`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:

$$
\begin{array}{ll}
\{\varphi_0\} & \\
C_1; & \\
\{\varphi_1\} & \text{justificação} \\
C_2; & \\
\vdots & \\
\{\varphi_{n-1}\} & \text{justificação} \\
C_n; & \\
\{\varphi_n\} &
\end{array}
$$

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 assistidas 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

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

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

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

Pré-condições mais fracas

Consequência

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

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

Exercício 21.1. *Mostrar com um* tableaux $\vdash_p \{y = 5\}x \leftarrow y + 1\{x = 6\}$. $\diamond$

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

Condicional

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

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

Podemos calcular $\{\varphi_1\}C_1\{\psi\}$ e $\{\varphi_2\}C_2\{\psi\}$, e então $\varphi \equiv (B \to \varphi_1) \wedge (\neg B \to \varphi_2)$

Pré-condições mais fracas - if_p

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

Exemplo 21.1. *Mostrar com um* tableaux

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

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

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 21.2. *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 - $while_p$

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

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

Pré-condições mais fracas **-** *while$_p$*

Exemplo 21.2. *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 $y = 1\ \wedge\ z = 0$:
   $y = 1\ \wedge\ z = 0 \rightarrow y = z!$
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$_p$*

$$
\begin{array}{lll}
y \leftarrow 1 & & \\
z \leftarrow 0 & & \\
\{y = z!\} & ? & \\
\texttt{while}\, \neg z = x\, \texttt{do} & & \\
\quad \{y = z!\ \wedge\ \neg z = x\} & & \\
\quad \{y \times (z+1) = (z+1)!\} & & cons_p \\
\quad z = z+1 & & \\
\quad \{y \times z = z!\} & ass_p & \\
\quad y = y \times z & & \\
\quad \{y = z!\} & ass_p & \\
\{y = x!\} & ? &
\end{array}
$$

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

Pré-condições mais fracas **-** *while$_p$*

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

Exemplos

Exercício 21.3. *Mostrar que*

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

◇

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

Exemplo

Exercício 21.4. *Mostra que*
$\{x \geq 0\} z \leftarrow x; y \leftarrow 0;$ $\mathtt{while}\ \neg z = 0\ \mathtt{do}\ (y \leftarrow y + 1; z \leftarrow z - 1)\{x = y\}$.
◇

Integridade e Completude

Para o sistema dedutivo de Hoare, vamos considerar duas propriedades usuais em sistemas lógicos:

- **Integridade:** Cada regra deve preservar validade. O que implica (por indução nas derivações) que os teoremas obtidos correspondem a asserções válidas de correcção parcial.

$$\vdash_p \{\varphi\}C\{\psi\} \qquad \Rightarrow \qquad \models_p \{\varphi\}C\{\psi\}.$$

- **Completude:** Gostaríamos que o sistema fosse suficientemente forte para inferir todas as asserções de correção parcial válidas.

$$\models_p \{\varphi\}C\{\psi\} \qquad \Rightarrow \qquad \vdash_p \{\varphi\}C\{\psi\}.$$

Vamos começar por formalizar a noção de execução/avaliação.

Estado de execução

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.

Representamos o conjunto de estados por

$$\mathbf{State} = \mathbf{Var} \to \mathbb{Z}$$

e $s \in \mathbf{State}$ tal que $s : \mathbf{Var} \to \mathbb{Z}$.

Seja $s\ x$ ou $s(x)$ o valor da variável x no estado s. Se $v \in \mathbb{Z}$,

$$s[v/x](y) = \begin{cases} s(y) & \text{se } y \neq x \\ v & \text{se } y = x \end{cases}$$

Semântica das expressões

Aexp - Expressões aritméticas

$\mathcal{A} : \mathbf{Aexp} \to (\mathbf{State} \to Z)$

$$\begin{aligned}
\mathcal{A}[\![n]\!]s &= n \\
\mathcal{A}[\![x]\!]s &= s(x) \\
\mathcal{A}[\![E_1 + E_2]\!]s &= \mathcal{A}[\![E_1]\!]s + \mathcal{A}[\![E_2]\!]s \\
\mathcal{A}[\![E_1 - E_2]\!]s &= \mathcal{A}[\![E_1]\!]s - \mathcal{A}[\![E_2]\!]s \\
\mathcal{A}[\![E_1 \times E_2]\!]s &= \mathcal{A}[\![E_1]\!]s . \mathcal{A}[\![E_2]\!]s
\end{aligned}$$

