftarrow x + 1$
    else
    5.1.2 $x \leftarrow x - 1$
  5.2 **If** $x < 0$ **then**
    5.2.1 $y \leftarrow NEI$
6 **If** $x > 0$ **then**
  6.1 $y \leftarrow NEI$
7 *Output* y

Merk.

- I virkelighetens verden har vi flere typer parenteser å holde styr på.
- Skal vi lage en algoritme som virker i en slik situasjon, kan vi ikke la x ta verdier i mengden av hele tall, men i mengden av lister av venstreparenteser.
- Hver gang vi treffer på en venstreparentes av et slag, legger vi den til listen.
- Hver gang vi treffer på en høyreparentes, må vi sammenlikne den med venstreparentesen ytterst i listen (hodet).
- Vi trenger et språk for håndtering av lister for å kunne skrive pseudokoder basert på denne skissen.
- For de som senere lærer om "push-down"-automater, vil det å lage en automat som realiserer denne algoritmen være en enkel oppgave.

Eksempel.

- La oss gå tilbake til den formen for induksjon som vi brukte for å vise at alle parentesuttrykk som oppfyller konklusjonene i Påstand 1 og 2 vil være korrekte.
- Da vi diskuterte kontrapositive argumenter, ga vi et eksempel på hvordan man kan vise at alle tall $n \geqslant 2$ kan faktoriseres i primtall (hvor det å konstatere at et tall er et primtall betraktes som en faktorisering).
- Det er lettere å formulere beviset hvis man kan bruke induksjon.
- La $n \geqslant 2$, og anta at påstanden holder for alle m slik at $2 \leqslant m < n$.
- Hvis n ikke er et primtall, fins tall $m < n$ og $k < n$ slik at $n = mk$.

Eksempel (Fortsatt).

- Ved antagelsen kan både m og k faktoriseres i primtall.
- Ved å bruke alle disse primtallsfaktorene sammen, får vi en primtallsfaktorisering av n.

Merk.

Dette argumentet gir oss ikke lov til å konkludere at det fins nøyaktig en måte å faktorisere et tall på, bare at det fins minst en.

- I eksemplet over viste vi en egenskap $P(n)$ ut fra antagelsen om at $P(m)$ og $P(k)$ holder for to utvalgte tall $m < n$ og $k < n$.
- Vi konkluderte med at $P(n)$ må holde for alle n.
- Vi kan gi en generell kontrapositiv begrunnelse for at det må være slik.
- Anta at det fins et tall n_1 slik at $P(n_1)$ ikke holder.
- Ved argumentet vårt må det da fins $n_2 < n_1$ slik at heller ikke $P(n_2)$ holder.
- Vi kan fortsette med å finne $n_3 < n_2$, $n_4 < n_3$ osv. slik at $P(n_i)$ ikke holder for noen $i \in \mathbb{N}$.

! Det fins imidlertid ingen strengt synkende følge av naturlige tall, så dette er umulig.

- Vi kan i prinsippet bruke induksjon som bevisform for enhver ordning som ikke tillater noen uendelig strengt synkende følge.
- Vi skal ikke presse denne sitronen lenger, det er sikkert surt nok for mange.