《软件分析与验证》

最弱前置条件

贺飞

清华大学软件学院

2024 年 4 月 19 日

上节课：数组

数组理论

- 操作：数组读，数组写
- 公理：写后读 1&2，数组同余，数组扩展

扩展数组

- 语法扩展、语义解释、霍尔推理规则
- 数组越界问题

程序验证过程的机械化

相比于直接基于程序语义的验证，霍尔逻辑提供了一套从语法上证明程序是否满足规约的验证方法。

霍尔证明系统给出了一系列推理规则。如何应用这些规则，往往需要人工参与：

- 何时应用前提加强或结论弱化规则?
- 如果需要应用，前提（或结论）该加强（或弱化）到什么程度?
- 如何检查这两条规则的条件（类似于 $\varphi \to \psi$ 的公式)?

程序验证过程机械化——朝自动程序验证方向迈出的重要一步!

- 1930 年 5 月 11 日 – 2002 年 8 月 6 日
- 荷兰计算机科学家
- 1972 年图灵奖得主
- 重要贡献：
    - 最短路径算法（Dijkstra's Algorithm）
    - 结构化程序设计语言
    - 谓词变换

From: https://en.wikipedia.org/wiki/Edsger_W._Dijkstra

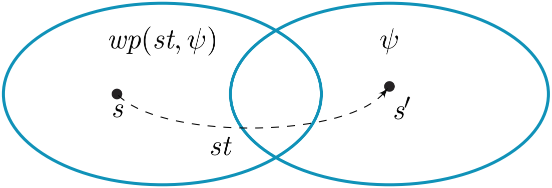
谓词变换（Predicate Transformer）：给定一个谓词表达式和一条程序语句，通过变换得到另一个谓词表达式。

主要的谓词变换有：

- 最弱前置条件（Weakest Precondition）[1] $wp(st, \psi)$：假定语句执行后满足 ψ，在执行这条语句之前必须满足的条件：
  1. $\{wp(st, \psi)\}\ st\ \{\psi\}$ 是有效式，即 $wp(st, \psi)$ 是一个能够让 ψ 成立的前置条件，以及
  2. 若存在 φ 使得 $\{\varphi\}\ st\ \{\psi\}$ 是有效式，则 $\varphi \Rightarrow wp(st, \psi)$，即 $wp(st, \psi)$ 是所有此类前置条件中最弱的一个条件。
- 最强后置条件（Strongest Postcondition）$sp(\varphi, st)$：假定语句执行前满足 φ，在执行这条语句之后必定满足的条件：
  1. $\{\varphi\}\ st\ \{sp(\varphi, st)\}$ 是有效式，以及
  2. 若存在 ψ 使得 $\{\varphi\}\ st\ \{\psi\}$ 是有效式，则 $sp(\varphi, st) \Rightarrow \psi$，即 $sp(\varphi, st)$ 是所有此类后置条件中最强的一个条件。

[1] 本课程中讨论的“最弱前置条件”实际上是指“最弱自由前置条件（weakest liberal precondition）”，即假定程序终止的情况下的“最弱前置条件”。

最弱前置条件



对一阶逻辑公式 ψ 和程序语句 st，最弱前置条件 $wp(st, \psi)$ 是满足以下条件的一阶逻辑公式：

- $\forall s \models wp(st, \psi)$，从 s 出发执行语句 st，如果执行能够终止，则终止后的状态 $s' \models \psi$;
- $\forall s \not\models wp(st, \psi)$，从 s 出发执行语句 st，如果执行能够终止，则执行后的状态 $s' \not\models \psi$。

给定程序 st 和规约 (φ, ψ)，验证条件 (verification conditions) 是一组满足下面条件的一阶逻辑公式：

如果所有验证条件都是有效式，则程序满足规约

最弱前置条件提供了一种生成验证条件的方法。

最弱前置条件：空语句

空语句的推理规则:

$$(\text{空语句})\ \frac{}{\{\varphi\}\ \textbf{skip}\ \{\varphi\}}$$

空语句最弱前置条件的计算公式:

$$wp(\textbf{skip}, \varphi) = \varphi$$

赋值语句的推理规则:

$$(\text{赋值})\ \frac{}{\{\varphi[x \mapsto e]\}\ x := e\ \{\varphi\}}$$

赋值语句最弱前置条件的计算公式:

$$wp(x := e, \varphi) = \varphi[x \mapsto e]$$

根据赋值语句推理规则的可靠性，$\varphi[x \mapsto e]$ 是有效的前置条件。

赋值语句示例

例

计算 $wp(y := x+1, (\forall x.\ x < z \to x < y) \to x+1 \leq y) = ?$

解

- 将公式中的 y 直接替换为 $x+1$，得到：

$$(\forall x.\ x < z \to x < x+1) \to x+1 \leq x+1$$

  对吗？

- 错误！ 在以 $x+1$ 代换下式中的 y 时，

$$(\forall x.\ x < z \to x < y)$$

  代换式中的 x 被原式中的约束变元 x 同名，被原式的量词 $\forall x$ 捕获了。

赋值语句示例（续）

解 (续)

- 原公式的约束变元 x 在代换项中出现，重命名为 x'，得到

$$(\forall x'.\ x' < z \to x' < y) \to x + 1 \le y)$$

  注意 x 的最后一次出现是自由出现，不需要重命名。

- 对重命名后的公式执行 $y \mapsto x + 1$ 代换，得到：

$$(\forall x'.\ x' < z \to x' < x + 1) \to x + 1 \le x + 1)$$

数组赋值规则的推理规则：

$$\text{（数组赋值）}\ \frac{}{\{\varphi[a \mapsto a\langle e_1 \triangleleft e_2\rangle]\}\ \ a[e_1] := e_2\ \ \{\varphi\}}$$

对应最弱前置条件的计算公式：

$$wp(a[e_1] := e_2, \varphi) = \varphi[a \mapsto a\langle e_1 \triangleleft e_2\rangle]$$

例

计算 $wp(b[i] := 5, b[i] = 5)$

解

$$
\begin{aligned}
& wp(b[i] := 5, b[i] = 5) \\
& = (b[i] = 5)[b \mapsto b\langle i \triangleleft 5\rangle] \\
& = (b\langle i \triangleleft 5\rangle[i] = 5) \\
& = (5 = 5) \\
& = \top
\end{aligned}
$$

数组赋值语句示例 2

例

计算 $wp(b[n] := x,\ \forall i.\ 1 \le i < n \to b[i] \le b[i+1])$

解

$$
\begin{aligned}
& wp(b[n] := x,\ \forall i.\ 1 \le i < n \to b[i] \le b[i+1]) \\
& = (\forall i.\ 1 \le i < n \to b[i] \le b[i+1])\ [b \mapsto b\langle n \lhd x\rangle] \\
& = (\forall i.\ 1 \le i < n \to b\langle n \lhd x\rangle[i] \le b\langle n \lhd x\rangle[i+1]) \\
& = b\langle n \lhd x\rangle[n-1] \le b\langle n \lhd x\rangle[n] \\
& \qquad\quad \wedge\ (\forall i.\ 1 \le i < n-1 \to b\langle n \lhd x\rangle[i] \le b\langle n \lhd x\rangle[i+1]) \\
& = b[n-1] \le x\ \wedge\ (\forall i.\ 1 \le i < n-1 \to b[i] \le b[i+1])
\end{aligned}
$$

顺序组合语句的推理规则：

$$(\text{顺序})\ \frac{\{\varphi_1\}\ st_1\ \{\varphi_2\} \qquad \{\varphi_2\}\ st_2\ \{\varphi_3\}}{\{\varphi_1\}\ st_1; st_2\ \{\varphi_3\}}$$

对应最弱前置条件的计算公式：

$$wp(st_1; st_2, \varphi_3) = wp(st_1, wp(st_2, \varphi_3))$$

- 从 st_2 开始，由后向前计算
- 将 st_2 的最弱前置条件处理为 st_1 的后置条件，然后继续计算
- 注意：不需要应用前提加强或者结论弱化规则

例

计算 $wp(t := x; x := y; y := t, y = x' \wedge x = y')$

解

$$
\begin{aligned}
& wp(t := x; x := y; y := t, y = x' \wedge x = y') \\
& = wp(t := x, wp(x := y, wp(y := t, y = x' \wedge x = y'))) \\
& = wp(t := x, wp(x := y, t = x' \wedge x = y')) \\
& = wp(t := x, t = x' \wedge y = y') \\
& = (x = x' \wedge y = y')
\end{aligned}
$$

