SOEN 331 (Section S):Formal Methods for Software Engineering

Assignment 1

Etienne Pham Do 40130483

Hamzah Muhammad 40156621

October 9, 2022

4.1. 1. $\forall\, a\, \exists\, b\;\; received\_request(a, b)$ (let this expression be A) translates to: "everyone received a request from someone".

$\exists\, b\, \forall\, a\;\; received\_request(a, b)$ (let this expression be B) translates to: "there is someone who sent a request to everyone".

For the 2 expressions to be logically equivalent, the biconditional of the 2 expressions must yield true. The conditional $B \rightarrow A$ must return true, which is the case. However, $A \rightarrow B$ is not true as it isn't necessarily true that there is someone who sent a request to everyone if everyone received a request from someone. Therefore, A and B are not logically equivalent.

2. The statement $\forall\, a\, \exists\, b\;\; received\_request(a, b) \rightarrow \exists\, b\, \forall\, a\;\; received\_request(a, b)$ is false as there is a possibility that there isn't a person who sent a request to everyone.

3. The statement $\exists\, b\, \forall\, a\;\; received\_request(a, b) \rightarrow \forall\, a\, \exists\, b\;\; received\_request(a, b)$ is true because if there is a person that sent a request to everyone then it must be true that everyone has received a request from someone.

4. $\forall\, b\, \exists\, a\;\; received\_request(a, b)$ (let this expression be C) translates to: "everyone has sent a request to someone".

$\exists\, a\, \forall\, b\;\; received\_request(a, b)$ (let this expression be D) translates to: "there is someone who has received a request from everyone".

The two expressions are not logically equivalent.

4.2. 1. "There are some nice people" translates to "Some people are not bad.", which is type O.

2. "There are no nice people" translates to "All people are bad", which is type A.

3. "Everybody is bad" translates to "All people are bad", which is type A.

4. "Some people are bad" is type I.

5. "Everybody is nice" translates to "No people are bad", which is type E.

6. "Some people are not nice" translates to "Some people are bad", which is type I.

4.3. 1. The declaration My_OS : $\mathbb{P}$OS is acceptable because it means that the variable My_OS can represent any value within the power set of OS. In this case, My_OS contains {BSD, Unix}, which is a subset of the power set that contains all possible subsets of OS.

2. The declaration $\mathbb{P}$OS is a legitimate type as it is simply a domain for any variable that represents a set (the power set of OS is the set of all subsets of OS).

3. The declaration My_OS : OS entails that the variable My_OS can represent any value within the set OS. This statement is not acceptable as the value of My_OS is a set and the set OS does not contain sets, only atomic elements.

4. $MacOS \notin \mathbb{P}OS$ because $\mathbb{P}OS$ is a set, where its elements are sets, whereas MacOS is not - it is simply an atomic variable.

5. OS, which is a set of atomic elements, is a legitimate type as it is simply a domain for any atomic variable.

6. $\{\} \in \mathbb{P}OS$ because $\mathbb{P}OS$ is a set that contains all subsets of OS, including the empty set.

7. $\{Linux, BSD\} \in \mathbb{P}OS$ because $\mathbb{P}OS$ is a set that contains all subsets of OS, including the set {Linux, BSD}.

8. $\{\{\}\} \notin \mathbb{P}OS$ because the set containing the empty set is not a part of all the subsets of OS.

9. $\{\} \notin OS$ because the empty set is not an element in OS.

10. {} is a legitimate value for variable My_Computer because the empty set is a subset of the power set of OS.

11. Yes.

12. $\{\{BSD, MacOS\}\}$ is a subset of $\mathbb{P}OS$ because $\mathbb{P}OS$ is a set that contains all subsets of OS and $\{\{BSD, MacOS\}\}$ is a set containing one of those subsets .

13. My_OS is not a subset of $\mathbb{P}OS$ because the powerset is a set of sets, therefore any subset of $\mathbb{P}OS$ must have to be a set of sets and variable My_OS is a set of atomic elements.

14. $\{\{BSD, MacOS\}\} \notin \mathbb{P}OS$ because the set containing the set {BSD, MacOS} is not a part of all the subsets of OS.

4.4. 1. map is a binary relation because the set of Flight is only mapped to another set: Airline, with the relation "is associated to". In other words, map can be described as a set of ordered pairs.

2. map is a function because the set Flight is the domain, which is mapped to the set Airline that represents the codomain. The type of the function is surjective.

3. The precondition for operation add is as follows: $flight \notin domairline$.

4. $airline' = airline \ \cup \{flight? \mapsto airline?\}$ or $airline' = airline \ \oplus \{flight? \mapsto airline?\}$

5. The result of calling that operation would be precondition failure and the pair will not be added to the relation.

6. The result of calling that operation would be a violation of the requirements (R2: Each flight is associated to a single airline, e.g. AA333 is an American Airlines flight). Thus, the call should be rejected and no pair is added to the relation.

7. It cam only be used if no record already exists whose first coordinate is identical to the corresponding one of the input. Otherwise, another record will be added with the same first coordinate causing a violation of the requirements.

8. The definition for the core functionality of operation delete is $map' = map \rhd \{flight?\}$

4.5. 1. $\{A330, 747\} \lhd airplanes \ = \{A330 \mapsto Airbus, 747 \mapsto Boeing\}$

2. $airplanes \rhd \{Comac, Embraer\} \ = \{C919 \mapsto Comac, E170 \mapsto Embraer, E175 \mapsto Embraer\}$

3. $\{A320, A330, A350, E170\} \ntriangleleft airplanes \ = \{A380 \mapsto Airbus, 737 \mapsto Boeing, 747 \mapsto Boeing, Superjet100 \mapsto Sukhoi, C919 \mapsto Comac, Global7500 \mapsto Bombardier, Global8000 \mapsto Bombardier, E175 \mapsto Embraer\}$

4. $airplanes \ntriangleright \{Airbus, Boeing\} \ = \{Superjet100 \mapsto Sukhoi, C919 \mapsto Comac, Global7500 \mapsto Bombardier, Global8000 \mapsto Bombardier, E170 \mapsto Embraer, E175 \mapsto Embraer\}$

5. $airplanes \oplus \{Su\_80 \mapsto Sukhoi\} = \{A320 \mapsto Airbus, A330 \mapsto Airbus, A350 \mapsto Airbus, A380 \mapsto Airbus, 737 \mapsto Boeing, 747 \mapsto Boeing, Su\_80 \mapsto Sukhoi, Superjet100 \mapsto Sukhoi, C919 \mapsto Comac, Global7500 \mapsto Bombardier, Global8000 \mapsto Bombardier, E170 \mapsto Embraer, E175 \mapsto Embraer\}$