Verificação Automática de Programas

Consideremos o seguinte programa para calcular $\sum_{m=1}^{100} m$:

```
x:=0;
y:=1;
(while y!=101 do x:=x+y;y:=y+1)
```

- Como podemos provar que este programa termina com $x = \sum_{m=1}^{100} m$.
- Correr o programa seguindo a sua semântica operacional é uma opção.
- Mas o que acontece se mudarmos a condição do `while`, para `y!=c`, para um determinado `c` inicializado no programa?
- Correr o programa, para sucessivos valores de `c` não é opção.

Verificação de Programas considerando Sistemas dedutivos

- Dado um programa e uma especificação, verificar se o primeiro satisfaz a segunda (sem executar o pprograma!!).
- Vamos considerar Lógicas de Floyd-Hoare em que as especificações são baseadas em pré e pós-condições:

  > Uma fórmula é uma asserção de que se uma pré-condição se verifica antes da execução do programa, então a pós-condição terá de se verificar após a sua execução.

- Para o exemplo anterior:

  `x:=0; y:=1;` $\{$`x=0 and y=1`$\}$ `(while y!=101 do x:=x+y;y:=y+1)` $\{$`x=`$\sum_{0}^{100}$ `n`$\}$

Uma linguagem imperativa simples - While

Categorias sintácticas

- **Num** inteiros, n
- **Bool** valores de verdade, `true` e `false`
- **Var** variáveis, x
- **Aexp** expressões aritméticas, E
- **Bexp** expressões booleanas, B
- **Com** comandos, C

BNFs (básicas)

Para n em **Num** e x em **Var**

$$\begin{array}{lcl} E & ::= & n \mid x \mid E + E \mid E - E \mid E \times E \\ B & ::= & \texttt{true} \mid \texttt{false} \mid E = E \mid E < E \mid !B \mid B \ \wedge \ B \\ C & ::= & \texttt{skip} \mid x := E \mid C \ ; \ C \mid \texttt{if } B \texttt{ then } C \texttt{ else } C \mid \texttt{while } B \texttt{ do } C \end{array}$$

Semântica Informal

As expressões denotam valores (inteiros ou booleanos). Para a avaliação duma expressão é necessário saber o valor das variáveis.

Um **estado** s é uma função que associa a cada variável um valor.

Neste caso, conjunto de estados pode ser visto como o conjunto de funções de **Var** $\longrightarrow \mathbb{Z}$.

Os comandos são avaliados num estado e podem alterar o estado.

A semântica de um programa é o estado em que termina.

A semântica de cada comando é a habitual...

Correcção Parcial e Total

Pretende-se agora verificar que um programa tem determinadas propriedades e não tanto o significado do programa.

Em particular, propriedades de *correcção parcial* :

Se no estado s se verifica ϕ, se depois de se executar o programa C o estado for s' então verifica-se ϕ', caso o programa termine.

correção parcial + terminação = correcção total

Dada a indecidibilidade da terminação genérica de programas as propriedades de *correção parcial* são muito importantes na verificação formal de software.

Asserções - Triplos de Hoare

As propriedades de correção parcial de programas são **asserções** da forma:

$$\{\phi\}\, C\, \{\psi\}$$

onde C é um comando e ϕ e ψ são predicados duma lógica de primeira ordem.O predicado ϕ é uma *pré-condição* e ψ é uma *pós-condição*. Informalmente a asserção é válida se:

- se ϕ se verifica no estado inicial

- se a execução de C termina num estado s'
- então ψ verifica-se no estado s'

Exemplos

$\{x = 1\}\texttt{x} := \texttt{x} + \texttt{1}\{x = 2\}$ esta asserção é verdadeira

$\{x = 1\}\texttt{y} := \texttt{x}\{y = 1\}$ esta asserção é verdadeira

$\{x = 1\}\texttt{y} := \texttt{x}\{y = 2\}$ esta asserção é falsa

$\{x = x_0 \ \wedge \ y = y_0\}\texttt{r} := \texttt{x}\,;\texttt{x} := \texttt{y}\,;\texttt{y} := \texttt{r}\{x = y_0 \ \wedge \ y = x_0\}$

As variáveis x_0 e y_0 são chamadas variáveis lógicas.

