《软件分析与验证》

# 过程


贺飞

清华大学软件学院

2024 年 4 月 26 日

# 上节课：谓词变换（最弱前置条件）

**最弱前置条件的计算：**

$$
\begin{aligned}
wp(\textbf{skip},\ \varphi) &= \varphi \\
wp(x := a,\ \varphi) &= \varphi[x \mapsto e] \\
wp(a[e_1] := e_2,\ \varphi) &= \varphi[a \mapsto a\langle e_1 \triangleleft e_2 \rangle] \\
wp(st_1; st_2,\ \varphi) &= wp(st_1, wp(st_2, \varphi)) \\
wp(\textbf{if}\ (p)\ \{st_1\}\ \textbf{else}\ \{st_2\},\ \varphi) &= (p \to wp(st_1, \varphi)) \\
&\quad \wedge\ (\neg p \to wp(st_2, \psi)) \\
wp(\textbf{while}\ (p)\{st\},\ \varphi) &= I \qquad (\text{设 } I \text{ 是循环不变式})
\end{aligned}
$$

**额外验证条件的生成：**

$$
VC(\textbf{while}\ (p)\{st\},\ \varphi) = \left\{ \begin{array}{ll} I \wedge \neg p & \to \varphi \\ I \wedge p & \to wp(st, I) \end{array} \right\}
$$

**过程（procedure）**：一段可以被多次调用、完成特定任务的代码块，是结构化程序设计的基本构造。

- 过程通常由一组语句序列组成，可以接受输入参数，执行特定的操作，最终返回执行的结果。
- 可通过函数或者子程序实现。
- 合理利用过程，可以有效提高程序的可读性、可维护性、可重用性和可扩展性。

下面讨论如何在 IMP 语言中扩展过程，以及如何对带有过程的程序进行验证。

# 目录

# 语言扩展

## 定义

IMP 的抽象语法递归定义如下：

$$
\begin{aligned}
e \in AExp &::= c \in \mathbb{Z} \mid x \in Var \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 * e_2 \mid a[e] \\
p \in BExp &::= \textbf{true} \mid \textbf{false} \mid e_1 = e_2 \mid e_1 \leq e_2 \mid \neg p \mid p_1 \wedge p_2 \\
st \in Stmt &::= \textbf{skip} \mid x := e \mid a[e_1] := e_2 \\
&\quad \mid \textbf{assume}\ p \mid \textbf{assert}\ p \\
&\quad \mid st_1; st_2 \\
&\quad \mid \textbf{if}\ (p)\ \{st_1\}\ \textbf{else}\ \{st_2\} \\
&\quad \mid \textbf{while}\ (p)\ \{st\}
\end{aligned}
$$

# 假设语句

假设语句：测试布尔表达式 $p$ 在当前状态下是否成立，只有在 $p$ 成立时才继续后续路径的分析

1. 引入对执行条件的假设，只对满足假设的路径执行分析
2. 不满足假设的路径并非不能执行，只是我们不关心
3. 假设语句不改变程序的状态

假设语句的语义：

$$[\![\textbf{assume } p]\!] = \{(s, s) \mid [\![p]\!]_s = true\}$$

假设语句的最弱前置条件：

$$wp(\textbf{assume } p, \psi) = p \to \psi$$

在下面的分析中，假设语句主要用来处理分支条件；事实上，假设语句可以在程序中直接使用，用来刻画前置条件。

# 断言语句

断言语句：检查布尔表达式 $p$ 在当前状态下是否成立，

- 若成立，程序维持当前状态并继续往下执行；
- 若不成立，程序终止执行，并进入到一个特殊的 $\lightning$ 状态，称为**错误状态**。

断言语句可以用来刻画程序执行到当前位置必须满足的后置条件。

断言语句的语义：

$$[\![\textbf{assert } p]\!] = \{(s, s) \mid [\![p]\!]_s = true\} \cup \{(s, \lightning) \mid [\![p]\!]_s = false\}$$

**定义**

IMP 的抽象语法递归定义如下：

$$
\begin{aligned}
e \in AExp &::= c \in \mathbb{Z} \mid x \in Var \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 * e_2 \mid a[e] \\
p \in BExp &::= \textbf{true} \mid \textbf{false} \mid e_1 = e_2 \mid e_1 \leq e_2 \mid \neg p \mid p_1 \wedge p_2 \\
st \in Stmt &::= \textbf{skip} \mid x := e \mid a[e_1] := e_2 \\
&\quad \mid \textbf{assume}\ p \mid \textbf{assert}\ p \\
&\quad \mid st_1; st_2 \\
&\quad \mid \textbf{if}\ (p)\ \{st_1\}\ \textbf{else}\ \{st_2\} \\
&\quad \mid \textbf{while}\ (p)\ \{st\} \\
&\quad \mid m(e_1, \ldots, e_n)
\end{aligned}
$$

