SOEN 331 (S):Formal Methods for Software Engineering

Assignment 1

Richard Badir

Valentin Gornostaev

March 1, 2024

Problem 1

To check if the proposition is true, we need to turn over all blue cards, and all number cards that do not have a prime number. This is because we need to check if blue cards have indeed a prime number on the other side and we need to check if number cards that do not have a prime number also do not have blue on the other side. Therefore, we need to turn over the card with the number 9, because 9 is not prime, we do not need to turn over the card with the number 11, because 11 is prime, we need to turn over the blue card, because it is blue, and we do not need to turn over the yellow card, because it is not blue.

Problem 2

part 1

1. x is an object, Planet(x) denotes that x is a planet.

   MassGreaterThan(x,m) denotes the mass of m is greater than m.

   OrbitsSun(x) denotes that x orbits the Sun.

   The formla for a planet in predicate logic becomes:

   $\forall x((\text{MassGreaterThan}(x, 0.33) \land \text{OrbitsSun}(x)) \rightarrow \text{Planet}(x))$

   Using the formula for planet we can construct the binary relation to find the which ones are satellites.

   SatelliteOf(x,y) denotes that object x is the satellite of object y

   Orbits(x,y) denotes that x orbits around y

   Planet(y) denotes that y is a planet (using the previous predicate logic)

   The formula in predicate logic is:

   $\forall x \forall y(Orbits(x, y) \land Planet(y)) \rightarrow (SatelliteOf(x, y))$

2. `is_planet(X) :- mass(X, Mass), Mass > 0.33, orbits(X, sun).`

   `is_satellite_of(X, Y) :- orbits(X, Y), is_planet(Y).`

   Code sample and examples included in the solar.pl file

3. `obtain_all_satellites(Object,List) :- is_planet(Object), findall(S, orbits(S, Object) , List).`

   Code sample with examples included in solar.pl file

part 2

1. Some numbers are not composite (Particular Negative):

$$\exists\, x(\text{number}(x) \land \neg\text{composite}(x))$$

2. No numbers are prime (Universal Affirmative):

$$\forall\, x(\text{number}(x) \rightarrow \text{composite}(x))$$

3. Some numbers are not prime (Particular Affirmative):

$$\exists\, x(\text{number}(x) \land \text{composite}(x))$$

4. All numbers are prime (Universal Negative):

$$\forall\, x(\text{number}(x) \rightarrow \neg\text{composite}(x))$$

part 3

1. The process of negating a statement corresponds to negating the predicate and changing the quantifier (de Morgan's Law, applicable in vice versa). Therefore, negating $(\forall\, s : S \mid s \in P)$ will negate the predicate and change the quantifier, which results in $\exists\, s : S \mid \neg(s \in P)$. This become particular $(\exists\, s)$ negative $\neg(s \in P)$.

2. The type E proposition is $\forall\, s : S \mid \neg(s \in P)$. By negating it, we apply De Morgan's law where we convert the universtal statement into an existential one and we negate the predicate (applicable in vice versa). this results in obtaining I since the statement

is now existential ($\exists\, s$) and the predicate is now double negative which becomes a positive ($s \in P$).

Problem 4

1. $\mathbb{P}$*Languages* is the power set of the *Languages* set. That means that it is the set comprised of all the possible combinations of items inside *Languages*. These items are combined into sets themselves.

2. (a) This expression signifies that *Favorite* is of type $\mathbb{P}$*Languages*.

   (b) It should be interpreted as saying that the variable *Favorite* can have any value in the powerset of *Languages*, and can not have any value that is not in that powerset.

   (c) All of the values in the powerset of *Languages* are legitimate for *Favorite*.

3. (a) This expression signifies that *Favorite* holds the value of the powerset of *Languages*.

   (b) These 2 expressions are not semantically equivalent $\mathit{Favorites} = \mathbb{P}\mathit{Languages}$ means that the variable *Favorite* holds the value of the powerset of *Languages*. $\mathit{Favorites} : \mathbb{P}\mathit{Languages}$ means that the variable *Favorites* can hold any value in the powerset of *Languages*, but not the powerset itself.

4. No, $\{\mathit{Lua}, \mathit{Groovy}, C\} \notin \mathbb{P}\mathit{Languages}$. $\{\mathit{Lua}, \mathit{Groovy}, C\}$ is not an element of $\mathbb{P}$*Languages*. This is because $\{\mathit{Lua}, \mathit{Groovy}, C\}$ is an element of *Languages*, so the set of $\{\mathit{Lua}, \mathit{Groovy}, C\}$ (meaning $\{\{\mathit{Lua}, \mathit{Groovy}, C\}\}$) would be an element of $\mathbb{P}$*Languages*. The set containing each element in *Languages* is in $\mathbb{P}$*Languages*, but not the elements themselves.

