Verificação de Dedutiva de Programas

1. Cálculos de correcção (Lógica de Hoare)
2. Pré-condições mais fracas e algoritmos de geração de condições de verificação.
3. Geração de obrigações de prova
4. Ferramentas para a especificação, verificação e certificação de programas imperativos.

`http://formalmethods2019.inesctec.pt/`

Verificação Automática de Programas

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

$x \leftarrow 0$;
$y \leftarrow 1$;
while $y! = 101$ **do**
    $x \leftarrow x + y$;
    $y \leftarrow y + 1$;

- 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 programa!!).
- Vamos considerar Lógicas de Floyd-Hoare (1970s) 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.

Exemplo

$x \leftarrow 0;$
$y \leftarrow 1;$
Require: $\{x = 0 \wedge y = 1\}$
while $y! = 101$ **do**
$x \leftarrow x + y;$
$y \leftarrow y + 1;$
Ensure: $\{x = \sum_{n=0}^{100} n\}$

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**

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

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 $\mathbf{Var} \to \mathbb{Z}$.

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:

$$\{\varphi\}\, C\, \{\psi\}$$

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

$\{x = 1\}\mathtt{x} \leftarrow \mathtt{x} + \mathtt{1}\{x = 2\}$ esta asserção é verdadeira

$\{x = 1\}\mathtt{y} \leftarrow \mathtt{x}\{y = 1\}$ esta asserção é verdadeira

$\{x = 1\}\mathtt{y} \leftarrow \mathtt{x}\{y = 2\}$ esta asserção é falsa

$\{x = x_0 \ \wedge \ y = y_0\}\mathtt{r} \leftarrow \mathtt{x}\,;\mathtt{x} \leftarrow \mathtt{y}\,;\mathtt{y} \leftarrow \mathtt{r}\{x = y_0 \ \wedge \ y = x_0\}$

As variáveis x_0 e y_0 são chamadas variáveis lógicas.

$\{\mathsf{true}\}C\{\psi\}$ se C terminar ψ verifica-se

$\{\varphi\}C\{\mathsf{true}\}$ é sempre verdadeira para qualquer C e φ.

Exemplo

$x \leftarrow 0;$
$y \leftarrow 1;$
Require: $\{x = 0 \wedge y = 1\}$
while $y! = 101$ **do**
$x \leftarrow x + y;$
$y \leftarrow y + 1;$
Ensure: $\{x = \sum_{n=0}^{100} n\}$

- Pretende-se inferir que $x = \sum_{m=1}^{100} m$ sabendo que antes do ciclo `while` tínhamos $x = 0$ e $y = 1$.
- 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, $\{\varphi\}C\{\psi\}$, φ, ψ são fórmulas $\varphi, \psi, \ldots$ 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 $\times$ (para termos)
- com símbolos de predicado $<$, $=$ (para predicados)
- 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 $\mathcal{N} \models_s \varphi$, dizemos que s satifaz φ, i.e., $s \models \varphi$.

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

Um triplo $\{\varphi\}C\{\psi\}$ 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,

$$\models_{par} \{\varphi\}C\{\psi\}.$$

Nota que

while true **do**
    $x \leftarrow 0$;

satisfaz todas as asserções.

Correcção total

Um triplo $\{\varphi\}C\{\psi\}$ 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,

$$\models_{tot} \{\varphi\}C\{\psi\}$$

Neste caso

while true **do**
    $x \leftarrow 0$;

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 $\{\varphi\}C\{\psi\}$ for derivada pelo calculus de correcção parcial dizemos que
  $$\vdash_{par} \varphi C\{\psi\}$$
  é *válido*.
- O cálculo é *integro* se:
  $$\vdash_{par} \{\varphi\}C\{\psi\} \text{ implica que } \models_{par} \{\varphi\}C\{\psi\}.$$

Sistema dedutivo para a correcção parcial

Lógica de Hoare

$[skip_p\ ]$

$$\{\varphi\}\,\mathtt{skip}\,\{\varphi\}$$

$[ass_p\ ]$

$$\{\varphi[E/x]\}\,x \leftarrow E\,\{\varphi\}$$

$[comp_p\ ]$

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

onde $\varphi[E/x]$ é a fórmula que se obtem substituindo x por E.

