SOEN 331: Introduction to Formal Methods for Software Engineering

Tutorial exercises on unordered and ordered structures

Dr. Constantinos Constantinides, P.Eng.

September 16, 2024

Problem 1

1. Are bags $\{w, w, y, z, z\}$ and $\{w, w, y, y, z, z\}$ equal?

2. Is (b, a, a, s) a valid tuple?

3. Are tuples (w, y, z) and (y, w, z) equal?

4. Define the Cartesian product for sets $A = \{a, b\}$ and $B = \{w, z\}$. This is defined as $A \times B = \{(a, w), (a, z), (b, w), (b, z)\}$.

   (a) Is $(z, b) \in A \times B$?

   (b) For set $C = \{(a, w), (a, z), (w, b)\}$, is $C \subset A \times B$?

   (c) For set $D = \{(a, w), (a, z)\}$, is $D \in \mathcal{P}(A \times B)$?

(d) Is $A \times B = B \times A$?

5. For $L = \langle\langle\rangle, x, \langle\rangle\rangle, y, \langle\rangle\rangle$, define $head(L)$ and $tail(L)$:

6. $cons(w, \langle x, y, z\rangle) =$

7. $cons(a, \langle\rangle) =$

8. $cons(all, \langle your, base\rangle) =$

9. Consider list $L = \langle w, x, y, z\rangle$, deployed to implement a Stack ADT, where the head of L would correspond to the topmost position of the Stack. To implement method $push(el, L)$, what would be the result of

   (a) $cons(el, L)$?

   (b) $list(el, L)$?

   (b) $concat(list(el), L)$?

10. Consider list $L = \langle w, x, y, z\rangle$, deployed to implement a Queue ADT, where the head of L would correspond to the front position of the Queue. To implement method $enqueue(el, L)$, what would be the result of

    (a) $cons(L, list(el))$? Is it acceptable? (i.e. Does it implement the Queue protocol?)

    (b) $list(L, el)$? Is it acceptable? (i.e. Does it implement the Queue protocol?)

Solutions:

1. Are bags $\{w, w, y, z, z\}$ and $\{w, w, y, y, z, z\}$ equal? No, because even though the order of elements is not important, repetitions are allowed and the two bags do not contain the same elements.

2. Is (b, a, a, s) a valid tuple? Yes, because repetitions are allowed in a tuple.

3. Are tuples (w, y, z) and (y, w, z) equal? No, because even though the two tuples contain the same elements, the order of the elements is not the same.

4. Define the Cartesian product for sets $A = \{a, b\}$ and $B = \{w, z\}$. This is defined as $A \times B = \{(a, w), (a, z), (b, w), (b, z)\}$.

   (a) Is $(z, b) \in A \times B$? No.

   (b) For set $C = \{(a, w), (a, z), (w, b)\}$, is $C \subset A \times B$? No.

   (c) For set $D = \{(a, w), (a, z)\}$, is $D \in \mathcal{P}(A \times B)$? Yes.

   (d) Is $A \times B = B \times A$? No.

5. For $L = \langle\langle\rangle, x, \langle\rangle\rangle, y, \langle\rangle\rangle$, define $head(L)$ and $tail(L)$:

   $head(L) = \langle\langle\rangle, x, \langle\rangle\rangle$, and
   $tail(L) = \langle y, \langle\rangle\rangle$.

6. $cons(w, \langle x, y, z\rangle) = \langle w, x, y, z\rangle$.

7. $cons(a, \langle\rangle) = \langle a\rangle$.

8. $cons(all, \langle your, base\rangle) = \langle all, your, base\rangle$.

9. Consider list $L = \langle w, x, y, z \rangle$, deployed to implement a Stack ADT, where the head of L would correspond to the topmost position of the Stack. To implement method $push(el, L)$, what would be the result of

   (a) $cons(el, L)$?
   $cons(el, L)$ would produce $\langle el, w, x, y, z \rangle$.

   (b) $list(el, L)$?
   $list(el, L)$ would produce $\langle el, \langle w, x, y, z \rangle \rangle$.

   (b) $concat(list(el), L)$?
   $concat(list(el), L)$ would produce $\langle el, w, x, y, z \rangle$

10. Consider list $L = \langle w, x, y, z \rangle$, deployed to implement a Queue ADT, where the head of L would correspond to the front position of the Queue. To implement method $enqueue(el, L)$, what would be the result of

    (a) $cons(L, list(el))$? Is it acceptable? (i.e. Does it implement the Queue protocol?)

