198:205
Discrete Structures
Professor McCarty

Solutions to Homework 10

Problem 1(c):

We need to show that *search* is single-valued and well-defined on every element in $\Sigma \times \Sigma^*$.

The proof is by structural induction on the second argument of *search*. For the basis step, $search(\sigma, \lambda) = \lambda$ by the base case of the definition, and thus *search* is single-valued and well-defined on the ordered pair (σ, λ) for every $\sigma \in \Sigma$.

For the inductive step, assume for an arbitrary $x \in \Sigma^*$ that $search(\sigma_1, x)$ is single-valued and well-defined for every $\sigma_1 \in \Sigma$. Then the same is true for $search(\sigma_1, \sigma_2 x)$, for every $\sigma_2 \in \Sigma$, by the recursive clause of the definition. In particular, if $\sigma_1 = \sigma_2$, then the value of *search* on the ordered pair $(\sigma_1, \sigma_2 x)$ equals $\sigma_2 x$, and if $\sigma_1 \neq \sigma_2$, then the value of *search* on $(\sigma_1, \sigma_2 x)$ is the same as its value on (σ_1, x), which is uniquely determined by the induction hypothesis.

It follows, by induction, that *search* is single-valued and well-defined on every element in $\Sigma \times \Sigma^*$.

Problem 2(c):

By the extremal clause in the definition of $\mathbf{S}$, the only elements in $\mathbf{S}$ are those generated by either the base clause or the recursive clause of the definition. But (ab, λ) cannot be generated by the base clause, since $\text{ab} \neq \lambda$, and (ab, λ) cannot be generated by the recursive clause, since σy cannot equal λ for any $\sigma \in \Sigma$. Therefore $(\text{ab}, \lambda) \notin \mathbf{S}$.

Problem 2(d):

Similarly, for any $x \in \Sigma^*$, if $x \neq \lambda$, then (x, λ) cannot be generated by either the base clause or the recursive clause of the definition of $\mathbf{S}$. Thus $(x, \lambda) \notin \mathbf{S}$.

Problem 3(a):

The proof is by structural induction on the second argument of *search*.

For the basis step, we need to prove:

$$\forall \sigma_1, \sigma_2 \in \Sigma, \forall z \in \Sigma^* : \quad search(\sigma_1, \lambda) = \sigma_2 z \quad \rightarrow \quad (\sigma_2 z, \lambda) \in \mathbf{S} \ \wedge \ \sigma_1 = \sigma_2.$$

But this implication is vacuously true, since its antecedent is false. To see this, note that $search(\sigma_1, \lambda) = \lambda$, by definition, but λ cannot be equal to $\sigma_2 z$ for any $\sigma_2 \in \Sigma$ (i.e., the empty string cannot be equal to a nonempty string). Thus the left-hand side of the proposition is false in the base case, which means that the proposition itself is true.

For the inductive step, we assume that the proposition is true for an arbitrary $x \in \Sigma^*$, and we try to show that it is also true when x is replaced by $\sigma_3 x$, for any $\sigma_3 \in \Sigma$. Our induction hypothesis is thus:

$$\forall \sigma_1, \sigma_2 \in \Sigma, \forall z \in \Sigma^* : \quad search(\sigma_1, x) = \sigma_2 z \quad \rightarrow \quad (\sigma_2 z, x) \in \mathbf{S} \ \wedge \ \sigma_1 = \sigma_2.$$

and we need to show:

$$\forall \sigma_1, \sigma_2 \in \Sigma, \forall z \in \Sigma^* : \quad search(\sigma_1, \sigma_3 x) = \sigma_2 z \quad \rightarrow \quad (\sigma_2 z, \sigma_3 x) \in \mathbf{S} \ \wedge \ \sigma_1 = \sigma_2.$$

We will construct a direct proof of this latter proposition, that is, we will assume that the left-hand side is true for some arbitrary σ_1 and $\sigma_2 \in \Sigma$ and some arbitrary $z \in \Sigma^*$, and try to prove that the right-hand side is also true.

