程序理论-第一次作业

Notes

IMP 定义

共 5 种语句，其中 var 为变量，$aexp$ 为算术表达式，$bexp$ 为布尔表达式：

$$
\begin{aligned}
com = {} & \mathbf{skip} && (\mathsf{skip}) \\
\mid {} & var := aexp && (\mathsf{assign}) \\
\mid {} & com; com && (\mathsf{seq}) \\
\mid {} & \mathbf{if}\ bexp\ \mathbf{then}\ com\ \mathbf{else}\ com && (\mathsf{if}) \\
\mid {} & \mathbf{while}\ bexp\ \mathbf{do}\ com && (\mathsf{while})
\end{aligned}
$$

程序状态

变量到值的映射，例如 $s = \llparenthesis a := 2, b := 3, c := 0 \rrparenthesis$；用 $\llbracket e \rrbracket_s$ 表示表达式 e 在状态 s 的值。
在 LaTeX 中引入 `stmaryrd` 包，并在公式环境中使用 `\llbracket` `\rrbracket` 打出 $\llbracket\ \rrbracket$，使用 `\llparenthesis` `\rrparenthesis` 打出 $\llparenthesis\ \rrparenthesis$。

大步操作语义 (Big-step operational semantics)

程序 c 将状态 s 转移到状态 t：

$$(c, s) \Rightarrow t$$

推理规则：

- $\dfrac{}{(\mathbf{skip}, s) \Rightarrow s}$ skip, $\dfrac{(c_1, s) \Rightarrow s' \quad (c_2, s') \Rightarrow s''}{(c_1; c_2, s) \Rightarrow s''}$ seq；
- $\dfrac{}{(v := e, s) \Rightarrow s(v := \llbracket e \rrbracket_s)}$ assign，其中 $s(v := \llbracket e \rrbracket_s)$ 表示用 $v := \llbracket e \rrbracket_s$ 更新 s；
- $\dfrac{\llbracket b \rrbracket_s = \mathsf{true} \quad (c_1, s) \Rightarrow t}{(\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, s) \Rightarrow t}$ ifTrue，$\dfrac{\llbracket b \rrbracket_s = \mathsf{false} \quad (c_2, s) \Rightarrow t}{(\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, s) \Rightarrow t}$ ifFalse；
- $\dfrac{\llbracket b \rrbracket_s = \mathsf{false}}{(\mathbf{while}\ b\ \mathbf{do}\ c, s) \Rightarrow s}$ whileFalse
- $\dfrac{\llbracket b \rrbracket_s = \mathsf{true} \quad (c, s) \Rightarrow s' \quad (\mathbf{while}\ b\ \mathbf{do}\ c, s') \Rightarrow s''}{(\mathbf{while}\ b\ \mathbf{do}\ c, s) \Rightarrow s''}$ whileTrue

小步操作语义 (Small-step operational semantics)

程序 c 执行一步，剩余部分为 c'，从状态 s 转移到状态 s'：

$$(c, s) \to (c', s')$$

如果 c 是 **skip** 表示程序终止了。

推理规则：

- $\dfrac{}{(v := e, s) \to (\mathbf{skip}, s(v := [\![e]\!]_s))}$ assign
- $\dfrac{(c_1, s) \to (c_1', s')}{(c_1; c_2, s) \to (c_1'; c_2, s')}$ seq1， $\dfrac{}{(\mathbf{skip}; c_2, s) \to (c_2, s)}$ seq2
- $\dfrac{[\![b]\!]_s = \mathsf{true}}{(\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, s) \to (c_1, s)}$ ifTrue， $\dfrac{[\![b]\!]_s = \mathsf{false}}{(\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, s) \to (c_2, s)}$ ifFalse
- $\dfrac{}{(\mathbf{while}\ b\ \mathbf{do}\ c, s) \to (\mathbf{if}\ b\ \mathbf{then}\ (c; \mathbf{while}\ b\ \mathbf{do}\ c)\ \mathbf{else\ skip}, s)}$ while

符号 $\to^*$ 表示 $\to$ 的自反传递闭包：$(c, s) \to^* (c', s')$ 表示 (c, s) 执行零步或多步到达 (c', s')，$(c, s) \to^* (\mathbf{skip}, s')$ 表示从 (c, s) 执行最终停止到状态 s' 。

