Exercise 1

1 Propositional Logic

Exercise 1 (Sheffer Stroke)**.** The operator $\uparrow$, also known as Sheffer stroke or NAND gate, has the following truth table:

$\uparrow$	0	1
0	1	1
1	1	0

Show that any propositional formula with language $\{\wedge, \vee, \rightarrow, \neg, 0, 1\}$ can be equivalently expressed using only the language $\{\uparrow\}$

Exercise 2 (Decision Procedure for SAT)**.** Recall the "Decision and Simplification" proof system (Infer_D):

$$\frac{F \quad G}{F\,[x := 0] \vee G\,[x := 1]}\,\text{CA} \qquad \frac{F}{F'}\,\textsc{Simp with } F \rightsquigarrow F'$$

where $\rightsquigarrow$ is the result of applying a finite sequence of any of the following simplifications anywhere in the formula:

$$0 \wedge F \rightsquigarrow 0 \qquad F \wedge 0 \rightsquigarrow 0 \qquad 1 \wedge F \rightsquigarrow F \qquad F \wedge 1 \rightsquigarrow F$$
$$0 \vee F \rightsquigarrow F \qquad F \vee 0 \rightsquigarrow F \qquad 1 \vee F \rightsquigarrow 1 \qquad F \vee 1 \rightsquigarrow 1$$
$$\neg 0 \rightsquigarrow 1 \qquad \neg 1 \rightsquigarrow 0$$

For each of the following formulas, express it using only the connectors $\neg$, $\wedge$ and $\vee$. Then prove they are tautologies by assuming their inverse and deriving 0 using Infer_D.

1. $((a \rightarrow b) \rightarrow a) \rightarrow a$
2. $\neg(a \wedge b) \rightarrow (\neg a \vee \neg b)$
3. $((\neg a \rightarrow b) \wedge (a \rightarrow b)) \rightarrow b$

Exercise 3 (if-then-else)**.**

1. Consider the two following small programs:

```
1   def f1(a:Boolean, b:Boolean, c:Boolean): Boolean = {
2     if a || b then
3       if b && !a then
4         b && c
5       else
6         !b && c
7     else
8       c
9   }
```

```
1   def f2(a:Boolean, b:Boolean, c:Boolean): Boolean = {
2     if c then
3       if a then
4         !b
5       else true
6     else false
7   } ensuring (_ == f1(a, b, c))
```

   Are they equivalent on all inputs? Show that they are or aren't by expressing those programs as propositional formulas. Note that the ensuring clause is automatically proved or disproved by Stainless!

2. Can any formula be expressed with only **if** a **then** b **else** c, where a, b and c are recursively constructed the same way? What if a is restricted to be a variable?

Exercise 4 (Propositional Tautologies)**.** For each of the following propositional logic formulas determine whether it is valid or not. If it is valid, prove it (using any acceptable mathematical argument of your choice), otherwise give a counterexample. (From *The Calculus of Computation* by A.R. Bradley and Z. Manna.)

1. $(P \wedge Q) \rightarrow P \rightarrow Q$
2. $(P \rightarrow Q) \vee (P \wedge \neg Q)$
3. $(P \rightarrow Q \rightarrow R) \rightarrow P \rightarrow R$
4. $(P \rightarrow Q \vee R) \rightarrow P \rightarrow R$
5. $\neg(P \vee Q) \rightarrow R \rightarrow \neg R \rightarrow Q$
6. $(P \rightarrow Q) \rightarrow P \rightarrow Q$

7. $((P \to Q) \to P) \to P$
8. $((P \to Q) \to P) \to Q$
9. $(\neg Q \to \neg P) \to P \to Q$
10. $(\neg R \to \neg Q \to \neg P) \to P \to Q \to R$
11. $(P \vee Q) \to (P \vee \neg Q) \to (\neg P \vee \neg Q) \to P$
12. $(P \vee Q) \to (P \vee \neg Q) \to (\neg P \vee \neg Q) \to Q$
13. $\neg(P \wedge Q) \to P \to \neg Q$

2 Transition Systems and Invariants

Notation: $M = (S, I, r, A)$ is a transition system where S are the states, $I \subseteq S$ the set of initial states, $r \subseteq S \times A \times S$ and A is the input alphabet.

Exercise 5 (Special Invariants)**.** Prove the following:

1. S is an inductive invariant. It is the largest among all invariants.
2. $Reach(M)$ is an inductive invariant.
3. $Reach(M)$ is the smallest of all possible invariants.

Exercise 6 (Closure of Invariants)**.** A set of closed under an operation if applying the operation to elements of the set gives the result in the set. For example, the set of even natural numbers is closed under addition, whereas the set of odd natural numbers is not closed under addition.

Is the set of all invariants of M closed under the following operations:

1. union
2. intersection
3. complement with respect to S
4. operation $f : 2^S \to 2^S$ defined by $f(X) = Reach(M) \cup (S \setminus X)$

Answer the same questions about all *inductive invariants* of M.

3 Relations

We consider relations $r, s, t, r_1, r_2, r_1', \ldots \subseteq A \times A$ (where A is just arbitrary set, nothing to do with the input signals of the transition systems). Here $X \subseteq A$. Define:

$$\begin{array}{rcl} \Delta_X & = & \{(x,x) \mid x \in X\} \\ r^{-1} & = & \{(x,y) \mid (y,x) \in r\} \\ X \bullet r & = & r[X] \\ ran(r) & = & \{y \mid \exists x.(x,y) \in r\} \end{array}$$

Exercise 7 (Relation Identities)**.** Prove the following or give a counterexample.

