Lógica
Aula 22

Renata Wassermann

renata@ime.usp.br

2020

Cálculo para Prova de Correção

Composição

$$\frac{(|\varphi|)\, C_1\, (|\chi|) \qquad (|\chi|)\, C_2\, (|\psi|)}{(|\varphi|)\, C_1 ; C_2\, (|\psi|)}$$

Cálculo para Prova de Correção

Composição

$$\frac{(|\varphi|)\,C_1\,(|\chi|) \qquad (|\chi|)\,C_2\,(|\psi|)}{(|\varphi|)\,C_1;C_2\,(|\psi|)}$$

Atribuição

$$\frac{}{(|\psi[E/x]|)\,x = E\,(|\psi|)}$$

Para escrever uma prova

$$\vdash_{par} (|\varphi|)P(|\psi|)$$

Seja P:

C_1;

C_2;

.

.

.

C_n;

Para escrever uma prova

$$\vdash_{par} (|\varphi|)P(|\psi|)$$

Seja P:

C_1;

C_2;

.

.

.

C_n;

$(|\varphi_n|)$

Para escrever uma prova

$$\vdash_{par} (|\varphi|)P(|\psi|)$$

Seja P:

$$C_1;$$

$$C_2;$$

$$\begin{array}{l} . \\ . \\ . \\ \quad (|\varphi_{n-1}|) \\ C_n; \\ \quad (|\varphi_n|) \end{array}$$

Para escrever uma prova

$$\vdash_{par} (|\varphi|)P(|\psi|)$$

Seja P:

$C_1;$

$C_2;$

$\quad (|\varphi_2|)$

.

.

.

$\quad (|\varphi_{n-1}|)$

$C_n;$

$\quad (|\varphi_n|)$

Para escrever uma prova

$$\vdash_{par} (|\varphi|)P(|\psi|)$$

Seja P:

$$
\begin{array}{l}
C_1; \\
\quad (|\varphi_1|) \\
C_2; \\
\quad (|\varphi_2|) \\
. \\
. \\
. \\
\quad (|\varphi_{n-1}|) \\
C_n; \\
\quad (|\varphi_n|)
\end{array}
$$

Para escrever uma prova

$$\vdash_{par} (|\varphi|)P(|\psi|)$$

Seja P:

$$
\begin{array}{l}
\quad (|\varphi_0|) \\
C_1; \\
\quad (|\varphi_1|) \\
C_2; \\
\quad (|\varphi_2|) \\
. \\
. \\
. \\
\quad (|\varphi_{n-1}|) \\
C_n; \\
\quad (|\varphi_n|)
\end{array}
$$

Para escrever uma prova

Cada φ_i deve valer no ponto em que aparece.

Para escrever uma prova

Cada φ_i deve valer no ponto em que aparece.

Cada transição

$$\begin{array}{l} \quad (|\varphi_{i-1}|) \\ C_i; \\ \quad (|\varphi_i|) \end{array}$$

usa alguma regra do cálculo e parte de φ_i para calcular a *pré-condição mais fraca* φ_{i-1}.

Cálculo para Prova de Correção

Implicação

$$\frac{\vdash \varphi' \to \varphi \qquad (|\varphi|)\,C\,(|\psi|) \qquad \vdash \psi \to \psi'}{(|\varphi'|)\,C\,(|\psi'|)}$$

Esta regra é importante para completar provas usando lógica de primeira ordem e aritmética de inteiros.

Cálculo para Prova de Correção

Implicação

$$\frac{\vdash \varphi' \to \varphi \qquad (|\varphi|)\,C\,(|\psi|) \qquad \vdash \psi \to \psi'}{(|\varphi'|)\,C\,(|\psi'|)}$$

Esta regra é importante para completar provas usando lógica de primeira ordem e aritmética de inteiros.

Ela permite escrever

$$\begin{array}{l} (|\varphi|) \\ (|\varphi'|) \end{array}$$

quando $\vdash \varphi \to \varphi'$.

