霍尔逻辑

霍尔三元组

```
EI ::= N | V | EI + EI | EI - EI | EI * EI
```

```
EB ::= TRUE | FALSE | EI < EI | EB && EB | ! EB
```

```
C ::= SKIP |
      V = EI |
      C; C |
      if (EB) then { C } else { C } |
      while (EB) do { C }
```

$\text{state} \triangleq \text{var\_name} \to \mathbb{Z}$

$[\![\text{EI}]\!] \in \text{state} \to \mathbb{Z}$

$[\![\text{EB}]\!] \in \text{state} \to \text{bool}$

$[\![\text{C}]\!] \subseteq \text{state} \times \text{state}$

```
y = 0;
while (y < 6) do {
  x = x + y;
  y = y + 1
}
```

- P 和 Q 是两条程序状态的性质，c 是一段程序

- P 和 Q 是两条程序状态的性质，c 是一段程序
- 对于任意程序状态 s_1 与 s_2，如果
  - s_1 满足性质 P
  - $(s_1, s_2) \in [\![c]\!]$
- 那么 s_2 满足性质 Q。

- P 和 Q 是两条程序状态的性质，c 是一段程序
- 对于任意程序状态 s_1 与 s_2，如果
  - s_1 满足性质 P
  - $(s_1, s_2) \in [\![c]\!]$
- 那么 s_2 满足性质 Q。
- 写作 $\{P\}c\{Q\}$，称为霍尔三元组，P 称为前条件，Q 称为后条件。

```
{ x == 0 }
y = 0;
while (y < 6) do {
  x = x + y;
  y = y + 1
}
{ x == 1 + 2 + 3 + 4 + 5 }
```

请用中文重述下面霍尔三元组表示的意思。

```
{ True }
c
{ x == 5 }
```

m 是某整数，

```
{ x == m }
c
{ x == m + 5 }
```

对于任意 $0 \leqslant m \leqslant 100$，

```
{ x == m }
c
{ x == m + 5 }
```

请用中文重述下面霍尔三元组表示的意思。

```
{ x <= y }
c
{ y <= x }
```

```
{ True }
c
{ False }
```

请用中文重述下面霍尔三元组表示的意思。

```
{ True }
c
{ y * y <= x && x < (y + 1) * (y + 1) }
```

对于任意 $m \geqslant 0$,

```
{ x == m }
c
{ y * y <= m && m < (y + 1) * (y + 1) }
```

请判断下面霍尔三元组是否成立。

```
{ True }
x = 5
{ x == 5 }
```

```
{ x == 2 }
x = x + 1
{ x == 3 }
```

请判断下面霍尔三元组是否成立。

```
{ True }
x = 5; y = 0
{ x == 5 }
```

```
{ x == 2 && x == 3 }
x = 5
{ x == 0 }
```

请判断下面霍尔三元组是否成立。

```
{ True }
x = x
{ False }
```

请判断下面霍尔三元组是否成立。

```
{ True }
while (1) do { skip }
{ False }
```

顺序执行规则、空语句规则与条件分支语句规则

顺序执行规则

如果 $\{P\}c_1\{Q\}$ 并且 $\{Q\}c_2\{R\}$，那么 $\{P\}c_1; c_2\{R\}$。

例子

```
{ x == m && y == n }
t = x;
x = y;
y = t
{ x == n && y == m }
```

例子

```
{ x == m && y == n }
t = x
{ t == m && y == n }
```

```
{ t == m && y == n }
x = y
{ t == m && x == n }
```

```
{ t == m && x == n }
y = t
{ x == n && y == m }
```

例子

```
{ x == m && y == n }
t = x
{ t == m && y == n }
```

```
{ t == m && y == n }
x = y;
y = t
{ x == n && y == m }
```