过程调用语句 $m(e_1, \ldots, e_n)$ 中的 $m$ 是过程名，$e_1, \ldots, e_n$ 是实参。

```
/*@ requires 0 <= l && u < len;
    ensures (\result == 1) <==> (\exists integer i;
        l <= i && i <= u && a[i] == e);  */
int LinearSearch(int a[], int len, int l, int u, int e)
```

每个过程都被标注了前置/后置条件对，称为契约（contract）：

- 前置条件用 `requires` 标注；
- 后置条件用 `ensures` 标注；
- 前置和后置条件可以使用过程的形参为变元；
- 后置条件还可以使用一个特殊的变元 `rv`，表示返回值。

# 过程程序的验证

# 过程程序的验证

验证的基本方法：

- 程序可看作为多个过程（包括 `main` 过程）的集合
- 对每个过程独立地执行验证，证明其正确性
  - 从满足前置条件的任意状态进入过程，如果执行能够终止，则在退出过程时一定满足其后置条件
- 过程的执行中可能会调用其他过程
- **程序的正确性 $\Leftrightarrow$ `main` 过程的正确性**

程序**基本路径**（basic path）是满足下列条件的语句序列：

- 开始于过程入口或循环头
- 终止于循环头、断言或过程出口
- 只包含一条分支

霍尔三元组的概念可以被扩展到基本路径：

$$\{\varphi\}\ st_1; st_2; \ldots; st_n\ \{\psi\}$$

其中，$\varphi$ 和 $\psi$ 可以是前置条件、循环不变式、断言或者后置条件。

# 示例：线性搜索



```
/*@ requires 0 <= l && u < len;
    ensures (\result == 1) <==> (\exists integer i;
        l <= i && i <= u && a[i] == e);  */
int LinearSearch(int a[], int len, int l, int u, int e)
{
    int i = l;
    /*@ loop invariant l <= i;
        loop invariant \forall integer j; l<=j<i ==> a[j]!=e; */
    while (i <= u)
    {
        if (a[i] == e) return 1;
        i = i + 1;
    }
    return 0;
}
```

第一条基本路径：

$\{0 \le l \wedge u < len\}$

$i := l$

$\{l \le i \wedge (\forall j.\ l \le j < i \rightarrow a[j] \neq e)\}$

第二条基本路径：

$\{l \le i \wedge (\forall j.\ l \le j < i \rightarrow a[j] \neq e)\}$

**assume** $i \le u$;

**assume** $a[i] = e$;

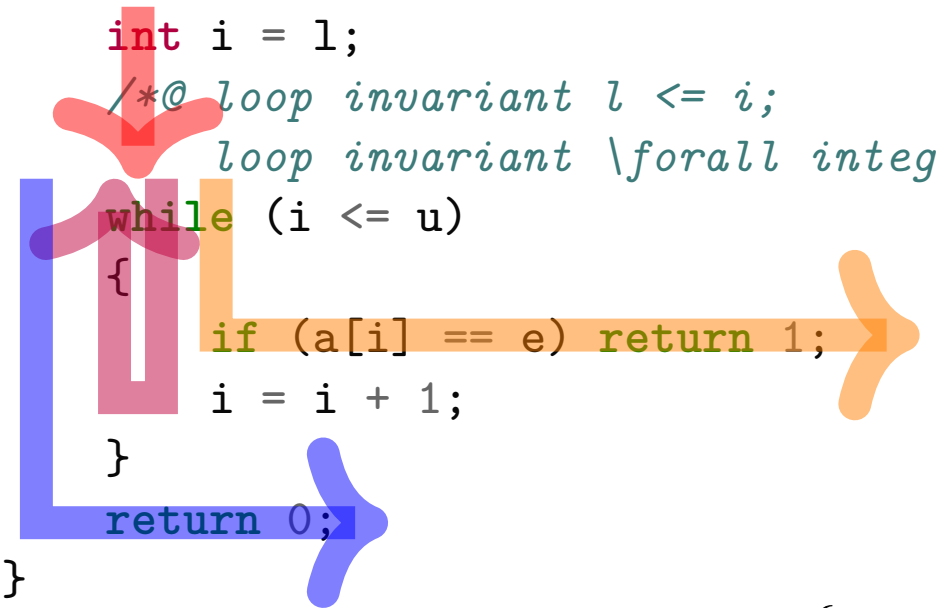
$rv := 1$;

$\{rv = 1 \leftrightarrow (\exists i.\ l \le i \le u \wedge a[i] = e)\}$

# 示例：线性搜索



```
/*@ requires 0 <= l && u < len;
    ensures (\result == 1) <==> (\exists integer i;
        l <= i && i <= u && a[i] == e);  */
int LinearSearch(int a[], int len, int l, int u, int e)
{
    int i = l;
    /*@ loop invariant l <= i;
        loop invariant \forall integer j; l<=j<i ==> a[j]!=e; */
    while (i <= u)
    {
        if (a[i] == e) return 1;
        i = i + 1;
    }
    return 0;
}
```



