《软件分析与验证》

数组

贺飞
清华大学软件学院

2024 年 4 月 19 日

IMP 程序规约:

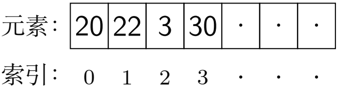
- 霍尔三元组
- 部分正确性、完全正确性

IMP 霍尔证明系统:

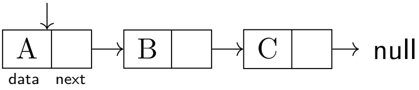
- 证明规则
- 循环和循环不变式
- 可靠、相对完备

数据结构

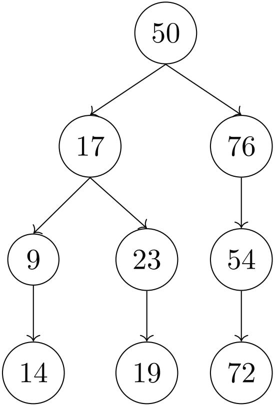
如何验证带数据结构（如数组、列表、树等）的程序?



(a) 数组



(b) 列表



(c) 树

图: 常见数据结构

如何对数据结构进行推理?

- 哥德尔编号 (Gödel numbering)[1]: 为数据结构的每个实例指派一个唯一的自然数 (称哥德尔数), 转化为整数理论问题。
- 数据结构的公理化系统: 在一阶逻辑基础上, 引入特殊符号表示数据结构的操作, 并以公理刻画这些操作的含义
  - **一阶理论!**

本节课的内容:

- 数组理论
- 在 IMP 语言中扩展数组
- 在霍尔证明系统中增加对数组的支持

[1] https://en.wikipedia.org/wiki/Gödel_numbering

目录

数组理论

元素:	20	22	3	30	·	·	·
索引:	0	1	2	3	·	·	·

什么是数组?

- 从数组索引到数组元素的映射函数
- 简单起见,假设数组索引和数组元素都是整数
- 数组的定义 $a : \mathbb{Z} \to \mathbb{Z}$
- 考虑左边的示例:

$$a(x) = \begin{cases} 20, & \text{如果 } x = 0 \\ 22, & \text{如果 } x = 1 \\ 3, & \text{如果 } x = 2 \\ \ldots \end{cases}$$

数组

元素:	20	22	3	30	·	·	·
索引:	0	1	2	3	·	·	·

$\Downarrow$

元素:	20	**21**	3	30	·	·	·
索引:	0	1	2	3	·	·	·

有哪些数组操作?

- 数组读 `a[0]`——读取数组 a 第 0 个位置的值
- 数组写 `a[1] = a[0] + 1`

数组写导致数组某个位置的值发生改变。将数组视作映射函数，则该函数发生了改变。

- 以上面的数组写为例，执行后 a 更新为 a'。
- a' 与 a 相比只有第 1 个位置发生了改变，其他位置上的值不变，即

$$a'(x) = \begin{cases} 21, & \text{如果 } x = 1 \\ a(x), & \text{否则} \end{cases}$$

回顾：一阶理论

定义

一阶理论（*first-order theory*） $\mathcal{T}$ 可表示为二元组 $(\Sigma, \mathcal{A})$，其中：

- Σ 是一个非逻辑符号集，称为签名（*signature*）；
- $\mathcal{A}$ 是一组定义在 Σ 上的闭公式，称为公理集（*axiom*）。

一阶理论是一阶逻辑的受限形式，其中：

- Σ 对理论中允许出现的非逻辑符号进行限定（注意一阶逻辑允许任意非逻辑符号）
- $\mathcal{A}$ 规定了这些非逻辑符号的含义

签名 Σ_A：$\{=, \cdot[\cdot], \cdot\langle\cdot \triangleleft \cdot\rangle\}$，其中

- $a[i]$ 是二元函数，表示读取数组 a 的第 i 个元素；
- $a\langle i \triangleleft v\rangle$ 是三元函数，表示将数组 a 的第 i 个元素更新为 v。

公理集 $\mathcal{A}_A$ 赋予两个数组操作 $a[i]$ 和 $a\langle i \triangleleft v\rangle$ 以含义

1. 等式理论中关于等号的自反、对称、传递公理
2. **数组同余：** $\forall a, i, j.\ i = j \to a[i] = a[j]$
3. **写后读 1：** $\forall a, v, i, j.\ i = j \to a\langle i \triangleleft v\rangle[j] = v$
4. **写后读 2：** $\forall a, v, i, j.\ i \neq j \to a\langle i \triangleleft v\rangle[j] = a[j]$

例

考虑下列 $\mathcal{T}_A$ 项，并解释其含义：

- $a\langle i \lhd v\rangle$
  代表一个数组。该数组在 i 位置上的值为 v，在其他位置上的值与 a 都相同。
- $a\langle i \lhd v\rangle[j]$
  以 a' 代表 $a\langle i \lhd v\rangle$，则原式等价于 $a'[j]$
- $a\langle i \lhd v\rangle\langle j \lhd w\rangle$
  代表一个数组。该数组在位置 i 和 j 上的值分别为 v 和 w，在其他位置上的值与 a 都相同。
- $a\langle i \lhd v\rangle\langle j \lhd w\rangle[k]$
  以 a'' 代表 $a\langle i \lhd v\rangle\langle j \lhd w\rangle$，则原式等价于 $a''[k]$