Cálculo para Prova de Correção

If

$$\frac{(|\varphi \wedge B|)C_1(|\psi|) \qquad (|\varphi \wedge \neg B|)C_2(|\psi|)}{(|(\varphi)|)\texttt{if}\ \ B\ \{C_1\}\ \texttt{else}\ \ \{C_2\}(|\psi|)}$$

Cálculo para Prova de Correção

If

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

If'

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

Exemplo

$(|\top|)$

```
a = x + 1;
if (a - 1 == 0) {
  y = 1;
} else {
  y = a;
}
```

$(|y = x + 1|)$

Exemplo

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

$(|\top|)$

```
a = x + 1;

if (a - 1 == 0) {

  y = 1;
  (|y = x + 1|)
} else {

  y = a;
  (|y = x + 1|)
}
  (|y = x + 1|)   If'
```

Exemplo

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

$(|\top|)$

```
a = x + 1;

if (a - 1 == 0) {

  y = 1;
```
$(|y = x + 1|)$
```
} else {
```
$(|a = x + 1|)$
```
  y = a;
```
$(|y = x + 1|)$ Atribuição
```
}
```
$(|y = x + 1|)$ If'

Exemplo

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

$(|\top|)$

```
a = x + 1;

if (a - 1 == 0) {

  y = 1;
```

$(|y = x + 1|)$

```
} else {
```

$(|a = x + 1|)$ If' (φ_2)

```
  y = a;
```

$(|y = x + 1|)$ Atribuição

```
}
```

$(|y = x + 1|)$ If'

Exemplo

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

$(|\top|)$

```
a = x + 1;

if (a - 1 == 0) {
```
$(|1 = x + 1|)$
```
  y = 1;
```
$(|y = x + 1|)$ Atribuição
```
} else {
```
$(|a = x + 1|)$ If' (φ_2)
```
  y = a;
```
$(|y = x + 1|)$ Atribuição
```
}
```
$(|y = x + 1|)$ If'

Exemplo

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

$(|\top|)$

```
a = x + 1;

if (a - 1 == 0) {
```
$(|1 = x + 1|)$ If' (φ_1)
```
  y = 1;
```
$(|y = x + 1|)$ Atribuição
```
} else {
```
$(|a = x + 1|)$ If' (φ_2)
```
  y = a;
```
$(|y = x + 1|)$ Atribuição
```
}
```
$(|y = x + 1|)$ If'

Exemplo

$(|\top|)$

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

```
a = x + 1;
```

$(|((a-1=0) \to (1 = x+1)) \wedge$
$\quad (\neg(a-1=0) \to (a = x+1))|)$

```
if (a - 1 == 0) {
```

$(|1 = x+1|)$ If' (φ_1)

```
  y = 1;
```

$(|y = x+1|)$ Atribuição

```
} else {
```

$(|a = x+1|)$ If' (φ_2)

```
  y = a;
```

$(|y = x+1|)$ Atribuição

```
}
```

$(|y = x+1|)$ If'

Exemplo

$$\frac{(|\varphi_1|)C_1(|\psi|) \qquad (|\varphi_2|)C_2(|\psi|)}{(|(B \to \varphi_1) \land (\neg B \to \varphi_2)|)\texttt{if } B\ \{C_1\}\texttt{ else }\{C_2\}(|\psi|)}$$

$(|\top|)$
$(|((x+1-1=0) \to (1=x+1)) \land$
$(\neg(x+1-1=0) \to (x+1=x+1))|)$
```
a = x + 1;
```
$(|((a-1=0) \to (1=x+1)) \land$
$(\neg(a-1=0) \to (a=x+1))|)$ Atribuição
```
if (a - 1 == 0) {
```
$(|1=x+1|)$ If' (φ_1)
```
  y = 1;
```
$(|y=x+1|)$ Atribuição
```
} else {
```
$(|a=x+1|)$ If' (φ_2)
```
  y = a;
```
$(|y=x+1|)$ Atribuição
```
}
```
$(|y=x+1|)$ If'

Exemplo

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

$(|\top|)$
$(|((x+1-1=0) \to (1 = x+1))\wedge$
$(\neg(x+1-1=0) \to (x+1 = x+1))|)$ Implicação

```
a = x + 1;
```

$(|((a-1=0) \to (1 = x+1))\wedge$
$(\neg(a-1=0) \to (a = x+1))|)$ Atribuição

```
if (a - 1 == 0) {
```

$(|1 = x+1|)$ If' (φ_1)

```
  y = 1;
```

$(|y = x+1|)$ Atribuição

```
} else {
```

$(|a = x+1|)$ If' (φ_2)

```
  y = a;
```

$(|y = x+1|)$ Atribuição

```
}
```

$(|y = x+1|)$ If'

