Notes on The Formal Semantics of Programming Languages

Ray Li

October 16, 2024

Contents

1 Chapter 3: some principles of induction

1.1 3.1 Mathematical induction Excercise Answers

Question 1. *(E3.2)*
A string is a sequence of symbols. A string $a_1a_2\cdots a_n$ with n positions occupied by symbols is said to have length n. A string can be empty in which case it is said to have length 0. Two strings s and t can be concatenated to form the string st. Use mathematical induction to show there is no string u which satisfies $au = ub$ for two distinct symbols a and b.

Proof. by induction on the length of u
Let $A = \{n \mid \text{len}(u) = n \text{ and } au \neq ub\}$.

1. Base Case:
   - $0 \in A$: By the hypothesis, since $a \neq b$, it is clear that when u is the empty string (i.e., length 0), $au \neq ub$. Thus, $0 \in A$.
   - $1 \in A$: If u has length 1, it is easy to see that $au \neq ub$ because $a \neq b$.
   - $2 \in A$: For $u = xy$, where $\text{len}(u) = 2$, we have $axy \neq xyb$, given that $a \neq b$.
2. Inductive Step: Assume $n \in A$. We want to show that $n+1 \in A$ for $n \geq 2$.

   Proof by contradiction: Suppose $\text{len}(u) = n+1$ and $au = ub$. Since $n+1 \geq 3$, we can write $u = qu'p$, where $\text{len}(q) = \text{len}(p) = 1$. Thus, $au = aqu'p$ and $ub = qu'pb$, leading to the equation $aqu'p = qu'pb$.

   From this, we deduce that $a = q$. By eliminating the leftmost characters, we are left with $qu'p = u'pb$. Since $q = a$, the equation becomes $au'p = u'pb$, where $\text{len}(u'p) = n$. By the induction hypothesis, $au'p \neq u'pb$, leading to a contradiction.

   Therefore, $n+1 \in A$.

By mathematical induction, we conclude that there is no string u such that $au = ub$ for two distinct symbols a and b.

□

Question 2. *(E3.6)*
What goes wrong when you try to prove the execution of commands is deterministic by using structural induction on commands?

Proof. The issue arises in verifying the **while** rule. By the **Rules for While-loops**, we aim to show that:

$$\langle \textbf{while } b \textbf{ do } c, \sigma\rangle \to \sigma_1 \quad \text{and} \quad \langle \textbf{while } b \textbf{ do } c, \sigma\rangle \to \sigma_2 \implies \sigma_1 = \sigma_2$$

When $\langle b, \sigma\rangle \to \text{false}$, there is no problem. However, the issue emerges when $\langle b, \sigma\rangle \to \text{true}$.

To argue this, we use **Proposition 2.8** (**while** b **do** $c \sim$ **if** b **then** $c; w$ **else skip**). This allows us to simplify the verification to:

$$\langle \textbf{if } b \textbf{ then } c; w \textbf{ else skip}, \sigma\rangle \to \sigma_1 \quad \textbf{and} \quad \langle \textbf{if } b \textbf{ then } c; w \text{ else skip}, \sigma\rangle \to \sigma_2 \implies \sigma_1 = \sigma_2$$

However, if we attempt to use structural induction, we would need to assume something smaller or a predecessor of w. Yet, in each "previous" case, we still need to argue about w itself (rather than a predecessor or a smaller set). Therefore, it is not possible to prove this by structural induction.

$\square$

[Ray's Note 1: Here there exists two kind of structural induction. Or more specifically one is structural induction and the other one is direvation induction. They are some subtle differences between the two. Not to confuse with the proof for Theorem 3.11 in the book. The structural induction here refers to a way of "syntax deviding", like how you proof arithmetic ones from the definition of

$$a ::= n \mid X \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1.$$

by each case (here the case refers to n, a_0, etc.). Yet the direvation induction is refers the different rules (here the case is one kind of rule.).]

Question 3. *(E 3.8) Let $\prec$ be a well-founded relation on a set B. Prove*

1. *its transitive closure $\prec^+$ is also well-founded,*
2. *its reflexive, transitive closure $\prec^*$ is a partial order.*

Proof. 1. The transitive closure $\prec^+$ is well-founded:

proof by way of contradiction (BWOC). Assume there exists an infinite chain:

$$\cdots \prec^+ a_{n+1} \prec^+ a_n \prec^+ \cdots$$

Now, pick an element a_k, and $\exists b \prec^+ a_k$ such that $b \prec a_k$, call such b a_{k+1} . If no such b exists, we would have $\forall b, a_k \prec b$, which contradicts the assumption of an infinite descending chain. By continuing this process, we can pick an infinite chain of $\prec$, which contradicts the well-foundedness of $\prec$. Hence, $\prec^+$ is well-founded.

2. The reflexive, transitive closure $\prec^*$ is a partial order:

By the definition of a partial order, we need to show that the relation is reflexive, antisymmetric, and transitive. Since $\prec^*$ is already reflexive and transitive by definition, we only need to prove antisymmetry.

BWOC. : Assume there exist $a, b \in B$ such that $a \prec^* b$ and $b \prec^* a$, yet $a \neq b$. If $a \neq b$, then it must be the case that either $a \prec^* b$ or $b \prec^* a$, but not both. This leads to a contradiction, hence $a = b$. Therefore, $\prec^*$ is antisymmetric, and thus a partial order.

$\square$

Question 4. *(E 3.9) For a suitable well-founded relation on strings, use the "no counterexample" approach described above to show there is no string u which satisfies $au = ub$ for two distinct symbols a and b. Compare your proof with another by well-founded induction (and with the proof by mathematical induction asked for in Section 3.1).*

Proof. Define $\prec$ as a relation on the set $U \times U$ (the Cartesian product of the set of strings U) such that for two strings a and b, if $\text{len}(a) < \text{len}(b)$, then $a \prec b$.

Claim: $\prec$ is well-founded on the set of strings: we can identify the minimum element, which is the empty string (having length 0).

Thus, we form the set $A = \{u \mid au = ub, \text{ for all } a \neq b\}$. We aim to show that such a set does not exist.

Proof by way of contradiction (BWOC): Assume A exists. Since $A \subseteq U$, there exists a smallest element, call it u, such that $au = ub$. By the argument in **Exercise 3.2**, we know that $\text{len}(u) \neq 0, 1$. Therefore, we can write $u = pu'$, where $\text{len}(p) = 1$.

Thus, $apu' = pu'b$, implying $a = p$. This gives us $apu' = au'b$, and by removing the leftmost characters, we get $pu' = u'b$, where $p = a$. Hence, $au' = u'b$, where $\text{len}(u') < \text{len}(u)$, contradicting the assumption that u is the smallest element in A.

From here, we can argue in two ways:

1. Directly conclude that this leads to a contradiction, thus proving the theorem.

2. Alternatively, we can state that no minimum element exists in A, meaning that $\prec$ is not well-founded on A. Hence, the theorem is proven.

$\square$

2 Ray's Notes Summary

[Ray's Note 1 (Page 3): Here there exists two kind of structural induction. Or more specifically one is structural induction and the other one is direvation induction. They are some subtle differences between the two. Not to confuse with the proof for Theorem 3.11 in the book. The structural induction here refers to a way of "syntax deviding", like how you proof arithmetic ones from the definition of

$$a ::= n \mid X \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1.$$

by each case (here the case refers to n, a_0, etc.). Yet the direvation induction is refers the different rules (here the case is one kind of rule.).]