自反传递闭包 $\to^*$ 的推理规则（归纳推理）：

- 自反性：$\dfrac{}{(c, s) \to^* (c, s)}$
- 传递性：$\dfrac{(c_1, s_1) \to (c_3, s_3) \quad (c_3, s_3) \to^* (c_2, s_2)}{(c_1, s_1) \to^* (c_2, s_2)}$

大步小步语义的等价性

课堂上讲了从大步语义 $(c, s) \Rightarrow t$ 推出小步语义 $(c, s) \to^* (\mathbf{skip}, t)$ 的证明过程：

证明由 $(c, s) \Rightarrow t$ 得出 $(c, s) \to^* (\mathbf{skip}, t)$ 。

首先需要证明小步语义的一个引理：$(c_1, s_1) \to^* (c_2, s_2)$ 推出 $(c_1; c', s_1) \to^* (c_2; c', s_2)$ 。

- 基础：由自反性 $(c, s) \to^* (c, s)$ 有 $(c; c', s) \to^* (c; c', s)$。
- 归纳：假定有 $(c_1, s_1) \to (c_3, s_3)$ 和 $(c_3, s_3) \to^* (c_2, s_2)$，需要证 $(c_1; c', s_1) \to^* (c_2; c', s2)$。
  - 由小步语义推理规则 seq1 有 $(c_1; c', s_1) \to (c_3; c', s_3)$;
  - 由归纳假设，根据 $(c_3, s_3) \to^* (c_2, s_2)$ 有 $(c_3; c', s_3) \to^* (c_2; c', s_2)$;
  - 由 $\to^*$ 的传递性，得 $(c_1; c', s_1) \to^* (c_2; c', s_2)$。

按大步语义推理的步数归纳 $(c, s) \Rightarrow t$，根据最后使用的大步语义推理规则划分情况：

- skip：当 $c = \mathbf{skip}$ 且 $s = t$，只需证 $(\mathbf{skip}, s) \to^* (\mathbf{skip}, s)$，由 $\to^*$ 自反性可得。
- assign：当 $c = (v := e)$ 且 $t = s(v := [\![e]\!]_s)$，只需证 $(v := e, s) \to^* (\mathbf{skip}, s(v := [\![e]\!]_s))$，由小步语义的 assign 推理规则可得。
- seq：当 $c = c_1; c_2$ 并且有状态 s' 使得 $(c_1, s) \Rightarrow s'$ 且 $(c_2, s') \Rightarrow t$。由归纳假设，有 $(c_1, s) \to^* (\mathbf{skip}, s')$ 和 $(c_2, s') \to^* (\mathbf{skip}, t)$。根据引理，有 $(c_1; c_2, s) \to^* (\mathbf{skip}; c_2, s')$，结合小步语义 seq2 推理规则，得 $(c_1; c_2, s) \to^* (\mathbf{skip}, t)$。

- ifTrue：当 $c = \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2$，$[\![b]\!]_s = \mathsf{true}$ 且 $(c_1, s) \Rightarrow t$，由归纳假设有 $(c_1, s) \to^* (\textbf{skip}, t)$，结合小步语义 ifTrue 推理规则可得 $(c, s) \to^* (\textbf{skip}, t)$。ifFalse 情况类似可证。
- whileFalse：当 $c = \textbf{while } b \textbf{ do } c'$，$[\![b]\!]_s = \mathsf{false}$ 且 $s = t$，由小步语义的 while 和 ifFalse 推理规则，可得 $(c, s) \to^* (\textbf{skip}, t)$。
- whileTrue：当 $c = \textbf{while } b \textbf{ do } c'$，$[\![b]\!]_s = \mathsf{true}$，$(c', s) \Rightarrow s'$ 且 $(c, s') \Rightarrow t$。根据归纳假设，有 $(c', s) \to^* (\textbf{skip}, s')$ 和 $(c, s') \to^* (\textbf{skip}, t)$。根据引理有 $(c'; c, s) \to^* (\textbf{skip}; c, s')$，结合小步语义的 while, ifTrue, seq2 推理规则得 $(c, s) \to^* (\textbf{skip}, t)$。

综上，对于任何的 $(c, s) \Rightarrow t$ 都可以推出 $(c, s) \to^* (\textbf{skip}, t)$。

Written Part

Problem 1

Carefully write down (as in the lecture slides) the other direction in the equivalence between big-step and small-step semantics.
从小步语义 $(c,s) \to^* (\mathbf{skip}, t)$ 推出大步语义 $(c,s) \Rightarrow t$。