例子

```
{ x == m && y == n }
t = x;
x = y;
y = t
{ x == n && y == m }
```

空语句规则

空语句规则

$\{P\}$ skip $\{P\}$

条件分支语句规则

如果 $\{P\}\mathrm{c}_1\{Q\}$ 并且 $\{P\}\mathrm{c}_2\{Q\}$，那么

$$\{P\}\ \text{if}\ (e)\ \text{then}\ \{\ c_1\ \}\ \text{else}\ \{\ c_2\ \}\ \{Q\}$$

```
{ True }
if (x < 0)
then { y = 2 }
else { y = x + 1 }
{ x <= y }
```

```
{ True }
y = 2
{ x <= y }
```

```
{ True }
y = x + 1
{ x <= y }
```

条件分支语句规则

如果 $\{P\&\&e\}\mathrm{c}_1\{Q\}$ 并且 $\{P\&\&!e\}\mathrm{c}_2\{Q\}$，那么

$$\{P\}\ \text{if}\ (e)\ \text{then}\ \{\ c_1\ \}\ \text{else}\ \{\ c_2\ \}\ \{Q\}$$

While 语句规则与循环不变量

While 语句规则

如果 $\{ P \,\&\&e \}\ \mathrm{c}\ \{ P \}$，**那么**

$$\{ P \}\ \text{while}\ (e)\ \text{do}\ \{ c \}\ \{ P \}$$

```
{ x >= 0 }
while (x < 10 && 0 < y) do
{
  x = x + y
}
{ x >= 0 }
```

```
{ x >= 0 && (x < 10 && 0 < y) }
x = x + y
{ x >= 0 }
```

```
{ x == 0 }
while (x < 10) do
{
  x = x + 1
}
{ x == 10 }
```

While 语句规则

如果 $\{\ P\ \&\&e\ \}\ \mathrm{c}\ \{\ P\ \}$，**那么**

$$\{\ P\ \}\ \text{while}\ (e)\ \text{do}\ \{\ c\ \}\ \{\ P\ \&\&\ !e\ \}$$

```
{ x == 0 }
while (x < 10) do
{
  x = x + 1
}
{ x == 10 }
```

```
{ x <= 10 }
while (x < 10) do
{
  x = x + 1
}
{ x <= 10 && ! (x < 10) }
```

```
{ x <= 10 && x < 10 }
  x = x + 1
{ x <= 10 }
```

循环语句规则（例子）

如何证明：

```
{ x >= 0 }
while (0 < x) do {
  x = x - 1
}
{ x == 0 }
```

循环语句规则（例子）

如何证明：

```
{ x >= 0 }
while (0 < x) do {
  x = x - 1
}
{ x == 0 }
```

以 `x >= 0` 为循环不变量。

循环语句规则（例子）

如何证明：

```
{ x >= 0 }
while (0 < x) do {
  x = x - 1
}
{ x == 0 }
```

以 `x >= 0` 为循环不变量。下面霍尔三元组成立

```
{ x >= 0 && x != 0 }
x = x - 1
{ x >= 0 }
```

并且 `x >= 0 && ! (0 < x)` 等价于 `x == 0` 。

循环语句规则（例子）

假如 m 与 n 是给定正整数，如何证明：

```
{ x == m && y == 0 }
while (! (x < n)) do {
  x = x - n;
  y = y + 1
}
{ n * y + x == m && 0 <= x < n }
```

考虑 `n` 的值为 3、 `m` 的值为 10 的具体情况。

```
{ x == 10 && y == 0 }
x = x - 3;
y = y + 1
{ x == 7 && y == 1 }
```

```
{ x == 7 && y == 1 }
x = x - 3;
y = y + 1
{ x == 4 && y == 2 }
```

```
{ x == 4 && y == 2 }
x = x - 3;
y = y + 1
{ x == 1 && y == 3 }
```

循环不变量 P 应当使得下面断言能推出 P:

```
x == 10 && y == 0 ||
x == 7 && y == 1 ||
x == 4 && y == 2 ||
x == 1 && y == 3
```

循环不变量 P 应当使得下面断言能推出 P:

```
x == 10 && y == 0 ||
x == 7 && y == 1 ||
x == 4 && y == 2 ||
x == 1 && y == 3
```

否则以下两个条件总有一个不成立:

- 循环的前条件能推出 P;
- $\{ P \,\&\&$循环条件$\}$ 循环体 $\{ P \}$。

循环不变量 P 也不能太弱，否则

$$P \;\&\&\; !\text{循环条件}$$

不足以推出循环整体的后条件。

取循环不变量 P 为:

取循环不变量 P 为:

```
x == 10 && y == 0 ||
x == 7 && y == 1 ||
x == 4 && y == 2 ||
x == 1 && y == 3
```

待证明的结论

```
{ x == 10 && y == 0 }
while (! (x < 3)) do {
  x = x - 3; y = y + 1
}
{ 3 * y + x == 10 && 0 <= x < 3 }
```

待证明的结论

```
{ x == 10 && y == 0 }
while (! (x < 3)) do {
  x = x - 3; y = y + 1
}
{ 3 * y + x == 10 && 0 <= x < 3 }
```

P 满足的性质

- 循环前条件 `x == 10 && y == 0` 能推出 P
- 循环体能保持循环不变量

```
{ P && ! (x < 3) }
x = x - 3; y = y + 1
{ P }
```

- `P && ! ! (x < 3)` 能推出循环后条件 `3 * y + x == 10 && 0 <= x < 3` 。

另一种循环不变量

```
3 * y + x == 10 && 0 <= x
```

构造循环不变量的一般思路

- 循环的前条件要能推出循环不变量
- 循环不变量不能太强，至少要保证程序运行中每次循环体执行结束后的程序状态应当满足循环不变量，否则循环体不满足霍尔三元组：

$$\{\ P\ \&\&\ e\ \}c\{\ P\ \};$$

- 循环不变量不能太弱，否则 $P\ \&\&\ !e$ 不足以推出循环的后条件；
- 通常情况下，可以考虑选择一个循环不变量，使得满足这个循环不变量的程序状态恰好是所有循环体执行结束之后的程序状态。

循环语句规则（例子）

假如 m 与 n 是给定正整数，如何证明：

```
{ x == m }
while (! (x < n)) do {
  x = x - n
}
{ exists y'. n * y' + x = m && 0 <= x < n }
```

循环语句规则（例子）

假如 m 与 n 是给定正整数，如何证明：

```
{ x == m }
while (! (x < n)) do {
  x = x - n
}
{ exists y'. n * y' + x = m && 0 <= x < n }
```

选择 `exists y', n * y + x == m && x >= 0` 作为循环不变量。

循环语句规则（例子）

假如 m 与 n 是给定正整数，如何证明：

```
{ x == m && y == n }
while (! (! (x < 0) && ! (0 < x))) do {
  y = y - 1;
  x = x - 1
}
{ y == n - m }
```

循环语句规则（例子）

假如 m 与 n 是给定正整数，如何证明：

```
{ x == m && y == n }
while (! (! (x < 0) && ! (0 < x))) do {
  y = y - 1;
  x = x - 1
}
{ y == n - m }
```

选择 `y - x == n - m` 作为循环不变量。

循环语句规则（例子）

假如 m 与 n 是给定正整数，如何证明：

```
{ x == m && y == n }
while (! (! (x < 0) && ! (0 < x))) do {
  y = y - 1;
  x = x - 1
}
{ y == n - m }
```

选择 `y - x == n - m` 作为循环不变量。

或选择 `exists k, y == n - k && x == m - k` 作为循环不变量。

循环语句规则（例子）

假如 m 是给定正整数，如何证明：

```
{ x == m && i == res == 0 }
while (i < x) do {
  res = res + x;
  i = i + 1
}
{ res == m * m }
```

循环语句规则（例子）

假如 m 是给定正整数，如何证明：

```
{ x == m && i == res == 0 }
while (i < x) do {
  res = res + x;
  i = i + 1
}
{ res == m * m }
```