Semântica das expressões

Bexp - Expressões booleanas

$\mathbf{T} = \{\mathbf{V}, \mathrm{F}\}$

$\mathcal{B} : \mathbf{Bexp} \to (\mathbf{State} \to \mathbf{T})$

$$\begin{aligned}
\mathcal{B}[\![\mathsf{true}]\!]s &= \mathbf{V} \\
\mathcal{B}[\![\mathsf{false}]\!]s &= \mathrm{F} \\
\mathcal{B}[\![E_1 = E_2]\!]s &= \begin{cases} \mathbf{V} & \text{se } \mathcal{A}[\![E_1]\!]s = \mathcal{A}[\![E_2]\!]s \\ \mathrm{F} & \text{se } \mathcal{A}[\![E_1]\!]s \neq \mathcal{A}[\![E_2]\!]s \end{cases} \\
\mathcal{B}[\![E_1 \leq E_2]\!]s &= \begin{cases} \mathbf{V} & \text{se } \mathcal{A}[\![E_1]\!]s \leq \mathcal{A}[\![E_2]\!]s \\ \mathrm{F} & \text{se } \mathcal{A}[\![E_1]\!]s > \mathcal{A}[\![E_2]\!]s \end{cases} \\
\mathcal{B}[\![\neg b]\!]s &= \begin{cases} \mathbf{V} & \text{se } \mathcal{B}[\![b]\!]s = \mathrm{F} \\ \mathrm{F} & \text{se } \mathcal{B}[\![b]\!]s = \mathbf{V} \end{cases} \\
\mathcal{B}[\![b_1 \wedge b_2]\!]s &= \begin{cases} \mathbf{V} & \text{se } \mathcal{B}[\![b_1]\!]s = \mathbf{V} \text{ e } \mathcal{B}[\![b_2]\!]s = \mathbf{V} \\ \mathrm{F} & \text{se } \mathcal{B}[\![b_1]\!]s = \mathrm{F} \text{ ou } \mathcal{B}[\![b_2]\!]s = \mathrm{F} \end{cases}
\end{aligned}$$

Semântica operacional natural (*big-step*)

Descreve a execução completa de cada comando.

Configurações: $\langle C, s\rangle$ ou s, onde C é um comando e s um estado $\Gamma = (\mathbf{Com} \times \mathbf{State}) \cup \mathbf{State}$

Configurações Finais: $s \in State$

Transições: $\langle C, s\rangle \longrightarrow s'$

Regras:

$$\frac{\langle C_1, s_1\rangle \longrightarrow s'_1 \ldots \langle C_n, s_n\rangle \longrightarrow s'_n}{\langle C, s\rangle \longrightarrow s'}$$

Hipóteses: $\langle C_i, s_i\rangle \longrightarrow s'_i$
Conclusão: $\langle C, s\rangle \longrightarrow s'$
Se $n = 0$ diz-se um **Axioma**.

Semântica operacional natural (*big-step*)

Semântica operacional para comandos do While

$$
\begin{array}{ll}
\texttt{att}_{sn} & \langle x \leftarrow E, s\rangle \longrightarrow s[\mathcal{A}[\![E]\!]s/x] \\
\texttt{comp}_{sn} & \dfrac{\langle C_1, s\rangle \longrightarrow s'\ ,\ \langle C_2, s'\rangle \longrightarrow s''}{\langle C_1; C_2, s\rangle \longrightarrow s''} \\
\texttt{if}^{v}{}_{sn} & \dfrac{\langle C_1, s\rangle \longrightarrow s'}{\langle \texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2, s\rangle \longrightarrow s'}\ \text{se}\ \mathcal{B}[\![B]\!]s = \mathbf{V} \\
\texttt{if}^{f}{}_{sn} & \dfrac{\langle C_2, s\rangle \longrightarrow s'}{\langle \texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2, s\rangle \longrightarrow s'}\ \text{se}\ \mathcal{B}[\![B]\!]s = \mathrm{F} \\
\texttt{while}^{v}{}_{sn} & \dfrac{\langle C, s\rangle \longrightarrow s',\ \langle \texttt{while}\ B\ \texttt{do}\ C, s'\rangle \longrightarrow s''}{\langle \texttt{while}\ B\ \texttt{do}\ C, s\rangle \longrightarrow s''}\ \text{se}\ \mathcal{B}[\![B]\!]s = \mathbf{V} \\
\texttt{while}^{f}{}_{sn} & \langle \texttt{while}\ B\ \texttt{do}\ C, s\rangle \longrightarrow s\ \text{se}\ \mathcal{B}[\![B]\!]s = \mathrm{F}
\end{array}
$$