So assume $search(\sigma_1, \sigma_3 x) = \sigma_2 z$. The recursive clause in the definition of *search* splits into two cases, depending on whether $\sigma_1 = \sigma_3$ or not, and we need to consider these two cases separately.

1. Suppose $\sigma_1 = \sigma_3$. Then $search(\sigma_1, \sigma_3 x) = \sigma_3 x$ by definition, which means that $\sigma_3 x = \sigma_2 z$. We can conclude two facts from this:

   (a) $\sigma_1 = \sigma_3 = \sigma_2$, and

   (b) $(\sigma_2 z, \sigma_3 x) \in \mathbf{S}$, since $(\sigma_2 z, \sigma_2 z) \in \mathbf{S}$ by the definition of $\mathbf{S}$.

   But this is just the right-hand side of the proposition we are trying to prove, so this completes the proof for the first case.

2. Suppose $\sigma_1 \neq \sigma_3$. Then by the definition of *search*:

   $$search(\sigma_1, \sigma_3 x) = search(\sigma_1, x),$$

   and we can apply our induction hypothesis. Combining the two equalities for $search(\sigma_1, \sigma_3 x)$, we obtain a third equality:

   $$search(\sigma_1, x) = \sigma_2 z.$$

   But this is just the left-hand side of the induction hypothesis! Thus we can conclude (under all the assumptions made so far) that the right-hand side of the induction hypothesis is true. Specifically:

(a) $\sigma_1 = \sigma_2$, and

(b) $(\sigma_2 z, x) \in \mathbf{S}$. This is almost what we want, and to finish the proof we just have to notice that we can add σ_3 back to the beginning of the second element of this ordered pair. In other words, from the recursive clause of the definition of $\mathbf{S}$, it follows that $(\sigma_2 z, \sigma_3 x) \in \mathbf{S}$.

This completes the proof for the second case

And this completes the proof of the inductive step.

Therefore, by induction, the proposition in **3(a)** is true for all $x \in \Sigma^*$.

Problem 3(b):

The proof is by structural induction on the second argument of *search*.

For the basis step, we need to prove:

$$\forall \sigma \in \Sigma : \quad search(\sigma, \lambda) = \lambda \quad \rightarrow \quad \neg \exists z [z \in \Sigma^* \wedge (\sigma z, \lambda) \in \mathbf{S}].$$

But this implication is trivially true, since its conclusion is true. To see this, note that we can rewrite the right-hand side of the proposition as:

$$\forall z [z \in \Sigma^* \rightarrow (\sigma z, \lambda) \notin \mathbf{S}],$$

which is true by the results of Problem **2(d)**.

For the inductive step, we assume that the proposition is true for an arbitrary $x \in \Sigma^*$, and we try to show that it is also true when x is replaced by $\sigma_2 x$, for any $\sigma_2 \in \Sigma$. Our induction hypothesis is thus:

$$\forall \sigma_1 \in \Sigma : \quad search(\sigma_1, x) = \lambda \quad \rightarrow \quad \neg \exists z [z \in \Sigma^* \wedge (\sigma_1 z, x) \in \mathbf{S}]$$

and we need to show:

$$\forall \sigma_1 \in \Sigma : \quad search(\sigma_1, \sigma_2 x) = \lambda \quad \rightarrow \quad \neg \exists z [z \in \Sigma^* \wedge (\sigma_1 z, \sigma_2 x) \in \mathbf{S}].$$

We will construct a proof by contradiction.

Assume, first, that $search(\sigma_1, \sigma_2 x) = \lambda$ for some arbitrary σ_1 and σ_2. Looking at the definition of *search*, it is clear that this value could not have been produced by the base clause, nor by the first part of the recursive clause, since $\sigma_2 x$ cannot be equal to λ. Therefore, it must be the case that $\sigma_1 \neq \sigma_2$ and $search(\sigma_1, \sigma_2 x) = search(\sigma_1, x) = \lambda$, i.e., the second part of the recursive clause in the definition of *search* must have been applied to obtain this result. Now assume that