解：与课上讲的大步语义证小步语义类似的归纳。如果有 $(c_1, s_1) \to (c_2, s_2)$，并且 $(c_2, s_2) \to^* (\mathbf{skip}, t)$ ，根据归纳假设有 $(c_2, s_2) \Rightarrow t$；由 $\to^*$ 的传递推理规则可以得到 $(c_1, s_1) \to^* (\mathbf{skip}, t)$。接下来只需要根据 $(c_1, s_1) \to (c_2, s_2)$ 所采用的小步语义推理规则分类讨论，并转到对应的大步推理规则从 $(c_2, s_2) \Rightarrow t$ 推出 $(c_1, s_1) \Rightarrow t$ 即可归纳得到对 $(c_1, s_1) \to^* (\mathbf{skip}, t)$ 也能推出 $(c_1, s_1) \Rightarrow t$。
基础情况：当 $c = \mathbf{skip}, s = t$ 时，由 $\to^*$ 的自反性，$(c,s) \to^* (\mathbf{skip}, t)$，由大步语义的 skip 推理规则，$(\mathbf{skip}, s) \Rightarrow t$，此时有 $(c,s) \Rightarrow t$ 成立。
归纳推理：按最后一步采用的小步语义推理规则分类

- assign：当 $c = (v := e)$ 且 $t = s(v := \llbracket e \rrbracket_s)$，有 $(v := e; s) \to (\mathbf{skip}, s(v := \llbracket e \rrbracket_s))$。只需证 $(v := e, s) \Rightarrow s(v := \llbracket e \rrbracket_s)$，由大步语义的 assign 推理规则可得。
- seq2：当 $c = \mathbf{skip}; c_2$ 时，有 $(\mathbf{skip}; c_2, s) \to (c_2, s)$。根据归纳假设，由 $(c_2, s) \to^* (\mathbf{skip}, t)$ 得 $(c_2, s) \Rightarrow t$；根据 $\to^*$ 的传递性，有 $(\mathbf{skip}; c_2, s) \to^* (\mathbf{skip}, t)$。只需证 $(\mathbf{skip}; c_2, s) \Rightarrow t$，可根据大步语义的 skip, seq 推理规则，由 $(\mathbf{skip}, s) \Rightarrow s$ 和 $(c_2, s) \Rightarrow t$ 得证。
- seq1：当 $c = c_1; c_2$ 且 $(c_1, s) \to (c_1', s')$，有 $(c_1; c_2, s) \to (c_1'; c_2, s')$。根据归纳假设，如果 $(c_1'; c_2, s') \to^* (\mathbf{skip}, t)$ 则 $(c_1'; c_2, s') \Rightarrow t$，只需证 $(c_1; c_2, s) \Rightarrow t$。由大步语义的 seq 规则，存在 s'' 使得 $(c_1', s') \Rightarrow s''$ 且 $(c_2, s'') \Rightarrow t$。根据课上证过的从大步语义到小步语义的等价性，由 $(c_1', s') \Rightarrow s''$ 可以得 $(c_1', s') \to^* (\mathbf{skip}, s'')$，再根据 $(c_1, s) \to (c_1', s')$ 和传递性得 $(c_1, s) \to^* (\mathbf{skip}, s'')$，由归纳假设得到 $(c_1, s) \Rightarrow s''$。再由 $(c_2, s'') \Rightarrow t$ 和大步语义的 seq 推理规则得到 $(c_1; c_2, s) \Rightarrow t$。
- ifTrue：当 $c = \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2$ 且 $\llbracket b \rrbracket_s = \mathsf{true}$，有 $(c,s) \to (c_1, s)$。如果 $(c_1, s) \to^* (\mathbf{skip}, t)$ 则由传递性有 $(c,s) \to^* (\mathbf{skip}, t)$，再由归纳假设有 $(c_1, s) \Rightarrow t$。根据大步语义的 ifTrue 推理规则，可得 $(c,s) \Rightarrow t$。ifFalse 与 ifTrue 类似。
- while：当 $c = \mathbf{while}\ b\ \mathbf{do}\ c'$ 有 $(c,s) \to (\mathbf{if}\ b\ \mathbf{then}\ (c'; \mathbf{while}\ b\ \mathbf{do}\ c')\ \mathbf{else}\ \mathbf{skip}, s)$。
  当 $\llbracket b \rrbracket_s = \mathsf{false}$ 时，根据大步语义的 whileFalse 推理规则，有 $(c,s) \Rightarrow s$。
  当 $\llbracket b \rrbracket_s = \mathsf{true}$ 时，当 $(c'; \mathbf{while}\ b\ \mathbf{do}\ c', s) \to^* (\mathbf{skip}, t)$ 时，由小步语义的 ifTrue 规则有 $(\mathbf{if}\ b\ \mathbf{then}\ (c'; \mathbf{while}\ b\ \mathbf{do}\ c')\ \mathbf{else}\ \mathbf{skip}, s) \to^* (\mathbf{skip}, t)$，再由传递性有 $(c,s) \to^* (\mathbf{skip}, t)$。此时只需证 $(c,s) \Rightarrow t$。根据归纳假设，$(c'; \mathbf{while}\ b\ \mathbf{do}\ c', s) \Rightarrow t$，由大步语义的 seq 推理规则，存在 s' 使得 $(c', s) \Rightarrow s'$ 并且 $(\mathbf{while}\ b\ \mathbf{do}\ c', s') \Rightarrow t$；再由大步语义的 whileTrue 推理规则，有 $(\mathbf{while}\ b\ \mathbf{do}\ c', s') \Rightarrow t$，即 $(c,s) \Rightarrow t$ 得证。

