

# MAT1030 – Forelesning 19

## Generell rekursjon og induksjon

Roger Antonsen - 25. mars 2009

(Sist oppdatert: 2009-03-25 11:06)

### Forelesning 19

- Forrige gang så vi på induktivt definerte mengder og noen eksempler på funksjoner definert ved rekursjon over oppbyggingen av elementene i en slik induktivt definert mengde.
- Vi så på mengden av ord over et alfabet, og på hvordan visse operasjoner på ord kunne defineres ved hjelp av rekursjon.
- Avslutningsvis definerte vi mengden av utsagnslogiske formler hvor vi begrenser oss til utsagnsvariablene  $p$ ,  $q$  og  $r$ , og til bindeordene  $\neg$ ,  $\wedge$  og  $\vee$ .
- Vi så hvordan utarbeidelsen av sannhetsverditabellen til en formel kan oppfattes som en rekursiv prosess, vi bygger den opp nedenfra ved å gå til mer og mer komplekse formler.
- 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  $t : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \times \mathbb{N}$  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 oppfylbar 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  $(\neg p \wedge q) \vee \neg(p \wedge \neg r)$ :

$$\begin{aligned}
& SF((\neg p \wedge q) \vee \neg(p \wedge \neg r)) = \\
& SF(\neg p \wedge q) \vee SF(\neg(p \wedge \neg r)) = \\
& (SF(\neg p) \wedge SF(q)) \vee SF_{\neg}(p \wedge \neg r) = \\
& (SF_{\neg}(p) \wedge q) \vee (SF_{\neg}(p) \vee SF_{\neg}(\neg r)) = \\
& (\neg p \wedge q) \vee (\neg p \vee SF(r)) = \\
& (\neg p \wedge q) \vee (\neg p \vee r).
\end{aligned}$$

### 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 pseudo-kode 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  $\epsilon$  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  $w$  en 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 en 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 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 kommer på neste side.

```

1 Input n
2 Input w = v1 ⋯ vn
3 x ← 0
4 y ← JA
5 For i = 1 to n do
    5.1 If vi = ( then
        5.1.1 x ← x + 1
        else
        5.1.2 x ← x - 1
    5.2 If x < 0 then
        5.2.1 y ← NEI
6 If x > 0 then
    6.1 y ← 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  $n \geq 2$  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  $n \geq 2$ , og anta at påstanden holder for alle  $m$  slik at  $2 \leq m < n$ .
- 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.
- Eksemplene med rekursjon på oppbyggingen av ord eller formler er så sentrale at de kan bli brukt som grunnlag for oppgaver gitt til eksamen eller til Oblig. 2.
- Det neste eksemplet har ikke den samme betydningen i informatikk, og kan derfor best sees på som et eksempel for innøvelse av forståelse og ferdigheter.
- I eksemplet beskriver vi “den universelle datatype”, en universell mengde hvor vi kan finne alle *virkelige* datatyper som delmengder.

### Eksempel.

- Vi definerer de hereditært endelige mengdene HF som den minste mengden slik at
  - $\emptyset \in \text{HF}$
  - Hvis  $X = \{a_1, \dots, a_n\}$  er en endelig delmengde av HF, så er  $X \in \text{HF}$ .
- Vi definerer  $f : \text{HF} \rightarrow \mathbb{N}_0$  ved rekursjon over HF ved
  - $f(\emptyset) = 0$
  - $f(\{a_1, \dots, a_n\}) = 2^{f(a_1)} + \dots + 2^{f(a_n)}$ .

Her må vi anta at mengden er beskrevet uten gjentakelse, det vil si at alle  $a_i$ 'ene er forskjellige.

### Eksempel (Fortsatt).

- Ved induksjon over HF ser vi at  $f(X) \in \mathbb{N}_0$  for alle  $X \in \text{HF}$ .  
Hvis  $X = \emptyset$  ser vi det direkte, og hvis  $X = \{a_1, \dots, a_n\}$  kan vi bruke induksjonsantagelsen som sier at  $f(a_i) \in \mathbb{N}_0$  for alle  $i$ .
- Hvis  $k \in \mathbb{N}_0$  vil det fins en og bare en  $X \in \text{HF}$  slik at  $f(X) = k$ .
- Her bruker vi induksjon på  $k$ .  
For  $n = 0$  ser vi at det bare er  $X = \emptyset$  som kan gi  $f(X) = 0$  siden  $2^{f(a_i)} > 0$  for alle mulige  $a_i \in X$ .  
For  $k > 0$ , fins det en og bare en måte å skrive  $k$  som en sum av forskjellige 2'erpotenser på, og hver eksponent  $k_i$  vil komme fra en og bare en mengde  $a_i$ .  
Det betyr at vi kan finne  $X$  slik at  $f(X) = k$  fra  $k$ .

### Oppgave.

La  $HF$  og  $f$  være definert som i eksemplet over, og la  $S : HF \rightarrow HF$  være definert ved

$$S(X) = X \cup \{X\}.$$

- Vis at  $X \in HF \Rightarrow S(X) \in HF$ .
- La  $X, Y \in HF$ .  
Vis at  $X \in Y \Rightarrow f(X) < f(Y)$ .
- Forklar hvorfor det ikke fins noen  $X \in HF$  slik at  $X \in X$
- La  $N \subseteq HF$  være den minste mengden slik at  $\emptyset \in N$  og slik at hvis  $X \in N$  så vil  $S(X) \in N$ .  
Drøft sammenhengen mellom  $N$  og  $\mathbb{N}_0$ .

- Hvis vår induktive definisjon av mengden av utsagnslogiske formler skulle vært gitt i en lærebok i informatikk på et avansert nivå, eller i en forskningsartikkel, ville den sett ut som noe slikt:

Utsagnsvariabel  $v$

$$v ::= p \mid q \mid r$$

Formel  $A$

$$A ::= v \mid \neg A \mid (A \wedge A) \mid (A \vee A)$$

- De to første linjene skal leses som følger:

I denne definisjonen lar vi  $v$  betegne en vilkårlig utsagnsvariabel.

$v$  kan stå for  $p$ ,  $q$  eller  $r$ .

- De to neste linjene skal leses som følger:

I denne definisjonen lar vi  $A$  betegne en vilkårlig formel.

En formel kan enten være en utsagnsvariabel, fremkommet fra en annen formel ved å skrive  $\neg$  foran, eller fremkommet fra to andre formler ved å skrive  $\wedge$  eller  $\vee$  mellom, og parenteser rundt.

- Vi kan bruke den samme effektive notasjonen for å definere mengden av korrekte parentesuttrykk:

Parentesuttrykk  $P$

$$P ::= e \mid (P) \mid PP$$

som uttrykker at vi får parentesuttrykkene ved å starte med det tomme ordet  $e$  og deretter enten sette parenteser rundt et korrekt uttrykk eller sette to korrekte uttrykk sammen.

- Som tidligere nevnt, liker informatikere og logikere å la de naturlige tallene starte med 0, altså å arbeide med  $\mathbb{N}_0$  i stedet for  $\mathbb{N}$ .

- Følgende definisjon er påtruffet i informatikkliteratur:

$n : NAT$

$$n ::= 0 \mid S(n).$$

- Dette skal leses som at vi definerer en datatype  $NAT$  ved å la  $n$  stå for et vilkårlig objekt, og vi finner objektene i  $NAT$ , enten som *symbolet* 0 eller som en *symbolsekvens*  $S(w)$  hvor  $w$  er en symbolsekvens vi allerede vet er av type  $NAT$ .