软件分析与验证 **截止时间: 2023 年 4 月 6 日**

作业 2

授课老师: 贺飞 你的姓名 (你的学号)
助教: 韩志磊、徐志杰、谢兴宇

在开始完成作业前，请仔细阅读以下说明:

- 我们提供作业的 LaTeX 源码，你可以在其中直接填充你的答案并编译 PDF（请使用 xelatex）。当然，你也可以使用别的方式完成作业（例如撰写纸质作业后扫描到 PDF 文件之中）。但是请注意，最终的提交一定只是 PDF 文件。提交时请务必再次核对，防止提交错误。
- 在你的作业中，请务必填写你的姓名和学号，并检查是否有题目遗漏。请重点注意每次作业的截止时间。截止时间之后你仍可以联系助教补交作业，但是我们会按照如下公式进行分数的折扣:

$$\text{作业分数} = \text{满分} \times (1 - 10\% \times \min(\lceil \text{迟交周数} \rceil, 10)) \times \text{正确率}.$$

- 本次作业为独立作业，禁止抄袭等一切不诚信行为。作业中，如果涉及参考资料，请引用注明。

Problem 1: 一阶理论

在一阶理论中，借助公理，我们可以证明很多有趣的有效式。例如，当我们考虑 Peano 算术 $\mathcal{T}_{PA}$ 时，$\forall x.\ 0 + x = x$ 是一个 $\mathcal{T}_{PA}$-有效式，证明如下:

Peano 算术为我们提供了归纳公理 $(F[0] \wedge \forall x.\ (F[x] \rightarrow F[x+1])) \rightarrow \forall x.\ F[x]$，我们可以利用它来证明 $\forall x.\ 0 + x = x$ 是 $\mathcal{T}_{PA}$-有效的。为了利用归纳公理，我们需要实例化 F，这里我们定义 $F[x] \triangleq 0 + x = 0$。观察归纳公理，为了证明蕴含后件 $\forall x.\ F[x]$ 成立，我们需要证明蕴含前件 $(F[0] \wedge \forall x.\ (F[x] \rightarrow F[x+1]))$ 成立。蕴含前件是一个合取式，其左合取项 $F[0]$ 对应数学归纳法中的“归纳基础”，右合取项 $\forall x.\ (F[x] \rightarrow F[x+1])$ 对应数学归纳法中的“归纳推理”。下面我们依次证明“归纳基础”和“归纳推理”成立。

(归纳基础) 即证 $0 + 0 = 0$ 成立。由加 0 公理 $\forall x. 0 + x = x$ 知这是成立的。

(归纳推理) 即证 $\forall x.\ (0 + x = x \rightarrow 0 + (x + 1) = x + 1)$ 成立。假设 $0 + x = x$ 成立，只需证明 $0 + (x + 1) = x + 1$ 成立。根据加法后继公理 $\forall x, y.\ x + (y + 1) = (x + y) + 1$，知 $0 + (x + 1) = (0 + x) + 1$ 成立。根据归纳假设，知 $(0 + x) + 1 = x + 1$ 成立。由“=”的传递性公理 $\forall x, y.\ x = y \wedge y = z \rightarrow x = z$，知 $0 + (x + 1) = x + 1$ 成立。

由归纳基础和归纳推理成立，结合归纳公理，我们知道 $\forall x.\ 0 + x = x$ 成立。

在上面的证明中，我们并没有严格遵循 $\mathcal{T}$-有效的定义，去证明 $\forall x.\ 0 + x = x$ 是一个 $\mathcal{T}_{PA}$-有效式，它更像是一个非形式化的数学证明，但依旧是严谨的。

参照上面的证明过程，请你给出如下命题的证明 (在保证严谨的前提下，你的证明可以适当简略)。

1-1 (加法结合律) $\forall x, y, z.\ x + (y + z) = (x + y) + z$ 是一个 $\mathcal{T}_{PA}$-有效式。

Solution ■

1-2 (加法交换律) $\forall x, y.\ x + y = y + x$ 是一个 $\mathcal{T}_{PA}$-有效式。

Solution ■

Problem 2: 程序语义

如果对于任意的状态 s，$[\![e_1]\!]_s = [\![e_2]\!]_s$，则称表达式 e_1 和 e_2 是语义等价的。

2-1 证明下面各组表达式语义等价：

(1) 算术表达式：$3 * x$ 和 $x + x + x$；

(2) 算术表达式：$e_1 * (e_2 - e_3)$ 和 $e_1 * e_2 - e_1 * e_3$；

(3) 布尔表达式：$\neg(e_1 \leq e_2)$ 和 $e_2 \leq e_1 - 1$；

(4) 布尔表达式：$p \wedge q$ 和 $q \wedge p$；

(5) 布尔表达式：$(p \vee q) \wedge (p \vee \neg q)$ 和 p。

注：本题目的之一是帮助大家区分语法和语义。程序中的算术表达式和布尔表达式都是程序语言中的语法对象，分别对应于一阶逻辑中的项和公式。在本题中，$p \wedge q$ 和 $q \wedge p$ 是不同的语法对象，但是二者语义等价。

Solution ■

2-2 表达式 e/c 在状态 s 下的值定义为 $[\![e/c]\!]_s = \lfloor [\![e]\!]_s / c \rfloor$，其中 c 是不为 0 的整数。

(1) 证明 $(x_1 + x_2)/2$ 和 $x_1 + (x_2 - x_1)/2$ 语义等价；

(2) 在本课程的第一次课上，介绍了一个二分查找的 C 语言程序。当时为了修复程序中的漏洞，我们将表达式 $(low + high)/2$ 修改成了 $low + (high - low)/2$。你认为在那个程序中，$(low + high)/2$ 与 $low + (high - low)/2$ 是否语义等价？你的结论和第 (1) 问是否矛盾呢？为什么？

(3) 为了更好地理解计算模型上的差异，你能给出 C 语言中两个 32 位 `int` 型变量相加的语义吗？即如何定义 $[\![n_1 + n_2]\!]_s$，其中 n_1, n_2 均为 `int` 型变量，故只需考虑满足 $2^{-31} \leq [\![n_1]\!]_s, [\![n_2]\!]_s < 2^{31}$ 的状态 s。根据你的定义，讨论当 low 和 $high$ 的值满足什么条件时，$(low + high)/2$ 与 $low + (high - low)/2$ 值相等；什么时候不相等？

2-3 证明 IMP 程序语句 **while** $(x < 0)$ $\{x := y * y\}$ 和 **if** $(x < 0)$ $\{x := y * y\}$ **else skip** 是语义等价的，并给出其关系语义的显式表达。

Solution ■

2-4 证明 IMP 程序语句 **while** (p) $\{st\}$ 和 **if** (p) $\{st; \textbf{while}\ (p)\ \{st\}\}$ **else skip** 是语义等价的 (课件上已经给出了本题的证明框架，因此你要做的是补全课件上的证明)。

Solution ■