    $cons(L, list(el))$ would produce $\langle \langle w, x, y, z \rangle, el \rangle$.
    This is not an acceptable solution as it does not implement the Queue protocol.

    (b) $list(L, el)$? Is it acceptable? (i.e. Does it implement the Queue protocol?)

    $list(L, el)$ would produce $\langle \langle w, x, y, z \rangle, el \rangle$.
    This is not an acceptable solution as it does not implement the Queue protocol.

Problem 2

Consider list $\Lambda = \langle w, x, y, z \rangle$, deployed to implement a Queue Abstract Data Type.

1. Let the head of Λ correspond to the front position of the Queue. Implement operations `enqueue(el, Λ)` and `dequeue(Λ)` using list construction operations. In both cases we can refer to Λ' as the state of the list upon successful termination of one of its operations.

2. Let us now reverse the way we manipulate our data structure and let the head of Λ correspond to the rear of the Queue.

   (a) What would be the result of $cons(el, \Lambda)$, and would it be a correct implementation for operation `enqueue(el, Λ)`?

   (b) What would be the result of $list(el, \Lambda)$, and would it be a correct implementation for operation `enqueue(el, Λ)`?

   (c) What would be the result of $concat(list(el), \Lambda)$, and would it be a correct implementation for operation `enqueue(el, Λ)`?

Solution:

1. We **enqueue** at the REAR of the Queue. In this case, this corresponds to the END of the list:

   $\Lambda' = concat(\Lambda, list(el))$ (or: $\Lambda' = concat(\Lambda, cons(el, \langle\rangle))$)

   We **dequeue** at the FRONT of the Queue. In this case, this corresponds to the HEAD of the list: We extract the front element with $head(\Lambda)$, and we let $\Lambda' = tail(\Lambda)$.

2. We **enqueue** at the REAR of the Queue. In this case, this corresponds to the head of the list. The results of the expressions are as follows:

   (a) $cons(el, \Lambda) = \langle el, w, x, y, z \rangle$. This is a correct implementation of operation `enqueue`.

   (b) $list(el, \Lambda) = \langle el, \langle w, x, y, z \rangle \rangle$. This is **not** a correct implementation of operation `enqueue`.

(c) $concat(list(el), \Lambda) = \langle el, w, x, y, z \rangle$. This is a correct implementation of operation `enqueue`.

Problem 3

Consider a list Λ deployed to implement a Stack Abstract Data Type of elements of some generic type T.

1. Let the head of Λ correspond to the topmost position of the Stack. Implement the body of operations `push(el, Λ)` and `pop(Λ)` (let return element be held in variable *topmost*) using list construction operations. In both cases a) we assume that appropriate preconditions exist, and b) we can refer to Λ' as the state of the list upon successful termination of one of its operations.

2. Let the last element of Λ correspond to the topmost position of the Stack. Implement the body of both operations as above. When applicable, use control flow statements in your answer.

Solution:

1. When the head of Λ corresponds to the topmost position of the Stack, operation `push(el, Λ)` can be defined as

   $$\Lambda' = cons(el, \Lambda)$$

   and operation `pop(Λ)` can be defined as

   $$topmost = head(\Lambda), \Lambda' = tail(\Lambda)$$

2. When the last element of Λ corresponds to the topmost position of the Stack, operation `push(el, Λ)` can be defined as

   $$\Lambda' = concat(\Lambda, list(el))$$

and operation pop(Λ) can be defined as

```
pop (Λ): T is
    let Λ2 = ⟨⟩
    while (length(Λ) > 1) {
      Λ2 = concat(Λ2, list(head(Λ)))
      Λ' = tail(Λ)
    }
    let topmost = head(Λ)
    Λ' = Λ2
    return topmost
```

Problem 4

Consider the sets

- $Laptop = \{Apple, IBM, Sony, HP, Acer, Dell, LG\}$, and
- $Favorite = \{Apple, Sony, Dell\}$.

Answer the following questions:

1. How do we interpret the expression $Favorite : \mathbb{P}Laptop$?

   Answer: This is interpreted as "The variable *Favorite* can assume any value supported by the powerset of *Laptop*.

2. Is $\mathbb{P}laptop$ a legitimate type?

   Answer: Yes.

3. What is the nature of the variable in $Favorite : \mathbb{P}Laptop$? (i.e. Atomic or composite? If composite, what type?)

   Answer: Variable *Favorite* is a **composite**. It is in fact a set.