注意：$\mathcal{T}_A$ 理论只定义了数组元素之间的等式，没有定义数组之间的等式。因此

$$a[i] = v \to a\langle i \triangleleft v\rangle = a$$

不是 $\mathcal{T}_A$ 的有效式。而应该表达为

$$a[i] = v \to \forall j.\ a\langle i \triangleleft v\rangle[j] = a[j]$$

同理，

$$a = b \to a[i] = b[i]$$

也不是 $\mathcal{T}_A$ 的有效式。

这为数组程序的规约和验证带来不便。

在 $\mathcal{T}_A$ 的基础上，增加一条公理：

$$\textbf{扩展：} \forall a, b.\ (\forall i.\ a[i] = b[i]) \leftrightarrow a = b$$

扩展后的数组理论记作 $\mathcal{T}_A^=$。

扩展后，

$$a[i] = v \to a\langle i \triangleleft v\rangle = a$$
$$a = b \to a[i] = b[i]$$

都是 $\mathcal{T}_A^=$ 的有效式。

数组理论的讨论对象除了数组索引和数组元素外，还有数组本身。因此，当用变元符号指代数组理论的讨论对象时，可以是数组索引和数组元素（整数类型 $\mathbb{Z}$），也可以是数组（函数类型 $\mathbb{Z} \to \mathbb{Z}$）。

将数组理论的变元符号分成整数变元和数组变元，约定

- 以 a, b, c 代表数组，i, j 代表数组索引，x, y, z 代表数组元素

状态是对程序所有变元（包括整数变元和数组变元）的赋值，即

$$State: \; Var \to \mathbb{Z} \cup (\mathbb{Z} \to \mathbb{Z})$$

给定状态 s，$s(x)$ 返回整数变元 x 在状态 s 上的赋值，$s(a)$ 返回数组变元 a 在状态 s 上的实例，是如下所示的一个函数：

元素：	20	22	3	30	·	·	·
索引：	0	1	2	3	·	·	·

示例

例

以 $|\cdot|$ 表示数组的长度。

- 变量 x 记录了数组 a 中的最大元素：
  $(\forall i.\ 0 \le i < |a| \to a[i] \le x) \land (\exists i.\ 0 \le i < |a| \land a[i] = x)$
- 数组 a 中的元素按从小到大排序：
  $\forall i.\ 0 \le i < |a| - 1 \to a[i] \le a[i+1]$
- 数组 a 中不含等于 0 的元素：
  $\forall i.\ 0 \le i < |a| \to a[i] \ne 0$

数组理论的性质

- $\mathcal{T}_A$ 和 $\mathcal{T}_A^=$ 都是不可判定的
- $\mathcal{T}_A$ 和 $\mathcal{T}_A^=$ 的无量词片段都是可判定的

扩展数组

定义

IMP 的抽象语法递归定义如下：

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

添加了对数组读和写的支持，其中数组下标可以是任意算术表达式。

数组读表达式 $a[e]$ 在状态 s 下的语义:

$$[\![a[e]]\!]_s^A = s(a)([\![e]\!]_s^A)$$

其中 $s(a)$ 是数组 a 在状态 s 下的实例; $[\![e]\!]_s^A$ 是一个整数，被用作数组索引。

数组赋值语句的语义:

$$[\![a[e_1] := e_2]\!] = \{(s, s') \mid s' = s[a \mapsto a\langle [\![e_1]\!]_s^A \triangleleft [\![e_2]\!]_s^A \rangle]\}$$

其中 $[\![e_1]\!]_s^A$ 和 $[\![e_2]\!]_s^A$ 都是整数，上面公式的含义是后状态 s' 相比于前状态 s，将数组 a 修改为 $a\langle [\![e_1]\!]_s^A \triangleleft [\![e_2]\!]_s^A \rangle$。

例

给定含整数变元 i, x 和数组变元 a 的程序 P；设当前状态为 s，并且在 s 下 $i \mapsto 1, x \mapsto 100$，数组 a 的所有值都为 0，

- $[\![a[i+1]]\!]_s^A \;= s(a)([\![i+1]\!]_s^A) = s(a)(2) = 0$
- $[\![a[i+1] := x+2]\!] \;= \{(s, s') \mid s' = s[a \mapsto a\langle 2 \triangleleft 102\rangle]\}$
- $post(\{s\}, [\![a[i+1] := x+2]\!])$

$$= post(\{s\}, \{(s, s') \mid s' = s[a \mapsto a\langle 2 \triangleleft 102\rangle]\})$$
$$= \{i = 1 \wedge x = 100 \wedge a = s(a)\langle 2 \triangleleft 102\rangle\}$$

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

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

引理 (数组赋值语句推理规则的可靠性)

霍尔三元组 $\{\varphi[a \mapsto a\langle e_1 \lhd e_2\rangle]\}\ a[e_1] := e_2\ \{\varphi\}$ 是有效式。

证明.

与赋值语句推理规则可靠性的证明类似。 □

如何处理数组越界？

在每次数组访问（读或者写）之前，执行 $0 \leq i < |a|$ 的检查，即

$$(\text{赋值 }*)\ \frac{}{\{0 \leq e \wedge e < |a| \wedge \varphi[x \mapsto a[e]]\}\ x := a[e]\ \{\varphi\}}$$

$$(\text{数组赋值 }*)\ \frac{}{\{0 \leq e_1 \wedge e_1 < |a| \wedge \varphi[a \mapsto a\langle e_1 \triangleleft e_2\rangle]\}\ a[e_1] := e_2\ \{\varphi\}}$$

数组越界问题只跟数组的大小有关，跟数组中的元素无关；建模数组越界问题并不一定需要用到数组理论。

示例：数组越界问题


```
while (i < N)
{
  a[i] := a[i+1];
  i := i + 1;
}
```

设 $|a| = N$，示例程序是否存在数组越界问题可以用下式来编码：

$$i < N \rightarrow (0 \leq i \wedge i < N \wedge 0 \leq i + 1 \wedge i + 1 < N)$$

总结：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

下节课内容

- 谓词变换

谢谢！