5. No, $\{\{\mathit{Lua}, \mathit{Groovy}, C\}\} \not\subset \mathbb{P}\mathit{Languages}$. This is because a subset is a set containing elements of a larger set. In this case $\{\{\mathit{Lua}, \mathit{Groovy}, C\}\}$ is itself an element of $\mathbb{P}$*Languages*. The set $\{\{\{\mathit{Lua}, \mathit{Groovy}, C\}\}\}$ would be a subset of $\mathbb{P}$*Languages*.

6. λ_1 is of type Languages, so it can hold any value in Languages. λ_2 is of type $\mathbb{P}$*Languages*, so it can hold any value in the powerset of *Languages*. λ_1 is atomic and λ_2 is non-atomic.

7. *Library* is of type $\mathbb{P}Languages$.

8. No, $\{\varnothing\} \notin \mathbb{P}Languages$. This is because the empty set $\varnothing$ is always part of the powerset of any set, but the set containing the empty set is not, unless the empty set is part of the original set, which is not the case in this scenario ($\varnothing \notin Languages$).

9.
```
CL-USER 1 > (defun is-memberp(element set)
(if (member element set) t nil))
IS-MEMBERP

CL-USER 2 > (is-memberp 'a '(b c d a))
T

CL-USER 3 > (is-memberp 'e '(b c d a))
NIL

CL-USER 4 > (defun equal-setsp(set1 set2)
(and (subsetp set1 set2)
(subsetp set2 set1)))
EQUAL-SETSP

CL-USER 5 > (equal-setsp '(b c d a) '(a b c d))
T

CL-USER 6 > (equal-setsp '(b c d a) '(b c d a 2))
NIL
```