Cálculo para Prova de Correção

While

$$\frac{(\!|\chi \wedge B|\!) C (\!|\chi|\!)}{(\!|\chi|\!) \texttt{while } B \; \{C\} \; (\!|\chi \wedge \neg B|\!)}$$

Cálculo para Prova de Correção

While

$$\frac{(|\chi \wedge B|)C(|\chi|)}{(|\chi|)\texttt{while}\ B\ \{C\}\ (|\chi \wedge \neg B|)}$$

Normalmente queremos provar

$$(|(\varphi)|)\texttt{while}\ B\ \{C\}\ (|\psi|)$$

Cálculo para Prova de Correção

While

$$\frac{(|\chi \wedge B|)C(|\chi|)}{(|\chi|)\texttt{while}\ B\ \{C\}\ (|\chi \wedge \neg B|)}$$

Normalmente queremos provar

$$(|(\varphi)|)\texttt{while}\ B\ \{C\}\ (|\psi|)$$

Achar χ tal que:

1. $\vdash \varphi \rightarrow \chi$

Cálculo para Prova de Correção

While

$$\frac{(|\chi \wedge B|)C(|\chi|)}{(|\chi|)\texttt{while}\ B\ \{C\}\ (|\chi \wedge \neg B|)}$$

Normalmente queremos provar

$$(|(\varphi)|)\texttt{while}\ B\ \{C\}\ (|\psi|)$$

Achar χ tal que:

1. $\vdash \varphi \to \chi$
2. $\vdash \chi \wedge \neg B \to \psi$

Cálculo para Prova de Correção

While

$$\frac{(|\chi \wedge B|)C(|\chi|)}{(|\chi|)\texttt{while}\ B\ \{C\}\ (|\chi \wedge \neg B|)}$$

Normalmente queremos provar

$$(|(\varphi)|)\texttt{while}\ B\ \{C\}\ (|\psi|)$$

Achar χ tal que:

1. $\vdash \varphi \to \chi$
2. $\vdash \chi \wedge \neg B \to \psi$
3. $\vdash_{par} (|(\chi)|)\texttt{while}\ B\ \{C\}\ (|\chi \wedge \neg B|)$

Invariante

Um *invariante* para um laço `while B {C}` é uma fórmula χ tal que:

$$\models_{par} (|\chi \wedge B|)C(|\chi|)$$

Invariante

Um *invariante* para um laço `while B {C}` é uma fórmula χ tal que:

$$\models_{par} (|\chi \wedge B|)C(|\chi|)$$

Obs.: χ pode ser falso **durante** a execução de C.

Invariante

Um *invariante* para um laço `while B {C}` é uma fórmula χ tal que:

$$\models_{par} (|\chi \wedge B|) C (|\chi|)$$

Obs.: χ pode ser falso **durante** a execução de C.

$\top$ e $\perp$ são invariantes para qualquer laço...

Exemplos

Fatorial:

```
y = 1;
z = 0;
while (z != x) {
   z = z + 1;
   y = y * z;
}
```

$$\vdash_{par} (|\top|) \text{ Fatorial } (|y = x!|)$$

Exemplos

Potência:

```
y = 1;
z = 0;
while (z != n) {
   z = z + 1;
   y = y * x;
}
```

$$\vdash_{par} (|\top|)\ \text{Potência}\ (|y = x^n|)$$

Aplicando a regra do while

1. Adivinhar χ

Aplicando a regra do while

1. Adivinhar χ
2. Provar $\vdash \chi \wedge \neg B \rightarrow \psi$ e $\vdash \varphi \rightarrow \chi$ (se falhar, volta para o primeiro passo)

Aplicando a regra do while

1. Adivinhar χ
2. Provar $\vdash \chi \wedge \neg B \rightarrow \psi$ e $\vdash \varphi \rightarrow \chi$ (se falhar, volta para o primeiro passo)
3. Subir χ por C, obtendo χ'

Aplicando a regra do while

1. Adivinhar χ
2. Provar $\vdash \chi \wedge \neg B \rightarrow \psi$ e $\vdash \varphi \rightarrow \chi$ (se falhar, volta para o primeiro passo)
3. Subir χ por C, obtendo χ'
4. Provar que $\vdash \chi \wedge B \rightarrow \chi'$: χ é invariante (se falhar, volta para o primeiro passo)