$$\exists z [z \in \Sigma^* \wedge (\sigma_1 z, \sigma_2 x) \in \mathbf{S}]$$

Looking at the definition of $\mathbf{S}$, since $\sigma_1 \neq \sigma_2$, it is clear that the base clause of the definition could not have produced this result, and that we must have generated $(\sigma_1 z, \sigma_2 x) \in \mathbf{S}$ by virtue of the recursive clause. In other words, we must have had $(\sigma_1 z, x) \in \mathbf{S}$. But we now have a contradiction, since the induction hypothesis tells us that such a z cannot exist when $\mathit{search}(\sigma_1, x) = \lambda$. We have thus shown that

$$\neg\exists z[z \in \Sigma^* \wedge (\sigma_1 z, \sigma_2 x) \in \mathbf{S}].$$

This completes the proof of the inductive step.

We have thus shown, by induction, that the proposition in **3(b)** is true for all $x \in \Sigma^*$.

Problem 4(a):

The first step is to formalize what was initially only an intuitive specification of the function *search*. To do this, we will interpret the phrase "σ occurs in x" to mean "there exists a suffix of x beginning with σ," and we will assume that the relation $\mathbf{S}$ correctly formalizes the concept of a suffix.

With these assumptions, there are two parts to the specification:

1. In English:

   > The function $\mathit{search}(\sigma, x)$ returns the empty string if σ does not occur in x.

   In Logic:

   $$\forall\sigma \in \Sigma, \forall x \in \Sigma^* :$$

   $$\neg\exists z[z \in \Sigma^* \ \wedge \ (\sigma z, x) \in \mathbf{S}] \ \ \rightarrow \ \ \mathit{search}(\sigma, x) = \lambda$$

2. In English:

   > Otherwise, if σ does occur in x, then $\mathit{search}(\sigma, x)$ returns a *suffix* of x that begins with σ.

   In Logic:

   $$\forall\sigma \in \Sigma, \forall x \in \Sigma^* :$$

   $$\exists z_1[z_1 \in \Sigma^* \ \wedge \ (\sigma z_1, x) \in \mathbf{S}] \ \ \rightarrow \ \ \exists z_2[\mathit{search}(\sigma, x) = \sigma z_2 \ \wedge \ (\sigma z_2, x) \in \mathbf{S}]$$

We will now prove that our logical specification is implied by the propositions in **3(a)** and **3(b)**.

For the first part, we will prove the contrapositive. Thus, pick an arbitrary σ and an arbitrary x and assume that $\mathit{search}(\sigma, x) \neq \lambda$. Since *search* is well-defined everywhere on $\Sigma \times \Sigma^*$, though, by the results of Problem **1(c)**, it must return a nonempty string, i.e., a string in the form $\sigma' z$ for some z. But now the proposition in **3(a)** tells us that $(\sigma' z, x) \in \mathbf{S}$ and $\sigma = \sigma'$. We have thus shown that

$$\exists z[z \in \Sigma^* \ \wedge \ (\sigma z, x) \in \mathbf{S}]$$

as required.

For the second part, we will construct a direct proof. Thus, assume that the left-hand side of the second part of the specification is true for an arbitrary σ and an arbitrary x. Using the contrapositive of the proposition in **3(b)**, we can conclude from this assumption that $\mathit{search}(\sigma, x) \neq \lambda$. But now, using the proposition in **3(a)** and the fact that *search* is well-defined everywhere on $\Sigma \times \Sigma^*$, i.e., using the same reasoning as above, we can conclude that

$$\exists z[\mathit{search}(\sigma, x) = \sigma z \ \wedge \ (\sigma z, x) \in \mathbf{S}].$$

This completes the proof of the specification.

Finally, we need to show that the recursive procedure `search(c,s)` computes exactly the same values as the recursive function *search* on $\Sigma \times \Sigma^*$. This should be obvious, since `search(c,s)` is just a clause-by-clause translation of *search* using the auxiliary procedures `head(s)` and `tail(s)`. But we could also prove this fact by induction, if desired. (This proof is omitted.)

In conclusion, since we have shown that our formal specification is true of *search*, and since *search* = `search`, our specification is obviously true of `search` as well.