$\{\top\}C\{\psi\}$ se C terminar ψ verifica-se

$\{\phi\}C\{\top\}$ é sempre verdadeira para qualquer C e ϕ.

Exemplo

```
x:=0; y:=1;
{x=0 and y=1}
(while y!=101 do x:=x+y;y:=y+1)
{x=∑_0^100 n}
```

Pretende-se inferir que $x = \sum_{m=1}^{100} m$ sabendo que antes do ciclo `while` tinhamos $y = 0$ e $x = 1$.É fácil ver que no fim do ciclo $y = 101$, mas queremos o valor de x! Temos que construir um *invariante* do ciclo: no início de cada iteração temos que

$$x = 1 + 2 + 3 + \cdots + (y - 1)$$

Linguagem das Condições

Numa asserção, $\{\phi\}C\{\psi\}$, ϕ, ψ são fórmulas $\phi, \psi, \ldots$ duma linguagem de lógica de primeira ordem para a aritmética, ou seja:

- constantes 0 e 1 (os inteiros decimais são abreviaturas)
- com símbolos funcionais $-$,$+$, $-$ e $\times$
- com símbolos de predicado $<$, $=$
- os habituais símbolos lógicos: operadores e quantificadores (onde os quantificadores só ligam as chamadas variáveis lógicas)

São interpretadas nos naturais numa estrutura $\mathcal{N} = (\mathbb{N}, \cdot)$ e os estados s, correspondem a atribuições de valores às variáveis.

Se $\mathcal{N} \models_s \phi$, dizemos que s satifaz ϕ, i.e., $s \models \phi$.

Por exemplo, se $s(x) = -2$, $s(y) = 5$, $s(z) = -1$,

$s \models \neg(x + y < z)$ verifica-se

$s \models y - x \times z < z$ não se verifica

Correcção parcial e total

Correcção parcial

Um triplo $\{\phi\}C\{\psi\}$ é satisfeito para a correcção parcial se para todos os estados que satisfazem ϕ, o estado que resulta de executar C satisfaz ψ, desde que C termine, $\models_{par} \{\phi\}C\{\psi\}$.

Nota que `while true do x := 0` satisfaz todas as asserções.

Correcção total

Um triplo $\{\phi\}C\{\psi\}$ é satisfeito para a correcção total se para todos os estados que satisfazem ϕ, é garantido que C termina e que o estado resultante satisfaz ψ, $\models_{tot} \{\phi\}C\{\psi\}$.

Neste caso o ciclo anterior não se verifica para nenhuma asserção.

Sistema dedutivo (*cálculo*) para a correcção parcial

Um sistema dedutivo é constituído por um conjunto de axiomas e um conjunto de regras de inferência.Uma derivação (demonstração) é uma sequência finita de aplicações das regras e dos axiomas.Se uma asserção $\{\phi\}C\{\psi\}$ for derivada pelo calculus de correcção parcial dizemos que $\vdash_{par} \phi C\{\psi\}$ é válido. O cálculo é integro se:

$$\vdash_{par} \{\phi\}C\{\psi\} \text{ implica que } \models_{par} \{\phi\}C\{\psi\}.$$

Sistema dedutivo para a correcção parcial

Lógica de Hoare

$[skip_p\ ]$

$$\{\phi\}\,\texttt{skip}\,\{\phi\}$$

$[ass_p\ ]$

$$\{\phi[E/x]\}\,x := E\,\{\phi\}$$

$[comp_p\ ]$

$$\frac{\{\phi\}\, C_1\, \{\eta\} \qquad \{\eta\}\, C_2\, \{\psi\}}{\{\phi\}\, C_1; C_2\, \{\psi\}}$$

Exemplos

Exercício 14.1. *Deduzir as seguintes asserções*

- $\{x = 1\}\mathtt{x} := \mathtt{x} + \mathtt{1}\{x = 2\}$
- $\{x = 1\}\mathtt{y} := \mathtt{x}\{y = 1\}$

$\diamond$

Sistema dedutivo (*calculus*) para a correção parcial

Lógica de Hoare (cont.)