第四条基本路径：

$\{l \le i \wedge (\forall j.\ l \le j < i \to a[j] \neq e)\}$

**assume** $i > u$;

$rv := 0$;

$\{rv = 1 \leftrightarrow (\exists i.\ l \le i \le u \wedge a[i] = e)\}$

第三条基本路径：

$\{l \le i \wedge (\forall j.\ l \le j < i \to a[j] \neq e)\}$

**assume** $i \le u$;

**assume** $a[i] \neq e$;

$i := i + 1$;

$\{l \le i \wedge (\forall j.\ l \le j < i \to a[j] \neq e)\}$

# 验证条件生成

验证条件生成：

1. 为每条基本路径生成验证条件
2. 如果所有基本路径的验证条件都是有效式，则程序满足规约

**注意**：基本路径中只含两种基本语句：

1. 赋值语句：$wp(x := e, \psi) = \psi[x \mapsto e]$
2. 假设语句：$wp(\textbf{assume}\ p, \psi) = p \to \psi$

# 示例：线性搜索

第一条基本路径：

$$\{0 \le l \land u < len\}$$
$$i := l$$
$$\{l \le i \land (\forall j.\ l \le j < i \to a[j] \neq e)\}$$

验证条件为：

$$0 \le l \land u < len \to wp\,(i := l, l \le i \land (\forall j.\ l \le j < i \to a[j] \neq e))$$

其中：

$$\begin{aligned} & wp\,(i := l, l \le i \land (\forall j.\ l \le j < i \to a[j] \neq e)) \\ & = l \le l \land (\forall j.\ l \le j < l \to a[j] \neq e) \\ & = true \land (\forall j.\ false \to a[j] \neq e) \\ & = true \end{aligned}$$

所以验证条件为：$(0 \le l \land u < len \to true) \Leftrightarrow true$

# 示例：线性搜索

第二条基本路径：

$$
\begin{aligned}
&\{\varphi : l \le i \wedge (\forall j.\ l \le j < i \to a[j] \neq e)\} \\
&st_1 : \textbf{assume}\ i \le u; \\
&st_2 : \textbf{assume}\ a[i] = e; \\
&st_3 : rv := 1; \\
&\{\psi : rv = 1 \leftrightarrow (\exists i.\ l \le i \le u \wedge a[i] = e)\}
\end{aligned}
$$

验证条件为：

$$
\begin{aligned}
& \varphi \to wp(st_1; st_2; st_3, \psi) \\
\Leftrightarrow\ & \varphi \to wp(st_1, wp(st_2; st_3, \psi)) \\
\Leftrightarrow\ & \cdots
\end{aligned}
$$

# 验证条件生成：循环

```
/*@ requires φ
      ensures ψ  */
proc f(···)
{
    while (p){
        st;
    }
}
```

设 $I$ 为循环不变式，则验证条件为：

$$\varphi \to I$$
$$I \wedge p \to wp(st, I)$$
$$I \wedge \neg p \to \psi$$

---

基本路径 1：

$\{\varphi\}$
**skip**;
$\{I\}$

基本路径 2：

$\{I\}$
**assume** $b$;
$c$;
$\{I\}$

基本路径 3：

$\{I\}$
**assume** $\neg b$;
$\{\psi\}$

# 验证条件生成：循环



基本路径 1：

$\{\varphi\}$
**skip**;
$\{I\}$

$\varphi \to I$

基本路径 2：

$\{I\}$
**assume** $b$;
$c$;
$\{I\}$

$I \to wp(\textbf{assume } b, wp(c, I))$
$\Leftrightarrow$
$I \wedge b \to wp(c, I)$

基本路径 3：

$\{I\}$
**assume** $\neg b$;
$\{\psi\}$

$I \to wp(\textbf{assume } \neg b, \psi)$
$\Leftrightarrow$
$I \wedge \neg b \to \psi$

基本路径分析方法与最弱前置条件方法产生的验证条件等价。

# 示例：二分查找



```
/*@ requires 0 <= l && u < len && sorted(a, len, l, u);
    ensures (rv == 1) == \exsits integer i;
        l <= i && i <= u && a[i] == e */
int BinarySearch(int a[], int len, int l, int u, int e)
{
    if (l > u) return 0;
    int m = (l + u) / 2;
    if (a[m] == e) return 1;
    else if (a[m] < e) return BinarySearch(a, len, m + 1, u, e);
    else return BinarySearch(a, len, l, m - 1, e);
}
```