Exemplos

Sendo $s_0 = [x = 5, y = 7]$ determinar o estado após a execução de:

$$(z \leftarrow x; x \leftarrow y); y \leftarrow z.$$

Para tal constrói-se uma **Árvore de derivação** com esse comando como raiz:

$$\dfrac{\dfrac{\langle z \leftarrow x, s_0\rangle \longrightarrow s_1 \quad \langle x \leftarrow y, s_1\rangle \longrightarrow s_2}{\langle z \leftarrow x; x \leftarrow y, s_0\rangle \longrightarrow s_2} \quad \langle y \leftarrow z, s_2\rangle \longrightarrow s_3}{\langle (z \leftarrow x; x \leftarrow y); y \leftarrow z, s_0\rangle \longrightarrow s_3}$$

onde,

$$
\begin{aligned}
s_1 &= s_0[5/z] \\
s_2 &= s_1[7/x] \\
s_3 &= s_2[5/y]
\end{aligned}
$$

Temos $s_=[z = 5, x = 7, y = 5]$.

Integridade da semântica axiomática

Teorema 21.1 (Integridade)**.** *Para todas as asserções de correcção parcial* $\{\varphi\}C\{\psi\}$,

$$\vdash_p \{\varphi\}C\{\psi\}\ \textit{implica}\ \models_p \{\varphi\}C\{\psi\}$$

A demonstração é por indução na árvore de inferência de $\vdash_p \{\varphi\}C\{\psi\}$:

- Mostrar que a propriedade se verifica para as árvores simples, i.e os **axiomas** do sistema de inferência.
- Mostrar que a propriedade se verifica para as Árvores de inferência compostas: para cada regra, supor que a propriedade se verifica para as premissas (e as condições se verificam) e mostrar que a propriedade também se verifica para a conclusão da regra.

Integridade da semântica axiomática

Caso ass_p**.** Suponhamos que $\vdash_p \{\varphi[E/x]\}x \leftarrow E\{\varphi\}$.

Seja

$$\langle x \leftarrow E, s\rangle \longrightarrow s'$$

e $s \models \varphi[E/x]$ se e só se $s[\mathcal{A}[\![E]\!]s/x] \models \varphi$. (Exercício)

Temos que provar que $s' \models \varphi$.

Por $[ass_{sn}]$ temos que $s' = s[\mathcal{A}[\![E]\!]s/x]$, e portanto

$$s' \models \varphi \text{ sse } s[\mathcal{A}[\![E]\!]s/x] \models \varphi$$

Integridade da semântica axiomática

Caso $comp_p$**.** Por hip. de indução $\models_p \{\varphi\}C_1\{\eta\}$ e $\models_p \{\eta\}C_2\{\psi\}$.

Queremos mostrar que $\models_p \{\varphi\}C_1; C_2\{\psi\}$. Sejam s e s'' estados, tal que $s \models \varphi$ e $\langle C_1; C_2, s\rangle \longrightarrow s''$. Pela regra $[comp_{sn}]$ existe s' tal que

$$\langle C_1, s\rangle \longrightarrow s' \text{ e } \langle C_2, s'\rangle \longrightarrow s''$$

De $\langle C_1, s\rangle \longrightarrow s'$, $s \models \varphi$ e $\models_p \{\varphi\}C_1\{\eta\}$, temos que $s' \models \eta$. De $\langle C_2, s'\rangle \longrightarrow s''$, $s' \models \eta$ e $\models_p \{\eta\}C_2\{\psi\}$, temos que $s'' \models \psi$. Que é o que queríamos.

Integridade da semântica axiomática

Caso if_p**.** Por hip. de indução $\models_p \{B \wedge \varphi\}C_1\{\psi\}$ e $\models_p \{\neg B \wedge \varphi\}C_2\{\psi\}$.

Para provar que

$$\models_p \{\varphi\}\, \texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2\, \{\psi\}$$

sejam s e s' estados tais que $s \models \varphi$ e $\langle \texttt{if}\, B\, \texttt{then}\, C_1\, \texttt{else}\, C_2, s\rangle \longrightarrow s'$.