$[if_p\ ]$

$$\frac{\{\phi\ \wedge\ B\}\, C_1\, \{\psi\} \qquad \{\phi\ \wedge\ \neg B\}\, C_2\, \{\psi\}}{\{\phi\}\, \mathtt{if}\, B\, \mathtt{then}\, C_1\, \mathtt{else}\, C_2\, \{\psi\}}$$

$[while_p\ ]$

$$\frac{\{\psi\ \wedge\ B\}\, C\, \{\psi\}}{\{\psi\}\, \mathtt{while}\, B\, \mathtt{do}\, C\, \{\psi\ \wedge\ \neg B\}}$$

$[cons_p\ ]$

$$\frac{\vdash \phi' \to \phi \qquad \{\phi\}\, C\, \{\psi\} \qquad \vdash \psi \to \psi'}{\{\phi'\}\, C\, \{\psi'\}}$$

Exemplos

Exemplo 14.1. *Mostrar que* $\vdash_{par} \{\top\}z := x; z := z + y; u := z\{u = x + y\}$

$$\dfrac{\dfrac{\{x + y = x + y\}z := x\{z + y = x + y\} \qquad \dfrac{\{z + y = x + y\}z := z + y\{z = x + y\} \qquad \{z = x + y\}u := z\{u = x + y\}}{\{z + y = x + y\}z := z + y; u := z\{u = x + y\}}}{\{x + y = x + y\}z := x; z := z + y; u := z\{u = x + y\}}}{\{\top\}z := x; z := z + y; u := z\{u = x + y\}}$$

Cálculo para a correcção parcial - Exemplos

Exercício 14.2. *Mostrar que*

$$\vdash_p \{x := r + (y \times q)\} r := r - y; q := q + 1 \{x := r + (y \times q)\}$$

◇

Exercício 14.3. *Mostrar que*

$$\vdash_p \{\top\} z := x + 1;\ \texttt{if}\ z - 1 = 0\ \texttt{then}\ y := 1\ \texttt{else}\ y := z \{y = x + 1\}$$

◇

`tableaux` para a correcção parcial

Seja $C = C_1; C_2; \ldots; C_n$ e que queremos $\vdash_p \{\phi\} C \{\psi\}$. Podemos considerar vários problemas da forma $\vdash_p \{\phi_i\} C_i \{\phi_{i+1}\}$. Para tal anotamos os comandos que constituem C com fórmulas ϕ_i e consideramos um `tableaux` de prova da forma:

$$\begin{array}{ll}
\{\phi_0\} & \\
C_1; & \\
\{\phi_1\} & \text{justificação} \\
C_2; & \\
\vdots & \\
\{\phi_{n-1}\} & \text{justificação} \\
C_n; & \\
\{\phi_n\} &
\end{array}$$

E mostrar que $\vdash_p \{\phi_i\} C_{i+1}; \{\phi_{i+1}\}$, começando por ϕ_n.Mas como obter cada ϕ_i?

Pré-condições mais fracas (wp)

Pré-condições mais fracas, wp

Para cada comando C e pós-condição ψ a fórmula $wp(C, \psi)$ pré-condição mais fraca que sendo verdade no estado s, garante que num estado s' obtido depois de C executar e se C terminar, a pós-condição ψ se verifica.

Isto é:

- $\models_p \{wp(C, \psi)\} C \{\psi\}$
- se $\models_p \{\phi\} C \{\psi\}$ então $\phi \to wp(C, \psi)$ (que é chamada condição de verificação)

`tableaux` para a correcção parcial

A fórmula ϕ_i obtida a partir de C_{i+1} e ϕ_{i+1} é a pré-condição mais fraca de C_{i+1}, dada a pós-condição ϕ_{i+1}, podemos escrever $wp(C_{i+1}, \phi_{i+1}) = \phi_i$.