1. $(X \bullet r_1) \bullet r_2 = X \bullet (r_1 \circ r_2)$
2. $(r \cup s) \circ t = (r \circ t) \cup (s \circ t)$
3. $(r \cap s) \circ t = (r \circ t) \cap (s \circ t)$
4. $(r_1 \circ r_2)^{-1} = (r_2^{-1} \circ r_1^{-1})$
5. $X \bullet r = ran(\Delta_X \circ r)$
6. If $r_1 \subseteq r_1'$ then $r_1 \circ r_2 \subseteq r_1' \circ r_2$ and $r_2 \circ r_1 \subseteq r_2 \circ r_1'$.
7. If $r_1 \subseteq r_1'$ then $r_1 \cup r_2 \subseteq r_1' \cup r_2$ and $r_2 \cup r_1 \subseteq r_2 \cup r_1'$.

Exercise 8 (Transitive relations)**.** Given a relation $r \subseteq A \times A$, prove that r is transitive if and only if $r \circ r \subseteq r$.

Exercise 9 (Symmetric relations)**.** Recall that a relation $r \subseteq A \times A$ is symmetric if $\forall x, y \in A.\ (x,y) \in r \Rightarrow (y,x) \in r$.
Now let r be an arbitrary relation. Prove that $r^{-1} \circ (r \cup r^{-1})^* \circ r$ is symmetric.

Exercise 10 (Transitive closure)**.** Recall that we define the powers of a relation $r \subseteq A \times A$ as follows:

$$r^0 = \triangle_A, \quad r^1 = r, \quad \text{and} \quad r^{n+1} = r^n \circ r$$

We showed that the *reflexive and transitive closure* $r^* = \bigcup_{n \geq 0} r^n$ is the smallest reflexive and transitive relation on A containing r. Show that for any relation r on a set A, $(r \cup r^{-1})^*$ is the least equivalence relation containing r. Precisely, show that

(i) $(r \cup r^{-1})^*$ is an equivalence relation, and

(ii) if s is an equivalence relation containing r, then $(r \cup r^{-1})^* \subseteq s$.

4 Finite State Machines with Boolean Variables

Exercise 11 (Finite State Machines with Boolean Variables)**.** We consider in this exercise finite-state machines enriched with boolean variables (FSM for short). Formally, an FSM is a pair (V, Q, δ) where:

- V is a finite set of (boolean) variable names,
- Q is a finite set of states,
- For every pair of states p and q, $\delta(p, q)$ is a propositional formula containing variables from V and V', where V' is a copy of V with primed variable names.

A configuration of the FSM is determined by a pair (q, m) where

- q is a state in Q, called the *control state*,
- $m : V \to \{\top, \bot\}$ is a mapping from variable names to true or false.

The FSM can move from a configuration (p, m) to (q, m') if the formula $\delta(p, q)$ is true when we interpret every variable from V using the mapping m, and every variable in V' using m'. We define the relation $r \subseteq (Q \times V \to \{\top, \bot\}) \times (Q \times V \to \{\top, \bot\})$ to contain all such pairs of configurations (p, m) and (q, m').

Consider now the FSM in Figure 1, inspired from Peterson's algorithm (we recommend reading this page before the exercise). We made use of two conventions when drawing the figure:

- When there is no transition drawn from a state p to a state q, we mean that $\delta(p, q) = \bot$ in the FSM.
- When a primed variable X' does not appear in a transition, we leave the variable unchanged, meaning that there is an implicit conjunct $X' \leftrightarrow X$ in the transition.

For example, the transition from $(0, 0)$ to $(0, 1)$ can be taken regardless of the initial mapping of boolean variables. It changes variable A to $\top$, and leaves the variables B and C unchanged. The transition from $(2, 0)$ to $(3, 0)$ can only be taken when the mapping of boolean variables respects $\neg A \vee C$, and leaves all variables unchanged.

Does there exist mappings m and m' such that the FSM can move (using multiple steps) from configuration $((0, 0), m)$ to $((3, 3), m')$, i.e. such that $(((0, 0), m), ((3, 3), m')) \in r^*$?

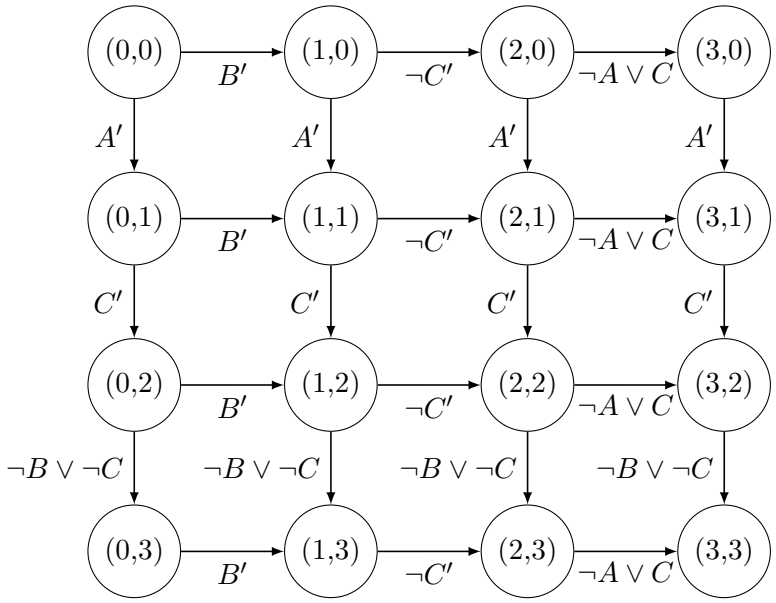

Figure 1: A finite-state machine with boolean variables $\{A, B, C\}$.