Se $\mathcal{B}[\![B]\!]s = \mathbf{V}$ então por $[if_{sn}]$, temos que $\langle C_1, s\rangle \longrightarrow s'$. Então dado que $\models_p \{B \wedge \varphi\}C_1\{\psi\}$, concluímos que $s' \models \psi$.

Analogamente se conclui, caso $\mathcal{B}[\![B]\!]s = \mathrm{F}$.

Integridade da semântica axiomática

Caso *while$_p$***.** Por hip. de indução

$$\models_p \{B \ \wedge \ \varphi\}C\{\varphi\}. \tag{1}$$

Para provar que

$$\models_p \{\varphi\}\, \texttt{while}\, B\, \texttt{do}\, C\, \{\neg B \ \wedge \ \varphi\},$$

sejam s e s'' estados tais que $s \models \varphi$ e

$$\langle \texttt{while}\, B\, \texttt{do}\, C, s\rangle \ \longrightarrow \ s''.$$

Temos que mostrar que $s'' \models \neg B \ \wedge \ \varphi$. Usamos indução na árvore de derivação da semântica natural.

Integridade da semântica axiomática

Caso *while$_p$***.** Há dois casos a considerar, consoante [$while_{sn}$].

Se $\mathcal{B}[\![B]\!]s = \mathrm{F}$ então $s'' = s$ e $s'' \models (\neg B \ \wedge \ \varphi)$.

Senão, $\mathcal{B}[\![b]\!]s =$ **V** e existe s' tal que $\langle C, s\rangle \ \longrightarrow \ s'$ e $\langle \texttt{while}\, B\, \texttt{do}\, C, s'\rangle \ \longrightarrow \ s''$.

Temos que $s \models (B \ \wedge \ \varphi)$ e pela hipótese (1) temos que $s' \models \varphi$. Aplicando a hipótese de indução a $\langle \texttt{while}\, B\, \texttt{do}\, C, s'\rangle \ \longrightarrow \ s''$, temos que $s'' \models (\neg B \ \wedge \ \varphi)$, como queríamos.

Integridade da semântica axiomática

Caso *cons$_p$***.** Por hip. de indução

$$\models_p \{\varphi'\}C\{\psi'\}, \ \varphi \rightarrow \varphi', \ \text{e} \ \psi' \rightarrow \psi. \tag{2}$$

Para provar que $\models_p \{\varphi\}C\{\psi\}$, sejam s e s' tal que $s \models \varphi$ e $\langle C, s\rangle \ \longrightarrow \ s'$.

Como $s \models \varphi$ e $\varphi \rightarrow \varphi'$ então $s \models \varphi'$ e pela hipótese (2), $s' \models \psi'$. Mas como $\psi' \rightarrow \psi$, temos que $s' \models \psi$, como queríamos.

Completude da semântica axiomática

Teorema 21.2 (Incompletude de Gödel (1931))**.** *Não existe um sistema de demonstração para* ***PA*** *(aritmética), de tal forma que os teoremas coincidam com as asserções válidas de* ***PA****.*

Teorema 21.3 (Completude)**.** *Para todas as asserções de correcção parcial* $\{\varphi\}C\{\psi\}$*,*

$$\models_p \{\varphi\}C\{\psi\} \ \textit{implica} \ \vdash_p \{\varphi\}C\{\psi\}$$

Note-se que $\models \psi$, se e só se $\models \texttt{true}\}\texttt{skip}\{\psi\}$. O que significa que a completude de $\vdash_p$ contrariaria o teorema de incompletude de Gödel.

Completude da semântica axiomática

Proposição 21.1. *Não existe um sistema de demonstração para asserções de correcção parcial, de tal forma que os teoremas coincidam com as asserções de correcção parcial válidas.*

Prova: Note-se que

$$\models \{\mathsf{true}\}C\{\mathsf{false}\}$$

se e só se o comando C diverge em todos os estados.

Um sistema de demonstração para asserções de correcção parcial, poderia ser usado para confirmar que o comando diverge (não para) em todos os estados. O que é impossível (*Halting Problem*).

Completude relativa

Teorema 21.4. *O sistema de prova para correcção parcial é relativamente completo, i.e. para qualquer asserção de correcção parcial* $\{\varphi\}C\{\psi\}$*:*

$$\vdash_p \{\varphi\}C\{\psi\} \text{ se } \models_p \{\varphi\}C\{\psi\}$$

O resultado de correcção parcial relativa foi estabelecido por S. Cook (1978).