A partir das $wp()$ e usando a regra da consequência ($cons_p$) podemos gerar automaticamente condições de verificação, que poderão ser demonstradas automaticamente ou assistidas por um demonstrador de teoremas.

De um modo geral se $\{\phi\}C\{\psi\}$ a condição de verificação é:

$$\phi \to wp(C, \psi)$$

***Pré-condições mais fracas* - ass_p**

Atribuição

$$\begin{array}{ll} \{\psi[E/x]\} & \\ x := E & \\ \{\psi\} & \quad ass_p \end{array}$$

A condição de verificação seria para $\{\phi\}x := E\{\psi\}$, $\phi \to \psi[E/x]$

Consequência

A regra $cons_p$ pode-se aplicar quando $\phi' \to \phi$ e temos $\{\phi\}\, C\, \{\psi\}$. Então neste caso admite-se no *tableaux* duas fórmulas seguidas: ϕ' e por baixo ϕ.

$$\begin{array}{ll} \{\phi'\} & \\ \{\phi\} & \quad cons_p \end{array}$$

Exercício 14.4. *Mostrar com um* tableaux $\vdash_p \{y = 5\}x := y + 1\{x = 6\}$. $\diamond$

***Pré-condições mais fracas* - if_p**

Condicional

Queremos determinar ϕ tal que $wp(\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2, \psi) = \phi$.

$$
\begin{array}{lll}
\{(B \to \phi_1) \ \wedge \ (\neg B \to \phi_2)\} & & \\
\texttt{if}\, B\, \texttt{then} & & \\
\ \{\phi_1\} & & \\
\ C_1 & & \\
\ \{\psi\} & & if_p \\
\texttt{else} & & \\
\ \{\phi_2\} & & \\
\ C_2 & & \\
\ \{\psi\} & & \\
\{\psi\} & & if_p
\end{array}
$$

Podemos calcular $\{\phi_1\}C_1\{\psi\}$ e $\{\phi_2\}C_2\{\psi\}$, e então $\phi \equiv (B \to \phi_1) \ \wedge \ (\neg B \to \phi_2)$

***Pré-condições mais fracas* - if_p**

As condições de verificação seriam as geradas por $\{\phi_1 \ \wedge \ B\}C_1\{\psi\}$ e por $\{\phi_2 \ \wedge \ \neg B\}C_2\{\psi\}$.

Exemplo 14.2. *Mostrar com um* tableaux

$$
\begin{array}{l}
\vdash_p \{\top\} \\
a := x + 1; \\
\texttt{if}\, a - 1 = 0\, \texttt{then} \\
y := 1 \\
\texttt{else} \\
y := a \\
\{y = x + 1\}
\end{array}
$$

$$
\begin{array}{lll}
\{\top\} & & \\
\{(x = 0 \to 1 = 1) \ \wedge \ (\neg(x = 0) \to x + 1 = x + 1)\} & & cons_p \\
\{(x + 1 - 1 = 0 \to 1 = x + 1) \ \wedge \ (\neg(x + 1 - 1 = 0) \to x + 1 = x + 1)\} & & cons_p \\
a := x + 1 & & \\
\{(a - 1 = 0 \to 1 = x + 1) \ \wedge \ (\neg(a - 1 = 0) \to a = x + 1)\} & & ass_p \\
\texttt{if}\, a - 1 = 0\, \texttt{then} & & \\
\{1 = x + 1\} & if'_p & \\
y := 1 & & \\
\{y = x + 1\} & ass_p & \\
\texttt{else} & & \\
\{a = x + 1\} & if'_p & \\
y := a & & \\
\{y = x + 1\} & ass_p &
\end{array}
$$

***Pré-condições mais fracas* - if_p**

Neste caso a regra de inferência usada é:

$[if'_p\ ]$

$$\frac{\{\phi_1\}\, c_1\, \{\psi\} \qquad \{\phi_2\}\, c_2\, \{\psi\}}{\{(B \to \phi_1)\ \wedge\ (\neg B \to \phi_2)\}\ \texttt{if}\, b\, \texttt{then}\, c_1\, \texttt{else}\, c_2\, \{\psi\}}$$

Exercício 14.5. *Mostra que esta regra se pode deduzir do sistema de inferência dado.* $\diamond$

***Pré-condições mais fracas* - $while_p$**

Queremos $\vdash_p \{\phi\}\texttt{while}\, B\, \texttt{do}\, C\, \{\psi\}$.