Problem 2

Define (by induction on the structure of the program) a function *assigned* from programs to a set of variables that may be assigned to in the program. Prove (by induction on big-step semantics) that if $(c, s) \Rightarrow t$ and x are not in the assigned set, then the value of x in s and t are the same.

解：要定义的函数 *assigned* 参数为一个 IMP 程序，返回值为一个集合。形式化的定义如下：

$$assigned(com) = \begin{cases} \varnothing & com = \textbf{skip} \\ \{var\} & com = (var := aexp) \\ assigned(com_1) \cup assigned(com_2) & com = (com_1; com_2) \\ assigned(com_1) \cup assigned(com_2) & com = (\textbf{if}\ bexp\ \textbf{then}\ com_1\ \textbf{else}\ com_2) \\ assigned(com) & com = (\textbf{while}\ bexp\ \textbf{do}\ com) \end{cases}$$

要证明的结论：如果 $(c, s) \Rightarrow t$ 并且 $x \notin assigned(c)$ 则 $[\![x]\!]_s = [\![x]\!]_t$。

归纳证明：

- $c = \textbf{skip}$ 则一定有 $t = s$ 和 $assigned(c) = \varnothing$，显然 $[\![x]\!]_s = [\![x]\!]_t$ 成立。
- $c = (v := e)$，此时 $t = s(v := [\![e]\!]_s)$ 且 $assigned(c) = \{v\}$。若 $x \notin assigned(c)$ 则 $x \neq v$，$[\![x]\!]_s = [\![x]\!]_{s(v:=[\![e]\!]_s)} = [\![x]\!]_t$，结论成立。
- $c = (c_1; c_2)$，设 $(c_1, s) \Rightarrow s', (c_2, s') \Rightarrow t$。根据归纳假设：
  - 当 $x \notin assigned(c_1)$ 时有 $[\![x]\!]_s = [\![x]\!]_{s'}$
  - 当 $x \notin assigned(c_2)$ 时有 $[\![x]\!]_{s'} = [\![x]\!]_t$

  因为 $assigned(c) = assigned(c_1) \cup assigned(c_2)$，$x \notin assigned(c)$ 意味着 $x \notin assigned(c_1)$ 且 $x \notin assigned(c_2)$，所以 $[\![x]\!]_s = [\![x]\!]_{s'} = [\![x]\!]_t$。
- $c = (\textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2)$，此时 $assigned(c) = assigned(c_1) \cup assigned(c_2)$，$x \notin assigned(c)$ 意味着 $x \notin assigned(c_1)$ 且 $x \notin assigned(c_2)$。

  对 b 分类讨论：
  - $[\![b]\!]_s = \mathsf{true}$，由 ifTrue 推理规则，设 $(c_1, s) \Rightarrow t$ 则 $(c, s) \Rightarrow t$。
    根据归纳假设，由 $x \notin assigned(c_1)$ 可推出 $[\![x]\!]_s = [\![x]\!]_t$。
  - $[\![b]\!]_s = \mathsf{false}$，由 ifFalse 推理规则，设 $(c_2, s) \Rightarrow t$ 则 $(c, s) \Rightarrow t$。
    根据归纳假设，由 $x \notin assigned(c_2)$ 可推出 $[\![x]\!]_s = [\![x]\!]_t$。

