Hoare Logic

Michael R. Hansen

mrh@imm.dtu.dk

Informatics and Mathematical Modelling

Technical University of Denmark

Oversigt

- Hoare Logic
- Prædikat transformere og Weakest Precondition

Disse foils består af et uddrag af M. Hansens foils “Design by Contract” fra 2007. De to sidste sider er ændret/udvidet af A. Haxthausen.

Kort om Prædikat Logik (I)

Atomare formler dannes fra *relationer*, *funktioner*, *variable* og *konstanter*. F.eks.

- $x < 2, x + y \geq 5, A[i] + A[i+1] > x$

En tilstand en (type-rigtig) tildeling af værdier til variable. F.eks.

- $x \mapsto 3, y \mapsto 5, i \mapsto 3, A_3 \mapsto 3, A_4 \mapsto 1$

En formel er enten sand eller falsk for en given tilstand.

Kort om Prædikat Logik (II)

Udsagns-logiske formler dannes ud fra de atomare formler ved

- *Konjunktion* $P \wedge Q$ — læses "P og Q"
  Både P og Q skal være sande.
- *Disjunktion* $P \vee Q$ — læses "P eller Q"
  Enten P eller Q eller begge skal være sande.
- *Implikation* $P \Rightarrow Q$ — "læses P medfører Q"
  Når P er sand, så skal Q også være sand
- *Negation* $\neg P$ — læses "not P"
  P er falsk

Kort om Prædikat Logik (III)

Predikat-logiske formler dannes ved hjælp af *al kvantor* $\forall$ og *eksistens kvantor* $\exists$:

- $\forall x.P(x)$ — læses "for alle x så er $P(x)$ sand"
- $\exists x.P(x)$ — læses: "der findes et x, hvorom $P(x)$ er sand"

Notationen $P(x)$ antyder at x kan forekomme i P.

Andre former $\forall x \in A.P(x)$ og $\exists x \in A.P(x)$, hvor A er en mængde.

Eksempler:

- $\forall x \in N.\exists y \in N.y < x$?
- $\forall i.0 < i < len(A) \Rightarrow A[i-1] < A[i]$

Hoare Logic

Formelt system (axiomer og inferensregler) til ressonnering omkring assertions $\{P\}\ S\ \{Q\}$. Hoare 1969

Intuitionen bag $\{P\}\ S\ \{Q\}$: partiel korrekthed

- Hvis S starter i i tilstand hvor P holder og S terminerer, så holder Q i sluttilstanden.

Bemærk: Udsagnet $\{P\}\ S\ \{Q\}$ giver ingen information ved ikke terminering.

Hoare's system er en elegant samling af simple regler, som hver især indfanger en væsentlig egenskab ved en sætning.

Hoare Logic — bevissystem

Axiomer for skip og assignment:

$$\frac{}{\{P\}\ \textbf{skip}\ \{P\}} \qquad \frac{}{\{P[x/E]\}\ x := E\ \{P\}}$$

- $P[x/E]$: erstat alle forekomster af x i P med E

Regler for de sammensatte sætninger:

Sekventiel: $\dfrac{\{P\}\ S\ \{Q\}\ ,\ \{Q\}\ T\ \{R\}}{\{P\}\ S;T\ \{R\}}$ Valg: $\dfrac{\{B \wedge P\}\ S\ \{Q\}\ ,\ \{\neg B \wedge P\}\ T\ \{Q\}}{\{P\}\ \textbf{if}\ B\ \textbf{then}\ S\ \textbf{else}\ T\ \textbf{endif}\ \{Q\}}$

Løkke: $\dfrac{\{P \wedge B\}\ S\ \{P\}}{\{P\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{done}\ \{\neg B \wedge P\}}$ P kaldes *løkkeinvarianten*

monotonicitet: $\dfrac{P' \Rightarrow P\ ,\ \{P\}\ S\ \{Q\}\ ,\ Q \Rightarrow Q'}{\{P'\}\ S\ \{Q'\}}$