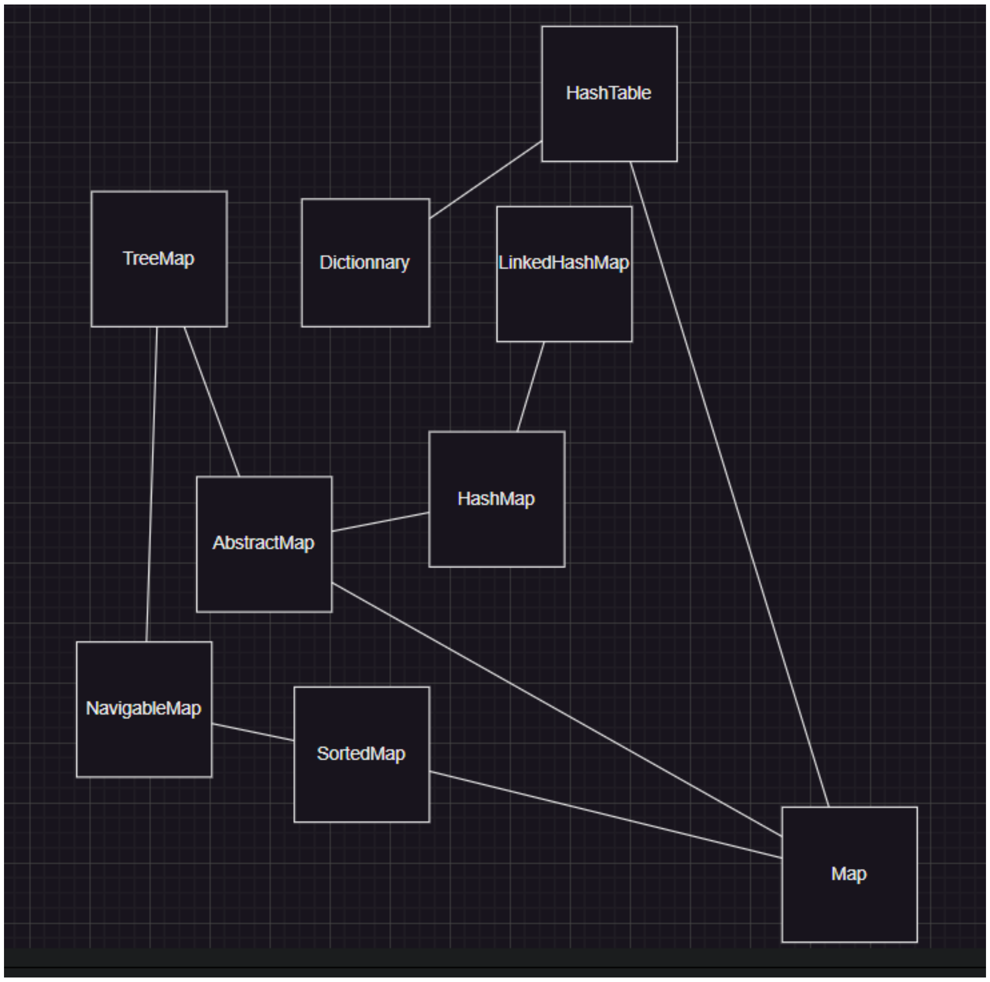
Problem 6

Part 1

1. In java, the "is of type" relation is of partial order. If a variable is of a certain type, then it is obviously itself of that type, making this a reflexive relation. This is an antisymmetric relation, because if a variable V_1 is of a parent type to another variable V_2, V_2 can not be a parent type of V_1. The Java API does not support cyclic inheritance. This relation is transitive, because if type A is the parent tpye of type B, and type B is the parent type of type C, then C automatically inherits from A. A type that inherits from a second type is also of the second type. Any relation that is reflexive, antisymmetric, and transitive is of partial order, therefore the "is of type" relation for the Java API is of partial order.

2. In this relation, we can see from the set of edges that there are no bidirectional edges, and there are no cycles between edges. Therefore, the relation is antisymmetric. In this relation, the vertices and edges represent graphical units in a Hasse diagram. From this information it can be assumed that each element is reflexive on itself, and that the xRx holds true for any x in V_1. For example , if R is simply "is a" then it can easily be deduced that xRx holds true. Also from the Hasse diagram, it can be assumed that any path from vertice x to vertice z, passing by vertice y, also indicates that xRz holds true. Therefore this reflexive, antisymmetric, and transitive relation is of partial order and so (V_1, R) is a poset.