4. Is $Apple \in \mathbb{P}Laptop$? Explain why or why not.

   Answer: No. Variable *Apple* is atomic. It cannot be of a powerset type.

5. Is $\{Apple\} \in \mathbb{P}Laptop$? Explain why or why not.

   Answer: Yes. Variable $\{Apple\}$ is a set. It is a legitimate element of the powerset of *Laptop*.

6. Is $\{\{\}\} \in \mathbb{P}Laptop$?

   Answer: No.

7. Is $\{\} \in \mathbb{P}Laptop$? Explain why or why not.

   Answer: Yes. The empty set is a subset of any set.

8. If we define variable $Favorite : \mathbb{P}Laptop$, is $\{\}$ a legitimate value for variable *Favorite*? Explain why or why not.

   Answer: Yes. The empty set is a subset of any set

9. Is $Favorite \in \mathbb{P}Laptop$? Explain.

   Answer: Yes. Set *Favorite* is an element of the powerset of *Laptop*.

10. Is $Favorite \subset \mathbb{P}Laptop$? Explain.

    Answer: No. The powerset is a set of sets. Set *Favorite* does not contain any sets as its elements.

Problem 5

Consider the following two sets:

- $OS = \{MacOS, Linux, BSD, Windows, Unix\}$, and
- $My\_OS = \{BSD, Unix\}$.

1. Is the following declaration acceptable: $My\_OS : \mathbb{P}OS$? Explain.

   Answer: Yes, because this is interpreted as "The variable My_OS can assume any value supported by the powerset of OS. In other words, any subset we can produce from OS can be a legitimate value of any variable of type My_OS.

2. Is $\mathbb{P}OS$ a legitimate type? Explain.

   Answer: Yes. It provides a description of the values that can be held by an element (variable).

3. What does the following statement signify? $My\_OS : OS$. Is the statement acceptable? Explain.

   Answer: This is a variable declaration statement. The statement is not acceptable, because My_OS is a set and it cannot assume any value supported by OS.

4. Is $MacOS \in \mathbb{P}OS$? Explain.

   Answer: No, because variable $MacOS$ is atomic. It cannot hold a set. Elements of $\mathbb{P}OS$ are all sets.

5. Is OS a legitimate type?

   Answer: Yes, because OS represents a set of values.

6. Is $\{\} \in \mathbb{P}OS$? Explain.

   Answer: Yes. The empty set is a subset of any set and thus it is included as an element in $\mathbb{P}OS$.

7. Is $\{Linux, BSD\} \in \mathbb{P}OS$? Explain.

   Answer: Yes, because $\{Linux, BSD\}$ is a set. It is a legitimate element of the powerset of OS.

8. Is $\{\{\}\} \in \mathbb{P}OS$?

   Answer: No. There is no element in $\mathbb{P}OS$ that contains the empty set.

9. Is $\{\} \in OS$? Explain.

   Answer: No. There is no element $\{\}$ in OS.

10. If we define variable $My\_computer : \mathbb{P}OS$, is $\{\}$ a legitimate value for variable $My\_Computer$? Explain.

    Answer: Yes, because the empty set is a subset of any set.

11. If we stated that $My\_computer = \{Windows\}$, would the statement make $My\_Computer$ an atomic variable?

    Answer: No, because $My\_Computer$ is still a set. It can hold any value supported by $\mathbb{P}OS$.

12. Is $\{\{BSD, MacOS\}\} \subset \mathbb{P}OS$? Explain.

    Answer: Yes, since the $\{\{BSD, MacOS\}\}$ is a set of sets.

13. Is $My\_OS \subset \mathbb{P}OS$? Explain.

    Answer: No, because the powerset is a set of sets. Any subset of $\mathbb{P}OS$ would have to be a set of sets. Variable My_OS is a set of atomic elements.

14. Is $\{\{BSD, MacOS\}\} \in \mathbb{P}OS$?

    Answer: No.

Problem 6

Consider the following two sets:

- $Car = \{bmw, audi, mercedes, jeep, infiniti, lexus, volkswagen\}$, and
- $Favorite = \{bmw, infiniti, jeep\}$.

Answer the following questions:

1. Is $\mathbb{P}Car$ a legitimate type?

   Answer: Yes.

2. Is the following declaration acceptable: $Favorite : Car$? Explain.

   Answer: No, because *Favorite* is a set and it cannot assume any value supported by Car.

