Problem Sheet 4

Henri Sota
h.sota@jacobs-university.de
Computer Science 2022

October 11, 2019

Problem 4.1

a) Let $\preceq \subseteq \Sigma^* \times \Sigma^*$ be a relation such that $p \preceq w$ for $p, w \in \Sigma^*$ if p is a prefix of w. Show that $\preceq$ is a partial order.

In order to show that $\preceq$ is a partial order on Σ^*, $\preceq$ should have the following properties: reflexive, antisymmetric, transitive

- Reflexive: $\forall p, p \in \Sigma^*. \quad (p, p) \in \Sigma^*$ because p can be a prefix to itself $(p = w)$ i.e. p = "foo"
- Antisymmetric: $\forall p, w \in \Sigma^*. \quad ((p, w) \in \Sigma^* \wedge (w, p) \in \Sigma^*) \implies p = w$ because if p is a prefix of w and w is a prefix of p when $p = w$ from the definition of prefix then antisymmetric property is true
- Transitive:
  $\forall p, w \in \Sigma^*. \quad (p, w) \in \Sigma^*$ p is a prefix of w
  $\forall w, v \in \Sigma^*. \quad (w, v) \in \Sigma^*$ w is a prefix of v
  $\therefore$ p is a prefix of v $(p, v) \in \Sigma^*$ (including case when $p = w$ and $w = v$)

b) Let $\prec \subseteq \Sigma^* \times \Sigma^*$ be a relation such that $p \prec w$ for $p, w \in \Sigma^*$ if p is a proper prefix of w. Show that $\prec$ is a strict partial order.

In order to show that $\prec$ is a strict partial order on Σ^*, $\prec$ should have the following properties: irreflexive, asymmetric, transitive

- Irreflexive: $\forall p, p \in \Sigma^*. \quad (p, p) \notin \Sigma^*$ because p can't be a proper prefix to itself $(p \neq w)$
- Asymmetric: $\forall p, w \in \Sigma^*. \quad (p, w) \in \Sigma^* \implies (w, p) \notin \Sigma^*$ because if p is a prefix of w then w can't be a prefix of p when $p \neq w$ from the definition of proper prefix then asymmetric property is true
- Transitive:
  $\forall p, w \in \Sigma^*$. $\quad (p, w) \in \Sigma^*$ p is a prefix of w
  $\forall w, v \in \Sigma^* \quad (w, v) \in \Sigma^*$ w is a prefix of v
  $\therefore$ p is a prefix of v $(p, v) \in \Sigma^*$ (excluding case when $p = w$ and $w = v$)

c) Both order relations, $\preceq$ and $\prec$, are not total. Total property for $\preceq$ and $\prec$ would be defined as:

$$\forall p, w \in \Sigma^*.(p, w) \in \Sigma^* \vee (w, p) \in \Sigma^*$$

I.e. if $p =$ "abc" and $w =$ "def", then $(p, w) \notin \Sigma^*$ and $(w, p) \notin \Sigma^*$ because p is not a prefix of w and neither w is a prefix of p.

Problem 4.2

a) If $g \circ f$ is bijective, then f is injective and g is surjective.

For f to be injective, every element of the codomain B of f should be mapped to by at most one element of the domain A: $\forall x, y \in A. f(x) = f(y) \implies x = y$
For g to be surjective, every element of the codomain C of g should be mapped to by at least one element of the domain B: $\forall y \in C. \exists x \in B. f(x) = y$

Let $A = \{x, y, z\}$, $B = \{m, n, o, p\}$, $C = \{a, b, c\}$

- f in this case is prescribed by $x \mapsto m$, $y \mapsto n$ and $z \mapsto p$. (leaving o as an element of the codomain for which there is no value which maps to it) $\rightarrow f$ is injective because every element of the codomain has been mapped by at most one element of the domain
- g in this case is prescribed by $m \mapsto a$, $n \mapsto b$, $o \mapsto c$ and $p \mapsto c$. $\rightarrow g$ is surjective because every element of the codomain has been mapped by at least one element of the domain
- $g \circ f$ has domain $A = \{x, y, z\}$ and codomain $C = \{a, b, c\}$. $g \circ f$ is bijective because every element of the codomain C is mapped to by exactly only one element of the domain A. $g \circ f$ is prescribed by $x \mapsto a$, $y \mapsto b$ and $z \mapsto c$.

b) Let $A = \{x, y, z\}$, $B = \{m, n, o\}$, $C = \{a, b\}$

- f in this case is prescribed by $x \mapsto m$, $y \mapsto n$ and $z \mapsto o$. $\rightarrow f$ is injective because every element of the codomain has been mapped by at most one element of the domain
- g in this case is prescribed by $m \mapsto a$, $n \mapsto a$ and $o \mapsto b$. $\rightarrow g$ is surjective because every element of the codomain has been mapped by at least one element of the domain
- $g \circ f$ has domain $A = \{x, y, z\}$ and codomain $C = \{a, b\}$. $g \circ f$ is not bijective because one element of the codomain C has been mapped by more than element of domain A. $g \circ f$ is prescribed by $x \mapsto a$, $y \mapsto a$ and $z \mapsto b$.

c) For f to be not surjective, there must at least one element of the codomain to which no element of the domain map to: $\exists y \in B \ s.t. \ \forall x \in A. f(x) \neq y$
For g to be not injective, there must be at least one 2 distinct elements of the domain which map to the same value in the codomain: $\exists x, y \in B, x \neq y : f(x) = f(y)$

Let $A = \{x, y\}$, $B = \{m, n, o\}$, $C = \{a, b\}$

- f in this case is prescribed by $x \mapsto m$ and $y \mapsto n$. $\rightarrow f$ is not surjective because at least one element of the codomain is not mapped by any element of the domain (o).
- g in this case is prescribed by $m \mapsto a$, $n \mapsto b$ and $o \mapsto a$. $\rightarrow g$ is not injective because there are 2 distinct elements of the domain B which map to the same element of the codomain C.
- $g \circ f$ has domain $A = \{x, y\}$ and codomain $C = \{a, b\}$. $g \circ f$ is bijective because every element of the codomain C is mapped to by exactly only one element of the domain A. $g \circ f$ is prescribed by $x \mapsto a$ and $y \mapsto b$.

Problem 4.3

a)
```haskell
-- Guards used to check if input that is Integer is a special prime or not
-- Integer is a special prime if it is prime and it is the sum of 2
   neighboring
-- primes and 1
-- Function outputs the result as a Boolean value
isSpecialPrime :: Integer -> Bool
isSpecialPrime num
  -- Comprehend a list made of all the primes smaller than num
  -- Call function checkSum to see if any neighboring primes and 1 equal to
     num
  | isPrime num == True = checkSum num [x | x <- [1..num-2], isPrime x]
  | otherwise = False

-- Pattern matching combined with guards to recursively iterate through our
   list
-- by taking 2 elements of our list and checking if their sum + 1 equals to
   num
-- In case that isn't True check the tail of our list with checkSum
-- If we have arrived at the end of the list (calling checkSum with empty list
   )
-- our num is not a special prime
checkSum :: Integer -> [Integer] -> Bool
checkSum num [] = False
checkSum num lst
  | ((head (take 2 lst)) + (last (take 2 lst)) + 1) == num = True
  | otherwise = checkSum num (tail lst)
```
Listing 1: isPrime is a function taken from Problem 3.3

In order to test this function, I've used multiple calls to this function with different input, given by a list comprehension, namely the list of numbers from 2 to 100, which yielded a result:

[13,19,31,37,43,53,61,79]