3. Minimums are *HashTable*, *TreeMap*, and *LinkedHashMap*. Maximums are *Dictionnary* and *Map*.



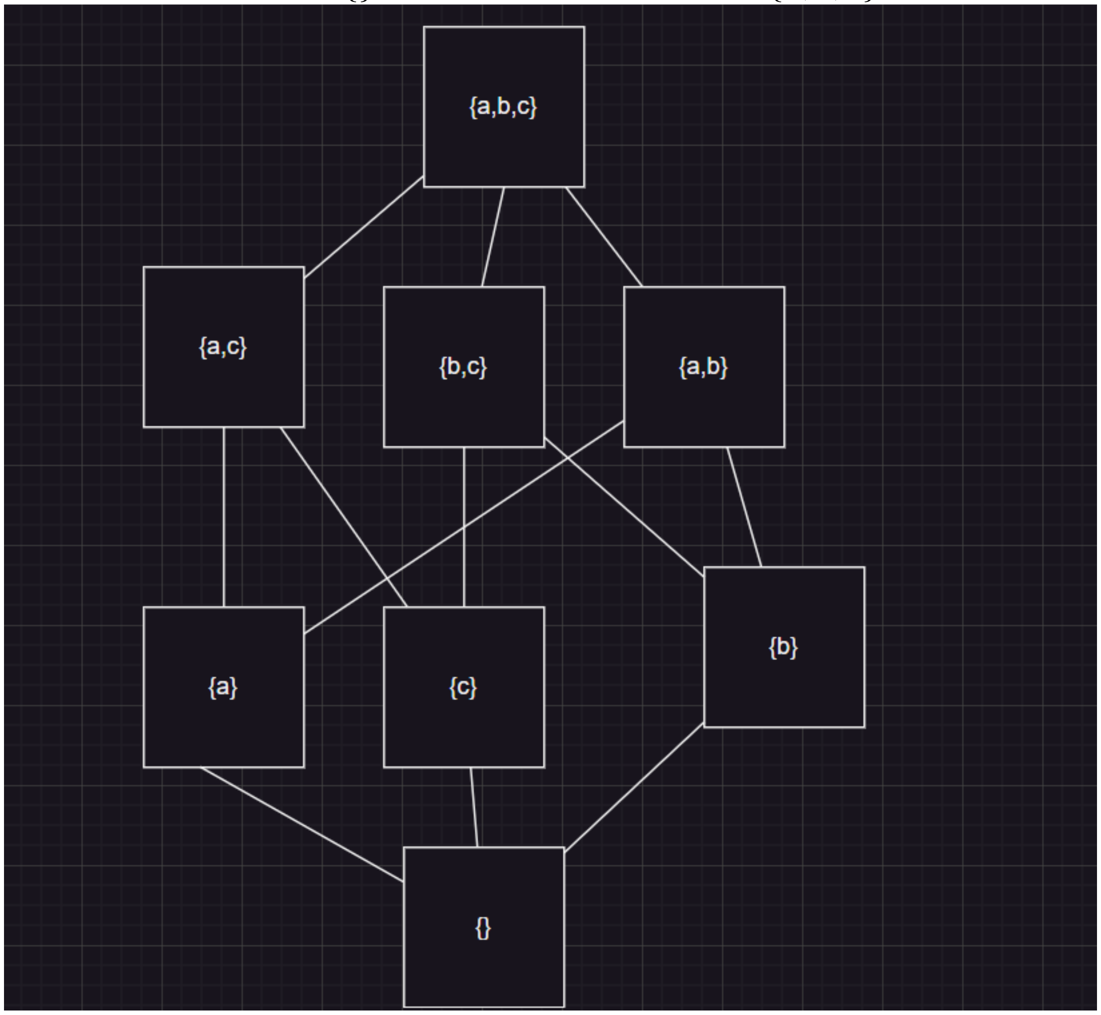
Part 2

1. $\subseteq$ is reflexive, because any set is a subset of itself. It is antisymmetric, because if a set is a subset of another set, it the reverse can not be true, unless both sets are the same. The set that is a subset would be fully comprised of some of the elements of the other set, it is impossible that that other set is itself fully comprised of some of the elements of its own subset, unless the two sets are equivalent. This relation is transitive, because for any set B that is a subset of set A, if set C is a subset of set B, then set C is automatically a subset of set A. Therefore, because it is reflexive, antisymmetric and transitive, $\subseteq$is a relation of partial order.

2. $\mathbb{P}V_2 = \{\{\}, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{a, b, c\}, \{b, c\}\}$. Every set is a subset of itself,

so this relation is reflexive. Every set A that is a subset of another set B can not be the subset of that set B if B is not equal to A, so this relation is antisymmetric. This relation is transitive, because if a set B is a subset of a set A, and the set C is a subset of the set B, then set C is automatically a subset of A. Therefore $(\mathbb{P}V_2, \subseteq)$ creates a poset.

3. The maximal element is $\{\}$ and the minimal element is $\{a, b, c\}$.



Part 3

For any element in the domain of map, there is an element in the codomain to which it maps, therefore it is a function. $\forall x \in domain(map), \exists y \in codomain(map), suchthatmap(x) \mapsto y$. *map* is a function, but it is a partial function, because it maps only a subset of (V_1, R). $(V_1, R) \not\mapsto (V_2, \subseteq)$. *map* is not an injective function, because a,b,c is mapped to by more than one element in the domain. $\exists\, a, b(a \not= b \mapsto map(a) = map(b))$. This function is surjective, because every element in the codomain of map is mapped to by at least one element from the domain. $b \in codomain(map), a \in domain(map), map(a) = b, \forall\, b\, \exists\, a(map(a) = b)$. Since it is not injective, map is not bijective. $\exists\, a, b(a \not= b \mapsto map(a) = map(b))$. map is not order preserving because $AbstractMap \succ TreeMap$, but $map(AbstractMap) \not\succ map(TreeMap)$. $\exists\, x \succ y, map(x) \not\succ mapy(y)$. It is neither order reflecting, because $\varnothing \succ \{c\}$, but $TreeMap \not\succ AbstractMap$. $\exists\, map(x) \succ map(y), x \not\succ y$. map is not order preserving nor order reflexing, so it can not be order embedding. $\exists\, map(x) \succ map(y), x \not\succ y$ and $\exists\, x \succ y, map(x) \not\succ mapy(y)$. map is not order embedding, so it can not be isomorphic. $\exists\, map(x) \succ map(y), x \not\succ y$ and $\exists\, x \succ y, map(x) \not\succ mapy(y)$.

Problem 7

1.
   ```
   compress(List) is
   List2 = ⟨⟩
   while List != ⟨⟩
   concat(List2, ⟨head(List)⟩)
   if (head(List) = head(tail(List)))) then
   x = head(List)
   while (x = head((List))
   List' = tail((List))
   else
   List' = tail(List)

   return List2
   ```

2.
   ```
   compress(List) is
   if List = ⟨⟩ then
   return List
   if (head(List) = head(tail(List)) then
   ```

```
List' = compress(tail(List))
else
List' = cons(head(List), compress(tail(List)))
return List
```

$compress(<a, a, b, b, c, a>) = compress(<a, b, b, c, a>)$

$= cons(a, compress(<b, b, c, a>))$

$= cons(a, compress(<b, c, a>))$

$= cons(a, cons(b, compress(<c, a>)))$

$= cons(a, cons(b, cons(c, compress<a>)))$

$= cons(a, cons(b, cons(c, <a>))$

$= cons(a, cons(b, <c, a>))$

$= cons(a, <b, c, a>)$

$= <a, b, c, a>$

4.
```
CL-USER 1 > (defun compress (list)
(if (null (rest list))
list
(if (equal (first list) (first (rest list)))
(compress (rest list))
(cons (first list) (compress (rest list))))))
COMPRESS


CL-USER 2 > (compress '(a a a b c d d e))
(A B C D E)


CL-USER 3 > (compress '())
NIL


CL-USER 4 > (compress '(a b c c c a))
```

```
(A B C A)


CL-USER 5 > (compress '(a a b b c c c c a b b))
(A B C A B)


CL-USER 6 >
```

———- Remove everything after and including this line after project is completed.