Exemplos

Exemplo 19.1. *Mostrar que* $\vdash_{par} \{\mathsf{true}\}z \leftarrow x; z \leftarrow z + y; u \leftarrow z\{u = x + y\}$

$$\dfrac{\dfrac{\{x+y=x+y\}z \leftarrow x\{z+y=x+y\} \qquad \dfrac{\{z+y=x+y\}z \leftarrow z+y\{z=x+y\} \qquad \{z=x+y\}u \leftarrow z\{u=x+y\}}{\{z+y=x+y\}z \leftarrow z+y;\, u \leftarrow z\{u=x+y\}}\, comp_p}{\{x+y=x+y\}z \leftarrow x;\, z \leftarrow z+y;\, u \leftarrow z\{u=x+y\}}\, comp_p}{\{\mathsf{true}\}z \leftarrow x;\, z \leftarrow z+y;\, u \leftarrow z\{u=x+y\}}\, cons_p$$

Exemplos

Exercício 19.1. *Deduzir as seguintes asserções*

- $\{x = 1\}\mathtt{x} \leftarrow \mathtt{x} + \mathtt{1}\{x = 2\}$
- $\{x = 1\}\mathtt{y} \leftarrow \mathtt{x}\{y = 1\}$
- $\{x = x_0 \wedge y = y_0\}\mathtt{r} \leftarrow \mathtt{x}\,;\mathtt{x} \leftarrow \mathtt{y}\,;\mathtt{y} \leftarrow \mathtt{r}\{x = y_0 \wedge y = x_0\}$

$\diamond$

Sistema dedutivo (*calculus*) para a correção parcial

Lógica de Hoare (cont.)

$[if_p\ ]$

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

$[while_p\ ]$

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

A ψ chama-se o *invariante de ciclo*

$[cons_p\ ]$

$$\frac{\vdash \varphi' \to \varphi \quad \{\varphi\}\, C\, \{\psi\} \quad \vdash \psi \to \psi'}{\{\varphi'\}\, C\, \{\psi'\}}$$

Cálculo para a correcção parcial - Exemplos

Exercício 19.2. *Mostrar que*

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

$\diamond$

Exercício 19.3. *Mostrar que*

$$\vdash_p \{\texttt{true}\} z \leftarrow x + 1;\ \texttt{if}\ z - 1 = 0\, \texttt{then}\, y \leftarrow 1\, \texttt{else}\, y \leftarrow z \{y = x + 1\}$$

$\diamond$

`tableaux` para a correcção parcial

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

$\{\varphi_0\}$
$C_1;$
$\{\varphi_1\}$ justificação
$C_2;$
$\vdots$
$\{\varphi_{n-1}\}$ justificação
C_n
$\{\varphi_n\}$

Mostrar que $\vdash_p \{\varphi_i\}C_{i+1};\{\varphi_{i+1}\}$, 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 $wp(C,\psi)$ é a 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 $\models_p \{\varphi\}C\{\psi\}$ então $\varphi \to wp(C,\psi)$ (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

$$wp(C_{i+1},\varphi_{i+1}) = \varphi_i.$$

- A partir das $wp()$ e usando a regra da consequência ($cons_p$) podemos gerar automaticamente condições de verificação,
- que poderão ser demonstradas automaticamente ou a sua demonstração assistida por um demonstrador de teoremas.
- De um modo geral se $\{\varphi\}C\{\psi\}$ a condição de verificação é:

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

Pré-condições mais fracas - ass_p

Atribuição

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

Temos que

$$wp(x \leftarrow E, \psi) = \psi[E/x].$$

A condição de verificação para $\{\varphi\}x \leftarrow E\{\psi\}$, seria

$$\varphi \rightarrow \psi[E/x]$$

Pré-condições mais fracas

Consequência

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

$$\begin{array}{ll} \{\varphi'\} & \\ \{\varphi\} & cons_p \end{array}$$

Exercício 19.4. *Mostrar com um* tableaux $\vdash_p \{y = 5\}x \leftarrow y + 1\{x = 6\}$. $\diamond$

Pré-condições mais fracas - if_p

Queremos determinar $wp(\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2, \psi)$.

$$\begin{array}{ll} \{(B \rightarrow \varphi_1) \wedge (\neg B \rightarrow \varphi_2)\} & \\ \texttt{if}\, B\, \texttt{then} & \\ \quad \{\varphi_1\} & \\ \quad C_1 & \\ \quad \{\psi\} & if_p \\ \texttt{else} & \\ \quad \{\varphi_2\} & \\ \quad C_2 & \\ \quad \{\psi\} & \\ \{\psi\} & if_p \end{array}$$

Podemos calcular $\{\varphi_1\}C_1\{\psi\}$ e $\{\varphi_2\}C_2\{\psi\}$, e então

$$wp(\texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2, \psi) = (B \to \varphi_1) \ \wedge \ (\neg B \to \varphi_2)$$

As condições de verificação são as geradas por $\{\varphi_1 \ \wedge \ B\}C_1\{\psi\}$ e por $\{\varphi_2 \ \wedge \ \neg B\}C_2\{\psi\}$.