分支语句的推理规则：

$$(\text{分支})\ \frac{\{\varphi \wedge p\}\ \ st_1\ \ \{\psi\} \qquad \{\varphi \wedge \neg p\}\ \ st_2\ \ \{\psi\}}{\{\varphi\}\ \ \mathbf{if}\ (p)\ \{st_1\}\ \mathbf{else}\ \{st_2\}\ \ \{\psi\}}$$

对应最弱前置条件的计算公式：

$$\begin{aligned} & wp(\mathbf{if}\ (p)\ \{st_1\}\ \mathbf{else}\ \{st_2\},\ \psi) \\ & \qquad\qquad = (p \to wp(st_1, \psi))\ \wedge\ (\neg p \to wp(st_2, \psi)) \end{aligned}$$

- 如果 p 成立，则 st_1 分支的最弱前置条件必须成立
- 否则，st_2 分支的最弱前置条件必须成立

分支语句示例

例

记 $\psi = (x \geq 0 \to y = x - 1) \wedge (x < 0 \to y = x + 1)$，

计算 $wp(\textbf{if } (y \geq 0)\{\texttt{y:=y-1}\} \textbf{ else } \{\texttt{y:=y+1}\}, \psi)$

解

$$
\begin{aligned}
& wp(\textbf{if } (y \geq 0)\{\textit{y:=y-1}\} \textbf{ else } \{\textit{y:=y+1}\}, \psi) \\
& = (y \geq 0 \to wp(\textit{y:=y-1}, \psi) \wedge (y < 0 \to wp(\textit{y:=y+1}, \psi))) \\
& = (y \geq 0 \to ((x \geq 0 \to y - 1 = x - 1) \wedge (x < 0 \to y - 1 = x + 1))) \\
& \ \wedge (y < 0 \to ((x \geq 0 \to y + 1 = x - 1) \wedge (x < 0 \to y + 1 = x + 1)))
\end{aligned}
$$

注意：这里每一步计算的结果都是确定的，无需应用前提加强和结论弱化规则。

回顾关于循环语句的等价关系

$$\textbf{while } (p)\{st\} \;\equiv\; \textbf{if } (p)\{st; \textbf{while } (p)\{st\}\} \textbf{ else skip}$$

以循环展开的方式计算循环语句的最弱前置条件：

$$
\begin{aligned}
& wp(\underline{\textbf{while } (p)\{st\}},\ \varphi) \\
&= wp(\underline{\textbf{if } (p)\{st; \textbf{while } (p)\{st\}\} \textbf{ else skip}},\ \varphi) \\
&= (p \to wp(st; \underline{\textbf{while } (p)\{st\}},\ \varphi)) \wedge (\neg p \to wp(\textbf{skip},\ \varphi)) \\
&= (p \to wp(st,\ wp(\underline{\textbf{while } (p)\{st\}},\ \varphi))) \wedge (\neg p \to \varphi) \\
&= (p \to wp(st,\ wp(\underline{\textbf{if } (p)\{st; \textbf{while } (p)\{st\}\} \textbf{ else skip}},\ \varphi))) \wedge (\neg p \to \varphi)
\end{aligned}
$$

未能得出任何有用的结论！

循环语句的最弱前置条件总是存在的 [2]，但不一定能被计算出来。

设 I 是 **while** (p) $\{st\}$ 的循环不变式，以 I 作为循环语句的近似最弱前置条件：

- 需要确保 I 必须是有效的前置条件
- 无法确保 I 一定是最弱的前置条件

[2] The Formal Semantics of Programming Languages: An Introduction (Section 7), Glynn Winiskel, 1993

如何确保 I 是有效的前置条件？

回顾循环语句的推理规则：

$$\text{（循环）}\ \frac{\{I \wedge p\}\ \ st\ \ \{I\}}{\{I\}\ \ \textbf{while}\ (p)\{st\}\ \ \{I \wedge \neg p\}}$$

设 ψ 为后置条件，$\{I\}\ \ \textbf{while}\ (p)\{st\}\ \ \{\psi\}$ 是有效式的条件为

- $\{I \wedge p\}\ \ st\ \ \{I\}$ 是有效式，即 $I \wedge p \Rightarrow wp(st, I)$
- $I \wedge \neg p \Rightarrow \psi$

记作

$$VC(\textbf{while}\ (p)\{st\},\ \psi) = \left\{ \begin{array}{c} I \wedge \neg p \rightarrow \psi \\ I \wedge p \rightarrow wp(st, I) \end{array} \right\}$$