1. Here is some code in Prolog

```
member(X, [X|_]).
member(X, [_|T]) :- member(X, T).
```

2. LTL formulas:

   (a) $\Box\phi \rightarrow \Diamond\psi$

   If ϕ is an invariant, then ψ becomes true eventually.

   (b) $\Box\phi \rightarrow \bigcirc\Box\Diamond\psi$

   If ϕ is an invariant, the ψ will be true infinitely often, starting from the next moment.

   (c) $(\phi \wedge \bigcirc\psi) \rightarrow \Diamond\Box\tau$

   If *phi* is true at time $= i$ and ψ is true at time $= i+1$, then eventually τ becomes true and stays true.

   (d) $\Box((\psi \wedge \bigcirc\chi) \rightarrow \bigcirc\tau)$

   It is always the case that if ψ is true at time $= i$ and if χ is true at time $= i+1$, then τ is true at time $= i+1$.

   (e) $(\chi \wedge \bigcirc\omega) \rightarrow \bigcirc^2(\phi\,\mathcal{U}\,\psi)$

   If χ is true at time $= i$ and if ω is true at time $= i+1$, then at time $= i+2$ ϕ becomes true and stays true until ψ becomes true.

(f) $(\phi \oplus \psi) \rightarrow \Box\omega$

If one of ϕ or ψ are true at time $= i$, then ω becomes at invariant at time $= i$.

3. The behavior of a program is expressed by the following temporal formula:

$$\Box \left[\begin{array}{c} 1.\ \textbf{start} \rightarrow \neg a \vee \neg b \\ \\ 2.\ \textbf{start} \rightarrow c \\ \\ 3.\ b \wedge c \rightarrow \bigcirc^2(d \oplus e) \\ \\ 4.\ a \vee c \rightarrow \bigcirc(k\,\mathcal{R}\,g) \\ \\ 5.\ (d \vee e) \rightarrow \bigcirc^2 k \\ \\ 6.\ c \rightarrow (h\,\mathcal{W}\,(e \wedge g)) \\ \\ 7.\ (d \wedge g \wedge h) \rightarrow \bigcirc^3 m \\ \\ 8.\ d \rightarrow m\,\mathcal{R}\,h \\ \\ 9.\ e \wedge \bigcirc^2(k \wedge g) \rightarrow \bigcirc^2 m \\ \\ 10.\ (e \wedge g) \rightarrow \bigcirc^3 c \\ \\ 11.\ k \wedge m \rightarrow \bigcirc h \\ \\ 12.\ (e \wedge \bigcirc^2 k) \rightarrow \bigcirc^3 b \end{array} \right]$$

4. Let $B(x)$ denote the subject "x is a bird" and $W(x)$ denote the predicate "x is white." Translate the following formal statements into English sentences and attach the corre-

sponding categorical form to each:

(a) $\forall x\,(B(x) \to W(x))$

(b) $\forall x\,(B(x) \to \neg W(x))$

(c) $\exists x\,(B(x) \wedge W(x))$

(d) $\exists x\,(B(x) \wedge \neg W(x))$

Solution:

(a) $\forall x\,(B(x) \to W(x))$: "All birds are white.": (A)

(b) $\forall x\,(B(x) \to \neg W(x))$: "All birds are non-white.": (E)

(c) $\exists x\,(B(x) \wedge W(x))$: "Some birds are white.": (I)

(d) $\exists x\,(B(x) \wedge \neg W(x))$: "There is an non-white bird.": (O)

5. Describe when the following predicate can be false: $\forall x \exists y P(x, y)$.

   Solution: The statement can be false when there is an x such that $P(x, y)$ is false for every y.

6. Let $P(x, y)$ be the statement "x asked y out to lunch" where the domain is all students in class. Express each of the following quantifications in English:

   (a) $\exists y \forall x P(x, y)$.

   Solution: Recall that this reads **"There in an y that makes $P(x, y)$ true for every x."** There is a student in class who has been asked out to lunch by every student in class.

   (b) $\forall x \exists y P(x, y)$.

Solution: Recall that this reads **"For every x there is a y for which $P(x, y)$ is true."** Every student in class has asked out to lunch some (at least one) student in class.

An important observation on functions is that we can view them as relations, and as such we can model a function as a set of pairs (tuples).