  无论 $[\![b]\!]_s$ 取何值，总有 $[\![x]\!]_s = [\![x]\!]_t$。
- $c = (\textbf{while}\ b\ \textbf{do}\ c')$，此时 $assigned(c) = assigned(c')$。对 b 分类讨论：
  - $[\![b]\!]_s = \mathsf{false}$，由 whileFalse 规则 $t = s$，显然 $[\![x]\!]_s = [\![x]\!]_t$ 成立。
  - $[\![b]\!]_s = \mathsf{true}$，由 whileTrue 规则设 $(c', s) \Rightarrow s', (\textbf{while}\ b\ \textbf{do}\ c', s') \Rightarrow t$。根据归纳假设，对 $(c', s) \Rightarrow s'$ 当 $x \notin assigned(c')$ 时 $[\![x]\!]_s = [\![x]\!]_{s'}$；对 $(\textbf{while}\ b\ \textbf{do}\ c', s') \Rightarrow t$ 当 $x \notin assigned(c)$ 时 $[\![x]\!]_{s'} = [\![x]\!]_t$。结合两者有 $[\![x]\!]_s = [\![x]\!]_t$ 成立。

  无论 $[\![b]\!]_s$ 取何值，总有 $[\![x]\!]_s = [\![x]\!]_t$。

Problem 3

Define a small-step semantics for the evaluation of arithmetic expressions, specifying a left-to-right evaluation order. The syntax for arithmetic expressions is given by:

$$expr = N\ int \mid V\ var \mid Plus\ expr\ expr$$

where N denotes constants, V denotes variables, and $Plus$ denotes addition. For example, it should be possible to derive from the semantics the following:

$$(Plus\ (Plus\ (N\ 3)\ (V\ x))\ (V\ y), (\!|x := 5, y := 2|\!)) \to^* (N\ 10, (\!|x := 5, y := 2|\!))$$

表达式求值过程的小步语义：$(e, s) \to (e', s)$ （显然状态 s 不会被改变）

- $e = N\ n$，小步执行终止。更进一步地，任何 e 都存在 n 使得 $(e, s) \to^* (N\ n, s)$。
- $e = V\ v$，有 $(V\ v, s) \to (N\ [\![v]\!]_s, s)$。
- $e = Plus\ e_1\ e_2$，分三种情况
  - 两个操作数均已确定，即 $e_1 = N\ n_1, e_2 = N\ n_2$，则 $(e, s) \to (N\ (n_1 + n_2), s)$
  - 第一个操作数已确定，则将第二个操作数运算一步，即 $(e, s) \to (Plus\ e_1, e_2', s)$
  - 两个操作数均未确定，则将第一个操作数运算一步，即 $(e, s) \to (Plus\ e_1', e_2, s)$

用形式化的小步语义推理规则描述：

- $(N\ n, s)$ 为终止状态，无推理规则
- $\dfrac{}{(V\ v, s) \to (N\ [\![v]\!]_s, s)}$ var
- $\dfrac{}{(Plus\ (N\ n_1)\ (N\ n_2), s) \to (N\ n_1 + n_2, s)}$ plus2
- $\dfrac{(e_2, s) \to (e_2', s)}{(Plus\ (N\ n_1)\ e_2, s) \to (Plus\ (N\ n_1)\ e_2', s)}$ plus1
- $\dfrac{(e_1, s) \to (e_1', s)}{(Plus\ e_1\ e_2, s) \to (Plus\ e_1'\ e_2, s)}$ plus0

示例表达式的小步推理过程：

$$\begin{aligned}
& (Plus\ (Plus\ (N\ 3)\ (V\ x))\ (V\ y), (\!|x := 5, y := 2|\!)) \\
\to\ & (Plus\ (Plus\ (N\ 3)\ (N\ 5))\ (V\ y), (\!|x := 5, y := 2|\!)) && (\mathsf{plus0}, \mathsf{plus1}) \\
\to\ & (Plus\ (N\ 8)\ (V\ y), (\!|x := 5, y := 2|\!)) && (\mathsf{plus0}, \mathsf{plus2}) \\
\to\ & (Plus\ (N\ 8)\ (N\ 2), (\!|x := 5, y := 2|\!)) && (\mathsf{plus1}, \mathsf{var}) \\
\to\ & (N\ 10, (\!|x := 5, y := 2|\!)) && (\mathsf{plus2})
\end{aligned}$$