虽然 **while** 是唯一引入额外验证条件的语句，为了更方便的生成整个程序的验证条件，有必要将 VC 的定义扩展到所有程序语句：

- $VC(x := a, \psi) = \varnothing$
- $VC(st_1; st_2, \psi) = VC(st_1, wp(st_2, \psi)) \cup VC(st_2, \psi)$
- $VC(\textbf{if } (p)\ \{st_1\} \textbf{ else } \{st_2\}, \psi) = VC(st_1, \psi) \cup VC(st_2, \psi)$
- $VC(\textbf{while } (p)\{st\}, \psi) = \left\{ \begin{array}{c} I \wedge \neg p \to \psi \\ I \wedge p \to wp(st, I) \end{array} \right\}$，其中 I 是循环不变式。

$$st \models (\varphi, \psi)?$$

1. 计算 $\varphi' = wp(st, \psi)$
2. 计算 $VC(st, \psi)$
3. 检查 $\varphi \to \varphi'$ 是否有效式
4. 检查 $VC(st, \psi)$ 中的每一个公式是否有效式

$VC(st, \psi)$ 和 $\varphi \to \varphi'$ 统称为验证条件（Verification Conditions）。

思考：

- 如果所有验证条件的检查都通过，则 $st \models (\varphi, \psi)$? 是
- 若 $st \models (\varphi, \psi)$，则一定能够通过所有验证条件的检查? 否

循环不变式是近似最弱前置条件

程序验证器一般由两部分组成：

- 验证器：可以理解为验证条件生成器，将程序验证问题编码为 SMT 公式；
- 求解器：实现了许多一阶理论的判定算法，支持 SMT 公式的可满足性判定。

程序验证示例

```
r = x;
q = 0;
while (y <= r){
  r = r-y;
  q = q + 1;
}
```

- 前置条件

$$\varphi : true$$

- 后置条件

$$\psi : r < y \land x = r + (q \times y)$$

- 设已经找到一个候选循环不变式

$$I : x = r + (q \times y)$$

$\{x = x + (0 \times y)\}$

```
r = x;
```

$\{x = r + (0 \times y)\}$

```
q = 0;
```

$\{x = r + (q \times y)\}$

```
while (y <= r){
  r = r-y;
  q = q + 1;
}
```

$\{r < y \wedge x = r + (q \times y)\}$

验证条件：

- $I \wedge \neg(y \leq r) \rightarrow \psi$
- $I \wedge y \leq r \rightarrow wp(r := r - y; q := q + 1,\ I)$
- $true \rightarrow (x = x + (0 \times y))$

$$
\begin{aligned}
& wp(r := r - y; q := q + 1,\ I) \\
& = wp(r := r - y, wp(q := q + 1,\ I)) \\
& = wp(r := r - y,\ x = r + ((q + 1) \times y)) \\
& = (x = (r - y) + ((q + 1) \times y)) \\
& = (x = r + (q \times y))
\end{aligned}
$$

程序验证示例

$\{x = x + (0 \times y)\}$

```
r = x;
```

$\{x = r + (0 \times y)\}$

```
q = 0;
```

$\{x = r + (q \times y)\}$

```
while (y <= r){
  r = r-y;
  q = q + 1;
}
```

$\{r < y \wedge x = r + (q \times y)\}$

验证条件:

- $I \wedge \neg(y \leq r) \rightarrow \psi$
- $I \wedge y \leq r \rightarrow (x = r + q \times y)$
- $true \rightarrow (x = x + (0 \times y))$

程序验证示例

$\{x = x + (0 \times y)\}$

```
r = x;
```

$\{x = r + (0 \times y)\}$

```
q = 0;
```

$\{x = r + (q \times y)\}$

```
while (y <= r){
  r = r-y;
  q = q + 1;
}
```

$\{r < y \land x = r + (q \times y)\}$

验证条件：

- $I \land \neg(y \leq r) \rightarrow \psi$
- $I \land y \leq r \rightarrow (x = r + q \times y)$
- $true \rightarrow (x = x + (0 \times y))$

验证条件的所有公式都是有效式。
所以示例程序满足给定的规约。

最弱前置条件的计算：

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

额外验证条件的生成：

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

下节课内容

- 为 IMP 语言添加过程调用

谢谢!