Eksempel: z = max(x,y)

Vi ønsker at bevise:

$$\{true\}\ \texttt{if}\ x < y\ \texttt{then}\ z := y\ \texttt{else}\ z := x\ \texttt{endif}\ \{z = max(x,y)\}$$

Bevistrin:

1. $\{y = max(x,y)\}\ z := y\ \{z = max(x,y)\}$ — axiom – tildeling
2. $\{x = max(x,y)\}\ z := x\ \{z = max(x,y)\}$ — axiom – tildeling
3. $\{x < y\}\ z := y\ \{z = max(x,y)\}$ — 1., monotonicitet $(x < y \Rightarrow y = max(x,y))$
4. $\{\neg(x < y)\}\ z := x\ \{z = max(x,y)\}$ — 2., monotonicitet $(x \geq y \Rightarrow x = max(x,y))$

Det ønskede resultat følger af 3., 4. og reglen for valg.

Bevisskitser

Korrekthedsbeviset kan skitseres ved at indføre prædikater (assertions) i programteksten.

En forekomst af et prædikat i en programteksten er et udsagn:

- hver eneste gang programudførslen når det pågældende sted, så opfylder tilstanden prædikatet.

En *bevisskitse* for det simple maximum-program:

{true }
    if x<y
        then $\{x < y\}$ z := y { z = max(x,y) }
        else $\{x \geq y\}$ z := x { z = max(x,y) }
{ z = max(x,y) }

Eksempel: $s = \Sigma_{i=0}^{n-1} A_i$

Givet array A af n heltal $A_0, A_1, \ldots, A_{n-1}$.

$\{n \geq 0\}$ pre-condition

k := 0; s := 0;

$\{s = \Sigma_{i=0}^{k-1} A_i \wedge 0 \leq k \leq n\}$ invariant

while $k \neq n$ do

$\{k \neq n \wedge s = \Sigma_{i=0}^{k-1} A_i \wedge 0 \leq k \leq n\}$ guard og invariant

(s := s + A_k ; k := k+1)

$\{s = \Sigma_{i=0}^{k-1} A_i \wedge 0 \leq k \leq n\}$ invariant

$\{k = n \wedge s = \Sigma_{i=0}^{k-1} A_i \wedge 0 \leq k \leq n\}$ not guard og invariant

$\{s = \Sigma_{i=0}^{n-1} A_i\}$ post-condition

Regel for terminering af løkker

Ved total korrekthed kræves yderligere et bevis for at programmer terminerer.

Udover invarianten P indføres et udtryk t, hvori variable kan indgå, og som angiver et naturligt tal:

$$\frac{\{P \wedge B \wedge t = v\}\ S\ \{P \wedge t < v\} \qquad P \Rightarrow t \geq 0}{\{P\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{done}\ \{\neg B \wedge P\}}$$

- reglen udtrykker, at t skal aftage med mindst 1 per løkkegennemløb

I foregående eksempel: $t = n - k$, da

- $s = \Sigma_{i=0}^{k-1} A_i \wedge 0 \leq k \leq n \ \Rightarrow t \geq 0$, og
- t aftager med 1 per gennemløb.

Prædikat Transformere

- Prædikat transformer bruges til at beskrive programmers mening. Dijkstra 75
- En *prædikat transformer* p mapper et givet program S og prædikat Q til et andet prædikat $P = p(S, Q)$, således at $\{P\}\ S\ \{Q\}$.
- *Weakest Precondition* wp
  er en "kanonisk" prædikat transformer givet ved:
  - $\{wp(S, Q)\}\ S\ \{Q\}$
  - hvis $\{P'\}\ S\ \{Q\}$ så skal der gælde: $P' => wp(S, Q)$

  Dvs. $wp(S, Q)$ er det svageste pre-betingelse P, der sikrer, at Q gælder efter udførelsen af S

Eksempel: $wp(x := x + y, x > 7) = x + y > 7$