Problem 5(a):

We first show that the given proposition, call it P, is a loop invariant. To do this, we assume that P is true before we enter the `begin/end` block, and we assume that the execution of this block terminates normally, i.e., we assume that the `return` statement inside the block is not executed. We also assume that $\mathtt{t} \neq \lambda$. We need to show that P is still true when the `begin/end` block terminates.

At the end of the block, we have a new $t' =$ `tail(t)`. Let us use the symbol t to refer to the original value of the variable `t`. Since $t \neq \lambda$, we must have $t = \sigma t'$ for some σ. Also, we know that $\mathit{head}(t) \neq \mathtt{c}$, since we are assuming that the `return` statement was not executed. We will show that

$$\exists y[(y, t) \in \mathbf{S} \ \wedge \ \mathit{head}(y) = \mathtt{c}] \iff \exists y[(y, t') \in \mathbf{S} \ \wedge \ \mathit{head}(y) = \mathtt{c}].$$

First, assume that the left-hand side is true, and instantiate y to some particular y_0. Since $head(y_0) = \mathtt{c}$ but $head(t) \neq \mathtt{c}$, we know that $y_0 \neq t$. This means that we cannot have $(y_0, t) \in \mathbf{S}$ by virtue of the base clause of the definition of $\mathbf{S}$. Instead, (y_0, t) must be a member of $\mathbf{S}$ by virtue of the recursive clause, that is, $(y_0, t) = (y_0, \sigma t') \in \mathbf{S}$ because $(y_0, t') \in \mathbf{S}$ already. This shows that the right-hand side is true. Conversely, assume that the right-hand side is true, and instantiate y to some particular y_0. Then, applying the recursive clause of $\mathbf{S}$ directly, it follows that $(y_0, t) = (y_0, \sigma t') \in \mathbf{S}$. This shows that the left-hand side is true. But now, since we have shown that the truth value of

$$\exists y[(y, \mathtt{t}) \in \mathbf{S} \ \wedge \ head(y) = \mathtt{c}]$$

does not change when we replace `t` by `tail(t)`, we have succeeded in showing that P remains true when the `begin/end` block terminates.

Problem 5(b):

We will now use the loop invariant P to prove that the procedure `search(c,s)` satisfies the specification that we formalized in Problem **4**. Note, first, that P is trivially true at the beginning of the `while` statement, since `t` is initialized to the value of `s`. Note also that `search(c,s)` is guaranteed to terminate by the execution of one of the `return` statements, since `t` gets shorter each time around the loop. It follows that the loop invariant P will be true at the point of termination.

Consider the first part of the specification in Problem **4**. The left-hand side is equivalent to:

$$\neg\exists y[(y, \mathtt{s}) \in \mathbf{S} \ \wedge \ head(y) = \mathtt{c}].$$

If our procedure terminated with the statement `return t`, we would have $(\mathtt{t}, \mathtt{t}) \in \mathbf{S}$ and `head(t)=c`. But this violates the loop invariant, which imposes the requirement that

$$\neg\exists y[(y, \mathtt{t}) \in \mathbf{S} \ \wedge \ head(y) = \mathtt{c}].$$

Thus our procedure must terminate with the statement `return ""`, as indicated on the right-hand side of the specification.

Consider the second part of the specification in Problem **4**. The left-hand side is equivalent to:

$$\exists y[(y, \mathtt{s}) \in \mathbf{S} \ \wedge \ head(y) = \mathtt{c}].$$

If our procedure terminated with the statement `return ""`, we would have a contradiction. The loop invariant imposes the requirement that:

$$\exists y[(y, \mathtt{t}) \in \mathbf{S} \ \wedge \ head(y) = \mathtt{c}],$$

but if $\mathtt{t} = \lambda$, then $(y, \mathtt{t}) \in \mathbf{S}$ only if $y = \lambda$, and $head(\lambda) \neq \mathtt{c}$. Thus, our procedure must terminate with the statement `return t`, in which case `head(t)=c`. It is straightforward to verify that $(\mathtt{t}, \mathtt{s}) \in \mathbf{S}$. This shows that the right-hand side of the specification is true.