Collaborators: None. Written Sources: Textbook

Problem 1

A *formula of set theory* is a predicate formula that only uses the predicate $x \in y$. The domain of discourse is the collection of sets, and $x \in y$ is interpreted to mean the set x is one of the elements in the set y.
For example, since x and y are the same set iff they have the same members, here's how we can express equality of x and y with a formula of set theory:

$$(x == y) ::= \forall z.(z \in x \text{ IFF } z \in y)$$

a. Explain how to write a formula Members(p, a, b) of set theory that means $p = \{a, b\}$.

$$\forall x.\, x \in p \text{ IFF } (x = a \text{ OR } x = b)$$

A *pair* (a, b) is simply a sequence of length two whose first item is a and whose second is b. Sequences are a basic mathematical data type we take for granted, but when we're trying to show how all of mathematics can be reduced to set theory, we need a way to represent the ordered pair (a, b) as a set. One way that will work is to represent (a, b) as

$$\text{pair}(a, b) ::= \{a, \{a, b\}\}$$

b. Explain how to write a formula Pair(p, a, b) of set theory that means $p = \text{pair}(a, b)$.

$$\text{Pair}(p, a, b) = \text{Members}(p, a, \{a, b\})$$

c. Explain how to write a formula Second(p, b) of set theory that means p is a pair whose second item is b.

$$\text{Second}(p, b) = \forall x, y.\, x \in p \text{ IFF } (x = y \text{ OR } x = \{y, b\})$$

Problem 2

Prove De Morgan's Law for set equality

$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

by showing with a chain of IFF's that $x \in$ the left hand side IFF $x \in$ the right hand side. You may assume the proportional version of De Morgan's Law:

$$\text{NOT}(P \text{ AND } Q) = \overline{P} \text{ OR } \overline{Q}$$

$$\begin{aligned} x \in \overline{A \cap B} &\text{ IFF } x \in \overline{A \text{ AND } B} \\ &\text{ IFF } x \in \text{NOT}(A \text{ AND } B) \\ &\text{ IFF } x \in \overline{A} \text{ OR } x \in \overline{B} \\ &\text{ IFF } x \in \overline{A} \cup \overline{B} \end{aligned}$$

Problem 3

A *binary word* is a finite sequence of `0's` and `1's`. For example, `(1, 1, 0)` and (1) are words of length three and one, respectively. We usually omit the parentheses and commas in the descriptions of words, so the preceding binary words would just be written as `110` and `1`.
The basic operation of placing one word immediately after another is called *concatentation.* For example, the concatentation of `110` and `1` is `1101`, and the concatentation of `110` with itself is `110110`.
We can extend this basic operation on words to an operation on *sets* of words. To emphasize the distinction between a word and a set of words, from now on, we'll refer to a set of words as a *language.* Now if R and S are languages, then $R \cdot S$ is the language consisting of all the words you can get by concatenating a word from R with a word from S. That is

$$R \cdot S ::= \{rs \mid r \in R \text{ AND } s \in S\}$$

For example

$$\{0, 00\} \cdot \{00, 000\} = \{000, 0000, 00000\}$$

Another example is $D \cdot D$, abbreviated as D^2, where $D ::= \{1, 0\}$ is just the two binary digits.

$$D^2 = \{00, 01, 10, 11\}$$

In other words, D^2 is the language consisting of all the length two words. More generally, D^n will be the language of length n words.
If S is a language, the language you get by concatenating any number of copies of words in S is called S^*-pronounced "S star." (By convention, the empty word λ is always included in S^*). For example, $\{0, 11\}^*$ is the language consisting of all the words you can make by stringing together `0's` and `11's`. This language could also be described as consisting of all the words whose blocks of `1's` are always of even length. Another example is $(D^2)^*$, which consists of all the even length words. Finally, the language B of all binary words is just D^*.
A language is called *concatenation-definable (c-d)* if it can be constructed by starting with finite languages and then applying the operations of concatenation, union and complement (relative to B) to these languages a finite number of times. Note that the *-operation is *not*

allowed. For this reason, the c-d languages are also called the "star-free languages" in the course textbook.
Lots of interesting languages turn out to be concatenation-definable, but some very simple languages are not. This problem ends with the conclusion that the language $\{00\}^*$ of even length words whose bits are all `0's` is not a c-d language.

a. Show that if R and S are c-d, then so is $R \cap S$.
If R and S are c-d, then we can apply c-d operations to them. First taking the complement, we know that $\overline{R}$ and $\overline{S}$ are both c-d. Their union $\overline{R} \cup \overline{S}$ is also c-d.
Taking the complement of the union, we have

$$\overline{\overline{R} \cup \overline{S}} = R \cap S$$

Therefore, $R \cap S$ is also c-d.