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

Exemplo 19.2. *Mostrar com um* tableaux

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

$\{\mathsf{true}\}$
$\{(x = 0 \to 1 = 1) \ \wedge \ (\neg(x = 0) \to x + 1 = x + 1)\} \qquad cons_p$
$\{(x + 1 - 1 = 0 \to 1 = x + 1) \ \wedge \ (\neg(x + 1 - 1 = 0) \to x + 1 = x + 1)\}\ cons_p$
$a \leftarrow x + 1$
$\{(a - 1 = 0 \to 1 = x + 1) \ \wedge \ (\neg(a - 1 = 0) \to a = x + 1)\} \qquad ass_p$
$\texttt{if}\, a - 1 = 0\, \texttt{then}$
$\{1 = x + 1\} \qquad if'_p$
$y \leftarrow 1$
$\{y = x + 1\} \qquad ass_p$
$\texttt{else}$
$\{a = x + 1\} \qquad if'_p$
$y \leftarrow a$
$\{y = x + 1\} \qquad ass_p$

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

Neste caso a regra de inferência usada é:

$[if'_p \ ]$

$$\frac{\{\varphi_1\}\, C_1\, \{\psi\} \qquad \{\varphi_2\}\, C_2\, \{\psi\}}{\{(B \to \varphi_1) \ \wedge \ (\neg B \to \varphi_2)\}\, \texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2\, \{\psi\}}$$

Exercício 19.5. *Mostra que esta regra se pode deduzir do sistema de inferência dado.* $\diamond$

Pré-condições mais fracas - $while_p$

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

É necessário uma fórmula η tal que:

- $\varphi \to \eta$
- $\eta \ \wedge \ \neg B \to \psi$ 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 - $while_p$

$\{\varphi\}$
$\{\eta\}$
$\texttt{while}\, B\, \texttt{do}$
  $\{\eta \ \wedge \ B\}$
  C
  $\{\eta\}$
$\{\eta \ \wedge \ \neg B\}$ $while_p$
$\{\psi\}$ $cons_p$

Temos

$$wp(\texttt{while}\, B\, \texttt{do}\, C\, , \psi) = \eta$$

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

Pré-condições mais fracas - $while_p$

Exemplo 19.3. *Mostrar que*

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

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

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

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 - $while_p$

$y \leftarrow 1$
$z \leftarrow 0$
$\{y = z!\}$?
`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 - $while_p$

$\{\mathsf{true}\}$
$\{1 = 0!\}$ $cons_p$
$y \leftarrow 1$
$\{y = 0!\}$ ass_p
$z \leftarrow 0$
$\{y = z!\}$ ass_p
$\mathtt{while}\ \neg z = x\ \mathtt{do}$
$\{y = z!\ \wedge\ \neg z = x\}$
$\{y \times (z+1) = (z+1)!\}$ $cons_p$
$z \leftarrow z + 1$
$\{y \times z = z!\}$ ass_p
$y \leftarrow y \times z$
$\{y = z!\}$ ass_p
$\{y = z!\ \wedge\ z = x\}$ $while_p$
$\{y = x!\}$ $cons_p$

Exemplos

Exercício 19.6. *Mostrar que*

$$\begin{aligned} &\vdash_p \{\mathsf{true}\} \\ &r \leftarrow x; q \leftarrow 0; \\ &\mathtt{while}\ y \leq r\ \mathtt{do} \\ &\quad r \leftarrow r - y; \\ &\quad q \leftarrow q + 1 \\ &\{r < y\ \wedge\ x = r + (y \times q)\} \end{aligned}$$

$\diamond$

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

Exemplo

Exercício 19.7. *Mostra que*
$\{x \geq 0\} z \leftarrow x; y \leftarrow 0;\ \mathtt{while}\ \neg z = 0\ \mathtt{do}\ (y \leftarrow y + 1; z \leftarrow z - 1)\{x = y\}$.
$\diamond$