Introduction to Computer Science
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CH08-320101
Date: 2017-10-03
Due: 2017-10-10

ICS Problem Sheet #4

Problem 4.1: *prefix order relations* (2+2+1 = 5 points)

Let Σ be a finite set (called an alphabet) and let Σ^* be the set of all words that can be created out of Σ (Kleene closure of Σ that includes the empty word). A word $p \in \Sigma^*$ is called a prefix of a word $w \in \Sigma^*$ if there is a word $q \in \Sigma^*$ such that $w = pq$. A prefix p is called a proper prefix if $p \neq w$.

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.

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

c) Are the two order relations $\preceq$ and $\prec$ total?

Make sure you write complete proofs for the properties of the order relations. Do not assume something is 'obvious' or 'trivial' — always reason with the definition of the order relation.

Problem 4.2: *function composition* (2+1+1 = 4 points)

Let A, B and C be sets and let $f : A \mapsto B$ and $g : B \mapsto C$ be two functions.

a) Prove the following statement: If $g \circ f$ is bijective, then f is injective and g is surjective.

b) Find an example demonstrating that $g \circ f$ is not bijective even though f is injective and g is surjective.

c) Find an example demonstrating that $g \circ f$ is bijective even though f is not surjective and g is not injective.

Problem 4.3: *anagram (haskell)* (1 point)

An anagram is word formed by rearranging the letters of a different word. Implement a function `anagrams` that takes a list of strings and groups set of strings that are anagrams into a separate list. The result should be a list of lists of strings. (Considering that strings are also lists, one could say the result should be of type list of lists of lists). (In more mathematical words, the function splits the list of strings into lists representing equivalence classes according to the anagram equivalence relation.)

Hint: You may want to implement a function that tests whether two strings are "anagram equivalent" and a function that splits a list according to an the "anagram equivalence relation".

Example function evaluation (white space added for readability):

```
anagrams ["shall", "siren", "skill", "slain", "halls",
          "kills", "resin", "risen", "nails", "snail", "smash", "swing"]
[["shall","halls"],["siren","resin","risen"],["skill","kills"],
 ["slain","nails","snail"],["smash"],["swing"]]
```