这里的后置条件总结了返回值 rv 和形参之间的关系

# 示例：二分查找



```
/*@ requires 0 <= l && u < len && sorted(a, len, l, u);
    ensures (rv == 1) == \exsits integer i;
        l <= i && i <= u && a[i] == e */
int BinarySearch(int a[], int len, int l, int u, int e)
{
    if (l > u) return 0;
    int m = (l + u) / 2;
    if (a[m] == e) return 1;
    else if (a[m] < e) return BinarySearch(a, len, m + 1, u, e);
    else return BinarySearch(a, len, l, m - 1, e);
}
```

第一条基本路径：

$\{0 \le l \land u < len \land \text{sorted}(a, len, l, u)\}$

**assume** $l > u$;

$rv := 0$;

$\{rv = 1 \leftrightarrow (\exists i.\ l \le i \le u \land a[i] = e)\}$

第二条基本路径：

$\{0 \le l \land u < len \land \text{sorted}(a, len, l, u)\}$

**assume** $l \le u$;

$m := (l + u)/2$;

**assume** $a[m] = e$;

$rv := 1$;

$\{rv = 1 \leftrightarrow (\exists i.\ l \le i \le u \land a[i] = e)\}$

# 示例：二分查找



```
/*@ requires 0 <= l && u < len && sorted(a, len, l, u);
    ensures (rv == 1) == \exsits integer i;
        l <= i && i <= u && a[i] == e */
int BinarySearch(int a[], int len, int l, int u, int e)
{
    if (l > u) return 0;
    int m = (l + u) / 2;
    if (a[m] == e) return 1;
    else if (a[m] < e) return BinarySearch(a, len, m + 1, u, e);
    else return BinarySearch(a, len, l, m - 1, e);
}
```

处理过程调用的基本方法：

- 假定被调用的过程满足契约，证明当前过程的正确性
- 通过其他条件保证被调用的过程确实满足契约

因此：

- 过程被调用时，其前置条件必须被满足
- 被调用的过程返回时，返回值必须满足其后置条件

# 示例：二分查找



```
/*@ requires 0 <= l && u < len && sorted(a, len, l, u);
    ensures (rv == 1) == \exsits integer i;
        l <= i && i <= u && a[i] == e */
int BinarySearch(int a[], int len, int l, int u, int e)
{
    if (l > u) return 0;
    int m = (l + u) / 2;
    if (a[m] == e) return 1;
    else if (a[m] < e) return BinarySearch(a, len, m + 1, u, e);
    else return BinarySearch(a, len, l, m - 1, e);
}
```

第三条基本路径：

$\{0 \le l \wedge u < len \wedge \text{sorted}(a, len, l, u)\}$

**assume** $l \le u$;

$m := (l+u)/2$;

**assume** $a[m] \ne e$;

**assume** $a[m] < e$;

$\{0 \le m+1 \wedge u < len \wedge \text{sorted}(a, len, m+1, u)\}$

第四条基本路径：

$\{0 \le l \wedge u < len \wedge \text{sorted}(a, len, l, u)\}$

**assume** $l \le u$;

$m := (l+u)/2$;

**assume** $a[m] \ne e$;

**assume** $a[m] < e$;

**assume** $v_1 = 1 \leftrightarrow \exists i.(m+1 \le i \le u \wedge a[i] = e)$

$rv := v_1$

$\{rv = 1 \leftrightarrow \exists i.(l \le i \le u \wedge a[i] = e)\}$

# 示例：二分查找



```
/*@ requires 0 <= l && u < len && sorted(a, len, l, u);
    ensures (rv == 1) == \exsits integer i;
        l <= i && i <= u && a[i] == e */
int BinarySearch(int a[], int len, int l, int u, int e)
{
    if (l > u) return 0;
    int m = (l + u) / 2;
    if (a[m] == e) return 1;
    else if (a[m] < e) return BinarySearch(a, len, m + 1, u, e);
    else return BinarySearch(a, len, l, m - 1, e);
}
```

第五条基本路径：

**练习**。。。

第六条基本路径：

$\{0 \le l \wedge u < len \wedge \text{sorted}(a, len, l, u)\}$

**assume** $l \le u$;

$m := (l + u)/2$;

**assume** $a[m] \ne e$;

**assume** $a[m] \ge e$;

**assume** $v_2 = 1 \leftrightarrow \exists i.(l \le i \le m - 1 \wedge a[i] = e)$

$rv := v_2$

$\{rv = 1 \leftrightarrow \exists i.(l \le i \le u \wedge a[i] = e)\}$

# 总结：过程



IMP 语言扩展

- 断言语句、假设语句、过程调用语句
- 过程契约

过程程序验证的基本方法

- 将程序分解为若干基本路径
- 基本路径中只含赋值语句和假设语句
- 分别为每条基本路径生成验证条件
- 如果所有基本路径的验证条件都是有效式，则程序满足规约

# 下节课内容



- 程序终止性证明

# 谢谢!