选择 `res == i * m && x == m` 作为循环不变量。

变量赋值语句规则与最强后条件

赋值语句的霍尔三元组（例子）

n 与 m 是给定正整数，对于

- 前条件 `x == n && y == m`
- 程序 `temp = x`

它们的最佳后条件是什么？

赋值语句的霍尔三元组（例子）

n 与 m 是给定正整数，对于

- 前条件 `x == n && y == m`
- 程序 `temp = x`

它们的最佳后条件是什么？答案： `x == n && y == m && temp == n`

如何严格定义最佳后条件？

如何严格定义最佳后条件？

- $\{P\}c\{Q\}$ 成立；
- 如果 $\{P\}c\{Q'\}$ 成立，那么 Q 能推出 Q'；

如何严格定义最佳后条件？

- $\{P\}c\{Q\}$ 成立；
- 如果 $\{P\}c\{Q'\}$ 成立，那么 Q 能推出 Q'；

亦称为最强后条件：strongest postcondition。

赋值语句的霍尔三元组（例子）

对于

- 前条件 `0 <= y`
- 程序 `x = y`

它们的最强后条件是什么？

赋值语句的霍尔三元组（例子）

对于

- 前条件 `0 <= y`
- 程序 `x = y`

它们的最强后条件是什么？答案：`0 <= y && x == y`

赋值语句的霍尔三元组（例子）

对于

- 前条件 `n * y + x == m && 0 <= x && n <= x`
- 程序 `x = x - n`

它们的最强后条件是什么？

赋值语句的霍尔三元组（例子）

对于

- 前条件 `n * y + x == m && 0 <= x && n <= x`
- 程序 `x = x - n`

它们的最强后条件是什么？答案： `n * y + x + n == m && 0 <= x`

赋值语句的霍尔三元组（例子）

对于

- 前条件 `n * y + x + n == m && 0 <= x`
- 程序 `y = y + 1`

它们的最强后条件是什么？

赋值语句的霍尔三元组（例子）

对于

- 前条件 `n * y + x + n == m && 0 <= x`
- 程序 `y = y + 1`

它们的最强后条件是什么？答案：`n * (y - 1) + x + n == m && 0 <= x`

赋值语句的霍尔三元组（例子）

对于

- 前条件 `x == n && y == m`
- 程序 `x = x + y`

它们的最强后条件是什么？

赋值语句的霍尔三元组（例子）

对于

- 前条件 `x == n && y == m`
- 程序 `x = x + y`

它们的最强后条件是什么？答案： `x == n + m && y == m`

赋值语句的霍尔三元组（例子）

对于

- 前条件 `n * y + x == m && 0 <= x < n`
- 程序 `y = 0`

它们的最强后条件是什么？

赋值语句的霍尔三元组（例子）

对于

- 前条件 `n * y + x == m && 0 <= x < n`
- 程序 `y = 0`

它们的最强后条件是什么？答案：

```
exists y', y == 0 && n * y' + x == m && 0 <= x < n
```

变量赋值规则（正向）

那么 $\{P\}\ x = e\ \{\exists x'.\ e[x \mapsto x'] = x\ \&\&\ P[x \mapsto x']\}$

变量赋值规则（正向）

那么 $\{P\}\ x = e\ \{\exists x'.\ e[x \mapsto x'] = x\ \&\&\ P[x \mapsto x']\}$

- $(x \geqslant 0)[x \mapsto x']$ **表示** $x' \geqslant 0$,
- $(\exists k.\ x = a_k)[x \mapsto x']$ **表示** $\exists k.\ x' = a_k$,
- $(x + y)[x \mapsto x']$ **表示** $x' + y$。
- $y[x \mapsto x']$ **表示** y。
- $(x + 1)[x \mapsto x']$ **表示** $x' + 1$。

赋值语句规则（例子）

```
{ x == m && y == n }
x = x + y
{ exists x'. x' + y == x && x' == m && y == n }
```

赋值语句规则（例子）

```
{ x == m && y == n }
temp = x
{ exists temp'. x == temp && x == m && y == n }
```

赋值语句规则（例子）

对于

- 前条件 `x == temp == m && y == n`
- 程序 `x = y`

根据变量赋值语句规则，它们的最强后条件是什么？

赋值语句规则（例子）

对于

- 前条件 `x == temp == m && y == n`
- 程序 `x = y`

根据变量赋值语句规则，它们的最强后条件是什么？答案：

```
exists x'. y == x && x' == temp == m && y == n
```