Aplicando a regra do while

1. Adivinhar χ
2. Provar $\vdash \chi \wedge \neg B \rightarrow \psi$ e $\vdash \varphi \rightarrow \chi$ (se falhar, volta para o primeiro passo)
3. Subir χ por C, obtendo χ'
4. Provar que $\vdash \chi \wedge B \rightarrow \chi'$: χ é invariante (se falhar, volta para o primeiro passo)
5. Colocar χ acima do while e φ acima do χ.

Exemplos

$$(\!|\top|\!)$$

```
y = 1;
z = 0;
while (z != x) {
   z = z + 1;
   y = y * z;
}
```

$$(\!|y = x!|\!)$$

Exemplo: Segmento de Soma Mínima

```
int a[n];
```

Exemplo: Segmento de Soma Mínima

```
int a[n];

a[0], a[1], ..., a[n-1]
```

Exemplo: Segmento de Soma Mínima

`int a[n];`

`a[0], a[1], ..., a[n-1]`

Segmento de a: `a[i], a[i+1], ..., a[j]` com $0 \leq i \leq j < n$

Exemplo: Segmento de Soma Mínima

```
int a[n];
```

`a[0], a[1], ..., a[n-1]`

Segmento de a: `a[i], a[i+1], ..., a[j]` com $0 \leq i \leq j < n$

Segmento de soma mínima:
`[-1, 3, 15, -6, 4, -5]` $\Longrightarrow$ `[-6, 4, -5]`

Exemplo: Segmento de Soma Mínima

```
int a[n];
```

`a[0], a[1], ..., a[n-1]`

Segmento de a: `a[i], a[i+1], ..., a[j]` com $0 \leq i \leq j < n$

Segmento de soma mínima:

`[-1, 3, 15, -6, 4, -5]` $\Longrightarrow$ `[-6, 4, -5]`

`[4, -8, 3, -4, 8, -6, -3, 5]` $\Longrightarrow$ `[-8, 3, -4]` e `[-6, -3]`

Exemplo: Segmento de Soma Mínima

Força bruta: listar todos os segmentos ($O(n^2)$), somar e comparar

Exemplo: Segmento de Soma Mínima

Força bruta: listar todos os segmentos ($O(n^2)$), somar e comparar

Solução: guardar a soma mínima até o ponto (s) e a soma mínima de todos os segmentos que terminam naquele ponto (t).

```
k = 1;
t = a[0];
s = a[0];
while (k != n) {
     t = min(t + a[k], a[k]);
     s = min(s,t);
     k = k + 1;
 }
```

Exemplo: Segmento de Soma Mínima

Força bruta: listar todos os segmentos ($O(n^2)$), somar e comparar

Solução: guardar a soma mínima até o ponto (s) e a soma mínima de todos os segmentos que terminam naquele ponto (t).

```
k = 1;
t = a[0];
s = a[0];
while (k != n) {
     t = min(t + a[k], a[k]);
     s = min(s,t);
     k = k + 1;
 }
```

Funciona sempre???

A função min

```
min:  if (x>y)
        z = y;
      else z = x;
```

$$\vdash_{par} (|\top|)\texttt{min}(|z = min(x, y)|)$$

Exemplo: Segmento de Soma Mínima

$$S_{i,j} = a[i] + a[i+1] + ... + a[j]$$

S1: $(|\top|)$ SomaMin $(|0 \le i \le j < n \to s \le S_{i,j}|)$

Exemplo: Segmento de Soma Mínima

$$S_{i,j} = a[i] + a[i+1] + ... + a[j]$$

S1: $(|\top|)$ SomaMin $(|0 \leq i \leq j < n \rightarrow s \leq S_{i,j}|)$
S2: $(|\top|)$ SomaMin $(|\exists i \exists j(0 \leq i \leq j < n \land s = S_{i,j}|)$

Exemplo: Segmento de Soma Mínima

$$S_{i,j} = a[i] + a[i+1] + ... + a[j]$$

S1: $(|\top|)$ SomaMin $(|0 \leq i \leq j < n \rightarrow s \leq S_{i,j}|)$
S2: $(|\top|)$ SomaMin $(|\exists i \exists j (0 \leq i \leq j < n \wedge s = S_{i,j}|)$