1. Consider the following relation:

$$phone : Model \leftrightarrow Brand$$

where

$$phone =$$
$$\{$$
$$iPhone7 \mapsto apple,$$
$$iPhoneX \mapsto apple,$$
$$galaxyS \mapsto samsung,$$
$$galaxyA \mapsto samsung,$$
$$galaxyJ \mapsto samsung,$$
$$mate20 \mapsto huawei,$$
$$p20 \mapsto huawei$$
$$\}$$

(a) What is the domain of the relation?

$$\text{dom } phone = \{iPhone7, iPhoneX, galaxyS, galaxyA, galaxyJ, mate20, p20\}.$$

(b) What is the range of the relation?

ran $\mathit{phone} = \{\mathit{apple}, \mathit{samsung}, \mathit{huawei}\}$.

(c) What is the result of the expression $\{\mathit{iPhone}7, \mathit{galaxyA}\} \lhd \mathit{phone}$?

Domain restriction selects pairs based on their first element. As a result,

$$\{\mathit{iPhone}7, \mathit{galaxyA}\} \lhd \mathit{phone} = \{\mathit{iPhone}7 \mapsto \mathit{apple}, \mathit{galaxyA} \mapsto \mathit{samsung}\}$$

Restriction operators are deployed to model database *queries*.

(d) What is the result of the expression $\mathit{phone} \rhd \{\mathit{apple}, \mathit{samsung}\}$?

Range restriction selects pairs based on their second element. As a result,

$$\begin{array}{l} \mathit{phone} \rhd \{\mathit{apple}, \mathit{samsung}\} = \\ \quad \{ \\ \qquad \mathit{iPhone}7 \mapsto \mathit{apple}, \\ \qquad \mathit{iPhoneX} \mapsto \mathit{apple}, \\ \qquad \mathit{galaxyS} \mapsto \mathit{samsung}, \\ \qquad \mathit{galaxyA} \mapsto \mathit{samsung}, \\ \qquad \mathit{galaxyJ} \mapsto \mathit{samsung} \\ \quad \} \end{array}$$

Consider the following Questions to be done in one sequence where we will make permanent modifications to the contents of *phone*:

(e) What is the result of the expression $\{iPhone7, iPhoneX, galaxyA, mate20\} \lhd phone$?

Domain subtraction removes elements from the domain of the relation:

$$
\begin{aligned}
&\{iPhone7, iPhoneX, galaxyA, mate20\} \lhd phone = \\
&\quad \{ \\
&\qquad galaxyS \mapsto samsung, \\
&\qquad galaxyJ \mapsto samsung, \\
&\qquad p20 \mapsto huawei \\
&\quad \}
\end{aligned}
$$

Note that for a modification to phone, we need to write

$$phone' = \{iPhone7, iPhoneX, galaxyA, mate20\} \lhd phone$$

which reads as: "The new value of phone is assigned the value of the evaluation of the expression on the right-hand-side."

(f) What is the result of the expression $phone \rhd \{huawei\}$?

Range subtraction removes elements from the codomain of the relation:

$$
\begin{aligned}
&\mathit{phone} \rhd \{\mathit{huawei}\} = \\
&\quad \{ \\
&\qquad \mathit{galaxyS} \mapsto \mathit{samsung}, \\
&\qquad \mathit{galaxyJ} \mapsto \mathit{samsung} \\
&\quad \}
\end{aligned}
$$

Assume now that we did

$$\mathit{phone}' = \mathit{phone} \rhd \{\mathit{huawei}\}$$

and as a result, the new value of *phone* will be

$$
\begin{aligned}
&\mathit{phone} = \\
&\quad \{ \\
&\qquad \mathit{galaxyS} \mapsto \mathit{samsung}, \\
&\qquad \mathit{galaxyJ} \mapsto \mathit{samsung} \\
&\quad \}
\end{aligned}
$$

(g) What is the result of $\mathit{phone} \oplus \{\mathit{iPhoneXSMax} \mapsto \mathit{apple}\}$?

Relational overriding can model database updates.

$$
\begin{aligned}
&\mathit{phone} \oplus \{\mathit{iPhoneXSMax} \mapsto \mathit{apple}\} = \\
&\quad \{ \\
&\qquad \mathit{iPhoneXSMax} \mapsto \mathit{apple}, \\
&\qquad \mathit{galaxyS} \mapsto \mathit{samsung}, \\
&\qquad \mathit{galaxyJ} \mapsto \mathit{samsung} \\
&\quad \}
\end{aligned}
$$

Note that for a modification to *phone*, we need to write

$$phone' = phone \oplus \{iPhoneXSMax \mapsto apple\}$$

2. Consider the sets

   - $Phone = \{Samsung, Huawei, Apple, Sony, Motorola, HTC\}$, and
   - $Favorite = \{Sony, HTC\}$.

