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.

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 **Var** $\rightarrow \mathbb{Z}$.

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 $y = 0$ 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, $\{\varphi\}C\{\psi\}$, 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 ← 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

[$skip_p$]

$$\{\varphi\}\,\texttt{skip}\,\{\varphi\}$$

[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.

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*

$[cons_p\ ]$

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

Exemplos

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

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

Exemplos

Exercício 19.1. *Deduzir as seguintes asserções*

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

$\diamond$

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*

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

$\diamond$