Para provar S1 - encontrar invariante

Exemplo: Segmento de Soma Mínima

$$S_{i,j} = a[i] + a[i+1] + ... + a[j]$$

S1: $(|\top|)$ SomaMin $(|0 \leq i \leq j < n \to s \leq S_{i,j}|)$
S2: $(|\top|)$ SomaMin $(|\exists i \exists j (0 \leq i \leq j < n \land s = S_{i,j}|)$

Para provar S1 - encontrar invariante

$$Inv1(s,k) = \forall i \forall j (0 \leq i \leq j < k \to s \leq S_{i,j})$$

Exemplo: Segmento de Soma Mínima

$$S_{i,j} = a[i] + a[i+1] + \ldots + a[j]$$

S1: $(|\top|)$ SomaMin $(|0 \leq i \leq j < n \rightarrow s \leq S_{i,j}|)$
S2: $(|\top|)$ SomaMin $(|\exists i \exists j (0 \leq i \leq j < n \wedge s = S_{i,j}|)$

Para provar S1 - encontrar invariante

$$Inv1(s,k) = \forall i \forall j (0 \leq i \leq j < k \rightarrow s \leq S_{i,j})$$

$$Inv2(t,k) = \forall i (0 \leq i < k \rightarrow t \leq S_{i,k-1})$$

Exemplo: Segmento de Soma Mínima

$$S_{i,j} = a[i] + a[i+1] + ... + a[j]$$

S1: $(|\top|)$ SomaMin $(|0 \leq i \leq j < n \rightarrow s \leq S_{i,j}|)$
S2: $(|\top|)$ SomaMin $(|\exists i \exists j (0 \leq i \leq j < n \wedge s = S_{i,j}|)$

Para provar S1 - encontrar invariante

$$Inv1(s,k) = \forall i \forall j (0 \leq i \leq j < k \rightarrow s \leq S_{i,j})$$

$$Inv2(t,k) = \forall i (0 \leq i < k \rightarrow t \leq S_{i,k-1})$$

$$Inv1(s,k) \wedge Inv2(t,k)$$

$(\!\vert \top \vert\!)$	
$(\!\vert \mathtt{Inv1}(a[0],1) \wedge \mathtt{Inv2}(a[0],1) \vert\!)$	Implied
`k = 1;`	
$(\!\vert \mathtt{Inv1}(a[0],k) \wedge \mathtt{Inv2}(a[0],k) \vert\!)$	Assignment
`t = a[0];`	
$(\!\vert \mathtt{Inv1}(a[0],k) \wedge \mathtt{Inv2}(t,k) \vert\!)$	Assignment
`s = a[0];`	
$(\!\vert \mathtt{Inv1}(s,k) \wedge \mathtt{Inv2}(t,k) \vert\!)$	Assignment
`while (k != n) {`	
$\quad(\!\vert \mathtt{Inv1}(s,k) \wedge \mathtt{Inv2}(t,k) \wedge k \neq n \vert\!)$	Invariant Hyp. $\wedge$ guard
$\quad(\!\vert \mathtt{Inv1}(\min(s,\min(t+a[k],a[k])),k+1)$	
$\quad\quad\wedge \mathtt{Inv2}(\min(t+a[k],a[k]),k+1) \vert\!)$	Implied `(Lemma 4.20)`
`    t = min(t + a[k], a[k]);`	
$\quad(\!\vert \mathtt{Inv1}(\min(s,t),k+1) \wedge \mathtt{Inv2}(t,k+1) \vert\!)$	Assignment
`    s = min(s,t);`	
$\quad(\!\vert \mathtt{Inv1}(s,k+1) \wedge \mathtt{Inv2}(t,k+1) \vert\!)$	Assignment
`    k = k + 1;`	
$\quad(\!\vert \mathtt{Inv1}(s,k) \wedge \mathtt{Inv2}(t,k) \vert\!)$	Assignment
`}`	
$(\!\vert \mathtt{Inv1}(s,k) \wedge \mathtt{Inv2}(t,k) \wedge \neg\neg(k=n) \vert\!)$	Partial-while
$(\!\vert \mathtt{Inv1}(s,n) \vert\!)$	Implied