Now we can show that the set B of all binary words is c-d as follows. Let u and v be any two different binary words. Then $\{u\} \cap \{v\}$ eqauls the empty set. But $\{u\}$ and $\{v\}$ are c-d by definition, so by part a, the empty set is c-d and therefore so is $\overline{0} = B$.
Now a more interesting example of a c-d set is the language of all binary words that include three consecutive `1's`.

$$B\mathtt{111}B$$

Notice that the proper expression here is $B \cdot \{\mathtt{111}\} \cdot B$. But it causes no confusion and helps readability to omit the dots in concatenations and the curly braces for sets with one element.

b. Show that the language consisting of the binary words that start with `0` and end with `1` is c-d.
The three sets `0`, `1` and B are all c-d. Therefore, their concatenation $\mathtt{0} \cdot B \cdot \mathtt{1}$ is also c-d.

c. Show that $\mathtt{0}^*$ is c-d.

$$B \cdot \mathtt{1} \cdot B \text{ is c-d.}$$

Let's take the complement which is also c-d.

$$\overline{B \cdot \mathtt{1} \cdot B} = \mathtt{0}^*$$

d. Show that $\{\mathtt{01}\}^*$ is c-d.
Our strategy consists of taking the complement of the union of all words that contain invalid two-bit combinations `00`, `10`, and `11` and one-bit combinations `0` and `1`.

$$\overline{(B \cdot \mathtt{00} \cdot B) \cup (B \cdot \mathtt{10} \cdot B) \cup (B \cdot \mathtt{11} \cdot B) \cup \mathtt{0} \cup \mathtt{1}} = \{\mathtt{01}\}^*$$

Let's say a language S is 0-*finite* when it includes only a finite number of words whose bits are all $\texttt{0's}$, that is when $S \cap 0^*$ is a finite set of words. A language S is 0-*boring*, boring for short, when either S or $\overline{S}$ is $\texttt{0}$-finite.

e. Explain why $\{\texttt{00}\}^*$ is not boring.

$$\{\texttt{00}\}^* \cap \texttt{0}^* = \{\texttt{00}\}^*$$

The set S contains all the even-length $\texttt{0's}$. The set $\overline{S}$ contains the odd-length $\texttt{0's}$. Therefore $\overline{S} \cap 0^*$ is not a finite set. Therefore, $\{\texttt{00}\}$ is not boring.

f. Verify that if R and S are boring, then so is $R \cup S$.

$$(R \cup S) \cap \texttt{0}^* = R \cap \texttt{0}^* \cup S \cap \texttt{0}^* \tag{1}$$

$$\overline{(R \cup S)} \cap \texttt{0}^* = \overline{R} \cap \texttt{0}^* \cap \overline{S} \cap \texttt{0}^* \tag{2}$$

There are four cases. 1) R and S are $\texttt{0}$-finite 2) R and $\overline{S}$ are $\texttt{0}$-finite 3) $\overline{R}$ and S are $\texttt{0}$-infinite and 4) $\overline{R}$ and $\overline{S}$ are $\texttt{0}$-infinite.
In 1), Eqn 1 would be $\texttt{0}$-finite because it's the union of two $\texttt{0}$-finite sets. In 2) Eqn 1 would be $\texttt{0}$-finite because it's the union of R, a $\texttt{0}$-finite set with a non $\texttt{0}$-finite set. In 3), Eqn 1 would be $\texttt{0}$-finite because it's the union of a S, a $\texttt{0}$-finite set with a non $\texttt{0}$-finite set. In 4) Eqn 2 would be $\texttt{0}$-finite because it's the intersection of two $\texttt{0}$-finite sets.

g. Verify that if R and S are boring, then so is $R \cdot S$.

$$(R \cdot S) ::= \{rs \mid r \in R \text{ AND } s \in S\} \tag{3}$$

$$(R \cdot S) \cap \texttt{0}^* \tag{4}$$

$$\overline{(R \cdot S)} \cap \texttt{0}^* \tag{5}$$

Similar to the previous part, Eqn 4 is $\texttt{0}$-finite for the first three cases and Eqn 5 is $\texttt{0}$-finite for the last case.

h. Explain why all c-d languages are boring.
c-d languages are constructed from finite languages which are boring by default because their complements are $\texttt{0}$-infinite sets. The union or concatenation of boring languages is still boring. Taking the complement also ensures that the new language is still boring (since two complements give the original language which was boring). Therefore all c-d languages are boring and the set $(\texttt{00})$ is not a c-d language.