   Answer the following questions:

   (a) How do we interpret the expression $Favorite : \mathbb{P}Phone$?

   (b) Is $\mathbb{P}Phone$ a legitimate type?

   (c) What is the nature of the variable in $Favorite : \mathbb{P}Phone$? (i.e. atomic or composite?)

   (d) Is $Apple \in \mathbb{P}Phone$?

   (e) Is $\{Apple\} \in \mathbb{P}Phone$?

   (f) Is $\{\{\}\} \in \mathbb{P}Phone$?

   (g) Is $\{\} \in \mathbb{P}Phone$?

   (h) If we define variable $Favorite : \mathbb{P}Phone$, is $\{\}$ a legitimate value for variable *Favorite*?

   (i) Is $Favorite \in \mathbb{P}Phone$?

   (j) Is $Favorite \subset \mathbb{P}Phone$?

   <u>Solution</u>:

   (a) How do we interpret the expression $Favorite : \mathbb{P}Phone$? <u>Answer</u>: This is interpreted as "The variable *Favorite* can assume any value supported by the powerset of *Phone*.

   (b) Is $\mathbb{P}Phone$ a legitimate type? **Yes.**

(c) What is the nature of the variable in $Favorite : \mathbb{P}Phone$? <u>Answer</u>: Variable $Favorite$ is a **set**.

(d) Is $Apple \in \mathbb{P}Phone$? **No**.

(e) Is $\{Apple\} \in \mathbb{P}Phone$? **Yes**.

(f) Is $\{\{\}\} \in \mathbb{P}Phone$? **No**.

(g) Is $\{\} \in \mathbb{P}Phone$? **Yes**.

(h) If we define variable $favorite : \mathbb{P}Phone$, is $\{\}$ a legitimate value for variable $Favorite$? **Yes**.

(i) Is $Favorite \in \mathbb{P}Phone$? **Yes**.

(j) Is $Favorite \subset \mathbb{P}Phone$? **No**.

3. Consider the following relation:

$$laptops : Model \leftrightarrow Brand$$

where

$$
\begin{aligned}
&laptops = \\
&\quad \{ \\
&\qquad legion5 \mapsto lenovo, \\
&\qquad macbookair \mapsto apple, \\
&\qquad xps15 \mapsto dell, \\
&\qquad spectre \mapsto hp, \\
&\qquad xps13 \mapsto dell, \\
&\qquad swift3 \mapsto acer, \\
&\qquad macbookpro \mapsto apple, \\
&\qquad dragonfly \mapsto hp, \\
&\qquad envyx360 \mapsto hp \\
&\quad \}
\end{aligned}
$$

(a) What is the domain and the range of the relation?

<u>Answer</u>:

- The domain is defined as:

$$\begin{aligned} &\operatorname{dom} \mathit{laptops} = \\ &\quad \{ \\ &\qquad \mathit{legion}5, \\ &\qquad \mathit{macbookair}, \\ &\qquad \mathit{xps}15, \\ &\qquad \mathit{spectre}, \\ &\qquad \mathit{xps}13, \\ &\qquad \mathit{swift}3, \\ &\qquad \mathit{macbookpro}, \\ &\qquad \mathit{dragonfly}, \\ &\qquad \mathit{envyx}360 \\ &\quad \} \end{aligned}$$

- The range is defined as: ran $\mathit{laptops} = \{\mathit{lenovo}, \mathit{apple}, \mathit{dell}, \mathit{hp}, \mathit{acer}\}$.

(b) What is the result of the expression

$$\{\mathit{xps}15, \mathit{xps}13, \mathit{swift}3, \mathit{envyx}360\} \lhd \mathit{laptops}$$

What is the meaning of operator $\lhd$ and where would you deploy such operator in the context of a database management system?

Answer:

The result is

$$\begin{aligned}
&\{xps15, xps13, \mathit{swift}3, envyx360\} \lhd laptops = \\
&\quad \{ \\
&\qquad xps15 \mapsto dell, \\
&\qquad xps13 \mapsto dell, \\
&\qquad \mathit{swift}3 \mapsto acer, \\
&\qquad envyx360 \mapsto hp \\
&\quad \}
\end{aligned}$$

Domain restriction selects pairs based on their first element. We deploy such operators to model database queries.

(c) What is the result of the expression

$$laptops \rhd \{lenovo, hp\}$$

What is the meaning of operator $\rhd$ and where would you deploy such operator in the context of a database management system?

Answer:

The result is

$$\begin{aligned}
&laptops \rhd \{lenovo, hp\} = \\
&\quad \{ \\
&\qquad legion5 \mapsto lenovo, \\
&\qquad spectre \mapsto hp, \\
&\qquad \mathit{dragonfly} \mapsto hp, \\
&\qquad envyx360 \mapsto hp \\
&\quad \}
\end{aligned}$$

Range restriction selects pairs based on their second element. We deploy such operators to model database queries.