3. Is the following declaration acceptable: $Favorite : \mathbb{P}Car$? Explain.

   Answer: Yes, because this is interpreted as "The variable *Favorite* can assume any value supported by the powerset of Car. In other words, any subset we can produce from Car can be a legitimate value for variable *Favorite*.

4. Is $audi \in \mathbb{P}Car$? Explain.

   Answer: No, because variable *audi* is atomic. It cannot hold a set.

5. Is $\{infiniti\} \in \mathbb{P}Car$? Explain.

   Answer: Yes, because $\{infiniti\}$ is a set. It is a legitimate element of the powerset of Car.

6. Is $\{\{\}\} \in \mathbb{P}Car$?

   Answer: No.

7. Is $\{\} \in \mathbb{P}Car$? Explain.

   Answer: Yes. The empty set is a subset of any set and thus it is included as an element in $\mathbb{P}Car$.

8. If we define variable $Favorite : \mathbb{P}Car$, is $\{\}$ a legitimate value for variable *Favorite*? Explain.

   Answer: Yes, because the empty set is a subset of any set.

9. Is $Favorite \subset \mathbb{P}Car$? Explain.

   Answer: No, because the powerset is a set of sets. Any subset of $\mathbb{P}Car$ would have to be a set of sets. Variable *Favorite* is a set of atomic elements.

10. Is $\{Favorite\} \subset \mathbb{P}Car$? Explain.

    Answer: Yes, since now the LHS is a set of sets.

Problem 7

(NOTE: For this problem you will need to refer to *Tutorial Notes on Construction Techniques*, Problem 5 for the implementation of functions `f`, and `g`.)

Consider the Queue ADT implemented over a List data structure, where the head of the list corresponds to the rear of the Queue.

1. Define operations *Enqueue* : $lists(\mathbb{T}) \times \mathbb{T} \rightarrow lists(\mathbb{T})$, and *Dequeue* : $lists(\mathbb{T}) \rightarrow \mathbb{T}$.

   Answer:

   ```
   enqueue(Λ, el) is
     Λ′ = cons(el, Λ)

   dequeue(Λ) is
     if not(is_empty(Λ))
       el = g(Λ)
       Λ′ = f(Λ)
     return el
   ```

2. Use Common LISP to define a global variable `queue` to hold the collection, and implement functions `enqueue` and `dequeue`. Place your code in file `queue-adt.lisp`. In your main submission file, include your interaction with the system while demonstrating the functionality of your code.

   Answer:

   ```
   (defparameter queue nil)


   (defun enqueue (el)
     (setf queue (cons el queue)))


   (defun dequeue ()
   ```

```
(let ((result (g queue)))    ;; obtain last element
  (setf queue (f queue))    ;; update collection
  result))                  ;; return last element
```

A sample interaction is shown below:

```
> queue
NIL
> (enqueue 'a)
(A)
> (enqueue 'b)
(B A)
> (enqueue 'c)
(C B A)
> (dequeue)
A
> queue
(C B)
```

Problem 8

Consider the Queue Abstract Data Type (ADT), $\mathbb{Q}$, defined over some generic type $\mathbb{T}$ and defined using two stacks, Σ_1 and Σ_2, where a Stack ADT is implemented as a list, Λ. (*Hint*: This description implies that the only available operations are those defined for a list.) Define operations *Enqueue*$(\mathbb{Q}, \mathbb{T})$ and *Dequeue*$(\mathbb{Q})$.

Answer:

```
enqueue(ℚ, el: 𝕋) is
  Σ1' = cons(el, Σ1)

dequeue(ℚ): T is
  if not(is_empty(Σ2))
    result = head(Σ2)
    Σ2' = tail(Σ2)
  else
    while not(is_empty(Σ1))
    push(Σ2, head(Σ1))
    Σ1' = tail(Σ1)
    result = head(Σ2)
    Σ2' = tail(Σ2)
  return result
```

Problem 9

Implement operations *head*, *tail* and *cons* in Prolog and demonstrate their usage (include your interaction with the language environment in your submission). Consider the following example executions:

```
?- head([a, b, c, d], H).
H = a .
?- head([x], H).
H = x .
?- head([], H).
false.
?- tail([a, b, c, d], T).
T = [b, c, d] .
?- tail([x], T).
T = [] .
?- tail([], T).
false.
?- cons(a, [b, c], NewList).
NewList = [a, b, c] .
?- cons(a, [], NewList).
NewList = [a] .
?- cons([], [], NewList).
NewList = [[]] .
```

Answer:

```
head([X|L], X).
tail([X|L], L).
cons(X, L, [X|L]).
```