É necessário uma fórmula η tal que:

- $\phi \to \eta$
- $\eta\ \wedge\ \neg B \to \psi$ e
- $\vdash_p \{\eta\}\texttt{while}\, B\, \texttt{do}\, C\{\eta\ \wedge\ \neg B\}$

Invariante

Um invariante do ciclo $\texttt{while}\, B\, \texttt{do}\, C$ é uma fórmula η tal que $\models_p \{\eta \wedge B\}C\{\eta\}$.

***Pré-condições mais fracas* - $while_p$**

$$\begin{array}{lll}
\{\phi\} & & \\
\{\eta\} & & \\
\texttt{while}\, B\, \texttt{do} & & \\
\quad \{\eta\ \wedge\ B\} & & \\
\quad C & & \\
\quad \{\eta\} & & \\
\{\eta\ \wedge\ \neg B\} & & while_p \\
\{\psi\} & cons_p &
\end{array}$$

Dado η, as condições de verificação são $\phi \to \eta$,$\eta\ \wedge\ \neg B \to \psi$ e as condicões de verificação de $\{\eta\ \wedge\ B\}C\{\eta\}$.

Pré-condições mais fracas - $while_p$

Exemplo 14.3. *Mostrar que*

$$\vdash_p \{\top\} y := 1; z := 0; \texttt{while}\ z\,! = x\ do\ (z := z+1; y := y \times z)\{y = x!\}$$

O invariante I a considerar é : $y = z!$ e verifica as condições necessárias:

1. É implicado pela pré-condição do `while` que é $y = 1 \ \wedge \ z = 0$:
   $y = 1 \ \wedge \ z = 0 \rightarrow y = z!$
2. $y = z! \ \wedge \ z = x \rightarrow y = x!$

Começamos com I dentro do ciclo até obter I' e mostramos que $I \ \wedge \ b \rightarrow I'$.

Pré-condições mais fracas - $while_p$

$y := 1$
$z := 0$
$\{y = z!\}$?
`while` $\neg z = x$ `do`
  $\{y = z! \ \wedge \ \neg z = x\}$
  $\{y \times (z+1) = (z+1)!\}$ $cons_p$
  $z = z + 1$
  $\{y \times z = z!\}$ ass_p
  $y = y \times z$
  $\{y = z!\}$ ass_p
$\{y = x!\}$?

porque $(y = z! \ \wedge \ \neg z = x) \rightarrow y = z! \rightarrow y \times (z+1) = (z+1)!$

Pré-condições mais fracas - $while_p$

$\{\top\}$
$\{1 = 0!\}$ $cons_p$
$y := 1$
$\{y = 0!\}$ ass_p
$z := 0$
$\{y = z!\}$ ass_p
`while` $\neg z = x$ `do`
$\{y = z! \ \wedge \ \neg z = x\}$
$\{y \times (z + 1) = (z + 1)!\}$ $cons_p$
$z = z + 1$
$\{y \times z = z!\}$ ass_p
$y = y \times z$
$\{y = z!\}$ ass_p
$\{y = z! \ \wedge \ z = x\}$ $while_p$
$\{y = x!\}$ $cons_p$

Exemplos

Exercício 14.6. *Mostrar que*

$$\begin{array}{r} \vdash_p \{\top\} \\ r := x; q := 0; \\ \texttt{while}\, y \leq r\, \texttt{do} \\ r := r - y; \\ q := q + 1 \\ \{r < y \ \wedge \ x = r + (y \times q)\} \end{array}$$

$\diamond$

A expressão $x = r + (y \times q)$ é um invariante de ciclo.