(d) What is the result of the expression

$$\{legion5, xps15, xps13, dragonfly\} \ntriangleleft laptops$$

What is the meaning of operator $\ntriangleleft$ and where would you deploy such operator in the context of a database management system?

<u>Answer</u>:

The result is

$$\begin{aligned}
&\{legion5, xps15, xps13, dragonfly\} \ntriangleleft laptops = \\
&\quad \{ \\
&\qquad macbookair \mapsto apple, \\
&\qquad spectre \mapsto hp, \\
&\qquad swift3 \mapsto acer, \\
&\qquad macbookpro \mapsto apple, \\
&\qquad envyx360 \mapsto hp \\
&\quad \}
\end{aligned}$$

Domain subtraction removes elements from the domain of the relation. We deploy such operation to model deletion of records.

(e) What is the result of the expression

$$laptops \ntriangleright \{apple, dell, hp\}$$

What is the meaning of operator $\ntriangleright$ and where would you deploy such operator in the context of a database management system?

Answer:

The result is

$$
\begin{aligned}
&\mathit{laptops} \rhd \{\mathit{apple}, \mathit{dell}, \mathit{hp}\} = \\
&\quad \{ \\
&\qquad \mathit{legion}5 \mapsto \mathit{lenovo}, \\
&\qquad \mathit{swift}3 \mapsto \mathit{acer} \\
&\quad \}
\end{aligned}
$$

Range subtraction removes elements from the codomain of the relation. We deploy such operation to model database updates (deletion of records).

(f) Consider the following expression

$$\mathit{laptops} \oplus \{\mathit{ideapad} \mapsto \mathit{lenovo}\}$$

i. What is the result of the expression?

ii. What is the meaning of operator $\oplus$ and where would you deploy such operator in the context of a database management system?

iii. Does the result of the expression have a permanent effect on the database (relation)? If not, describe in detail how would you ensure a permanent effect.

<u>Answer</u>:

i. The result is

$$\begin{array}{l} laptops \oplus \{ideapad \mapsto lenovo\} = \\ \quad \{ \\ \qquad ideapad \mapsto lenovo, \\ \qquad legion5 \mapsto lenovo, \\ \qquad macbookair \mapsto apple, \\ \qquad xps15 \mapsto dell, \\ \qquad spectre \mapsto hp, \\ \qquad xps13 \mapsto dell, \\ \qquad swift3 \mapsto acer, \\ \qquad macbookpro \mapsto apple, \\ \qquad dragonfly \mapsto hp, \\ \qquad envyx360 \mapsto hp \\ \quad \} \end{array}$$

ii. Relational overriding can model database updates (addition of records).

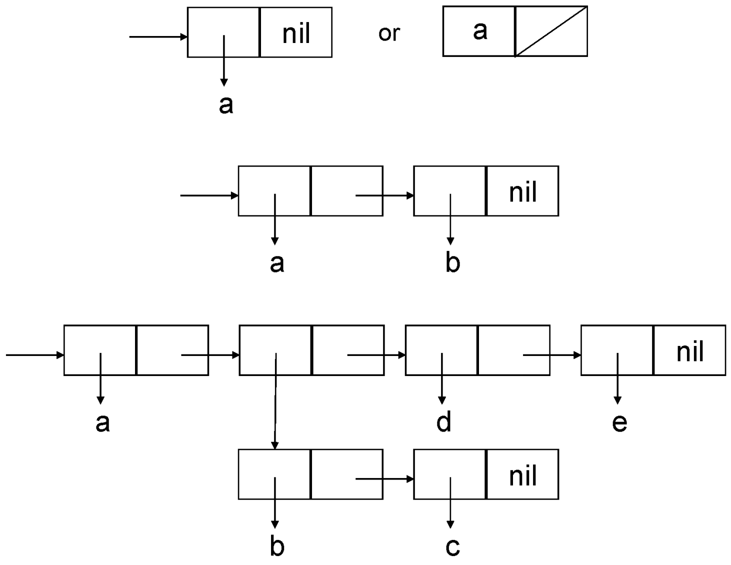
iii. The expression does not have a permanent effect on the database (relation). To ensure a permanent effect on the relation, we need to define an assignment statement

$$laptops' = laptops \oplus \{ideapad \mapsto lenovo\}$$

which reads "The value of variable (relation) *laptops* is assigned the result of the expression on the right-hand-side of the assignment statement."

4. You can produce a figure using any drawing package and save it as an image e.g. `.png`.

   The image file can then be embedded in your `.tex` document as follows:


nil
or
a
a
nil
a
b
nil
a
d
e
nil
b
c