O facto de $\vdash_p \{\varphi\}C\{\psi\}$ ser uma prova depende do facto de certas asserções em **PA** serem válidas.

Para a demonstração de completude relativa ver Capítulo 7 [Winskel].

Cálculo para a correcção total

Na linguagem imperativa apresentada, o único comando que pode levar à não terminação é o comando `while`.

O *cálculo* $\vdash_{tot}$ irá ser igual ao $\vdash_p$ excepto na regra $\mathtt{while}_{tot}$.

Para demonstrar que um programa termina temos que lhe associar uma expressão estritamente decrescente, denominada variante.

No caso do `while`, podemos associar uma expressão inteira não negativa e mostrar que em cada iteração o valor dessa expressão diminui, mantendo-se não negativa: temos a certeza que `while` termina pois essa expressão só pode tomar um número finito de valores até chegar a zero!!!

No caso do factorial:

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

podemos tomar o variante $x - z$.

***Cálculo* para a correcção total**

Lógica de Hoare (correcção total)

As regras ass_{tot}, $comp_{tot}$, if_{tot} e $cons_{tot}$ coincidem com as do *cálculo* para a correcção parcial.

$[while_{tot}\ ]$

$$\frac{\{\eta\ \wedge\ B\ \wedge\ 0 \leq E\ \wedge\ E = e_0\}\, C\, \{\eta\ \wedge\ 0 \leq E\ \wedge\ E < e_0\}}{\{\eta\ \wedge\ 0 \leq E\}\, \texttt{while}\, B\, \texttt{do}\, C\, \{\eta\ \wedge\ \neg B\}}$$

onde e_0 é uma variável lógica cujo valor é o da expressão E antes da execução do comando C.

Pré condição mais fraca **-** $while_{tot}$

$\{\varphi\}$

$\{\eta\ \wedge\ 0 \leq E\}$

$\texttt{while}\, B\, \texttt{do}$

$\{\eta\ \wedge\ B\ \wedge\ 0 \leq E\ \wedge\ E = e_0\}$

C

$\{\eta\ \wedge\ 0 \leq E\ \wedge\ E < e_0\}$

$\{\eta\ \wedge\ \neg B\}$ $\qquad while_{tot}$

$\{\psi\}$ $\qquad cons_{tot}$

Exemplo

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

$\{x \geq 0\}$

$\{1 = 0! \wedge 0 \leq x - 0\}$

$\mathtt{y} \leftarrow \mathtt{1}$

$\{y = 0! \wedge 0 \leq x - 0\}$

$\mathtt{z} \leftarrow \mathtt{0}$

$\{y = z! \wedge 0 \leq x - z\}$ $\quad ass_{tot}$

$\mathtt{while}\ z\,! = x\ do$

$\{$

$\{y = z! \wedge z! = x \wedge 0 \leq x - z \wedge x - z = e_0\}$ $\quad cons_{tot}$

$\{y \times (z + 1) = (z + 1)! \wedge 0 \leq x - (z + 1) \wedge x - (z + 1) < e_0\}$ $\quad ass_{tot}$

$\mathtt{z} \leftarrow \mathtt{z} + \mathtt{1}$

$\{y \times z = z! \wedge 0 \leq x - z \wedge x - z < e_0\}$ $\quad ass_{tot}$

$\mathtt{y} \leftarrow \mathtt{y} \times \mathtt{z}$

$\{y = z! \wedge 0 \leq x - z \wedge x - z < e_0\}$

$\}$

$\{y = z! \wedge x = z\}$

$\{y = x!\}$

Como determinar um variante ?

Os variantes são mais difíceis de encontrar que os invariantes...porque não é possível saber genericamente se um programa termina

$\vdash_{tot} \{x > 0\}$

$\quad \mathtt{c} = \mathtt{x}$

$\quad \mathtt{while(c! = 1)do}$

$\quad\quad \mathtt{if(c\%2 == 0)c = c/2}$

$\quad\quad \mathtt{else\ c = 3 * c + 1}$

$\quad \{\top\}$

Será que este triplo é válido? Neste caso este triplo só estabelecia a terminação do programa...

Mas não se sabe se termina ou não!

Exemplos

Exercício 21.5. *Mostra que*

$$\begin{aligned}
\vdash_{tot} \{y > 0\}\\
\texttt{while}\, y \leq r\, \texttt{do}\\
r \leftarrow r - y;\\
q \leftarrow q + 1\\
\{\top\}
\end{aligned}$$

$\diamond$