Section 03: Solutions

1. Canonical Forms

Consider the boolean functions $F(A, B, C)$ and $G(A, B, C)$ specified by the following truth table:

A	B	C	$F(A,B,C)$	$G(A,B,C)$
1	1	1	1	0
1	1	0	1	1
1	0	1	0	0
1	0	0	0	0
0	1	1	1	1
0	1	0	1	0
0	0	1	0	1
0	0	0	1	0

(a) Write the DNF and CNF expressions for $F(A, B, C)$. **Solution:**

DNF:
From rows 1, 2, 5, 6, and 8, we get the following minterms respectively, and add them together:
$ABC + ABC' + A'BC + A'BC' + A'B'C'$

CNF:
From rows 3, 4, and 7, we get the following maxterms respectively, and multiply them together:
$(A' + B + C')(A' + B + C)(A + B + C')$

(b) Write the DNF and CNF expressions for $G(A, B, C)$. **Solution:**

DNF:
From rows 2, 5, and 7, we get the following minterms respectively, and add them together:
$ABC' + A'BC + A'B'C$

CNF:
From rows 1, 3, 4, 6, and 8, we get the following maxterms respectively, and multiply them together:
$(A' + B' + C')(A' + B + C')(A' + B + C)(A + B' + C)(A + B + C)$

2. Translate to Logic

Express each of these system specifications using predicates, quantifiers, and logical connectives. For some of these problems, more than one translation will be reasonable depending on your choice of predicates.

(a) Every user has access to an electronic mailbox. **Solution:**

Let the domain be users and mailboxes. Let $\texttt{User}(x)$ be "x is a user", let $\texttt{Mailbox}(y)$ be "y is a mailbox", and let $\texttt{Access}(x, y)$ be "x has access to y".

$$\forall x\ (\texttt{User}(x) \rightarrow (\exists y\ (\texttt{Mailbox}(y) \wedge \texttt{Access}(x, y))))$$

(b) The system mailbox can be accessed by everyone in the group if the file system is locked. **Solution:**

Solution 1: Let the domain be people in the group. Let $\texttt{CanAccessSM}(x)$ be "x has access to the system mailbox". Let p be the proposition "the file system is locked."

$$p \rightarrow \forall x \ \texttt{CanAccessSM}(x).$$

Solution2: Let the domain be people and mailboxes and use $\texttt{Access}(x, y)$ as defined in the solution to part (a), and then also add $\texttt{InGroup}(x)$ for "x is in the group", and let `SystemMailBox` be the name for the system mailbox. Then the translation becomes

$$\texttt{FileSystemLocked} \rightarrow \forall x \ (\texttt{InGroup}(x) \rightarrow \texttt{Access}(x, \texttt{SystemMailBox})).$$

(c) The firewall is in a diagnostic state only if the proxy server is in a diagnostic state.

Solution:

Let the domain be all applications. Let $\texttt{Firewall}(x)$ be "x is the firewall", and let $\texttt{ProxyServer}(x)$ be "x is the proxy server." Let $\texttt{Diagnostic}(x)$ be "x is in a diagnostic state".

$$\forall x \ \forall y \ ((\texttt{Firewall}(x) \wedge \texttt{Diagnostic}(x)) \rightarrow (\texttt{ProxyServer}(y) \rightarrow \texttt{Diagnostic}(y))$$

(d) At least one router is functioning normally if the throughput is between 100kbps and 500 kbps and the proxy server is not in diagnostic mode. **Solution:**

Let the domain be all applications and routers. Let $\texttt{Router}(x)$ be "x is a router", and let $\texttt{ProxyServer}(x)$ be "x is the proxy server." Let $\texttt{Diagnostic}(x)$ be "x is in a diagnostic state". Let p be "the throughput is between 100kbps and 500 kbps". Let $\texttt{Functioning}(y)$ be "y is functioning normally".

$$(p \wedge \forall x \ (\neg \texttt{ProxyServer}(x) \vee \neg \texttt{Diagnostic}(x))) \rightarrow \exists y \ (\texttt{Router}(y) \wedge \texttt{Functioning}(y))$$

3. Translate to English

Translate these system specifications into English where $F(p)$ is "Printer p is out of service", $B(p)$ is "Printer p is busy", $L(j)$ is "Print job j is lost," and $Q(j)$ is "Print job j is queued". Let the domain be all printers and all print jobs.

(a) $\exists p \ (F(p) \wedge B(p)) \rightarrow \exists j \ L(j)$ **Solution:**

If at least one printer is busy and out of service, then at least one job is lost.

(b) $(\forall j\ B(j)) \rightarrow (\exists p\ Q(p))$ **Solution:**

If all printers are busy, then there is a queued job.

(c) $\exists j\ (Q(j) \wedge L(j)) \rightarrow \exists p\ F(p)$ **Solution:**

If there is a queued job that is lost, then a printer is out of service.

(d) $(\forall p\ B(p) \wedge \forall j\ Q(j)) \rightarrow \exists j\ L(j)$ **Solution:**

If all printers are busy and all jobs are queued, then there is some lost job.

4. Domain Restriction

Translate each of the following sentences into logical notation. These translations require some of our quantifier tricks. You may use the operators $+$ and $\cdot$ which take two numbers as input and evaluate to their sum or product, respectively. Remember:

- To restrict the domain under a $\forall$ quantifier, add a hypothesis to an implication.
- To restrict the domain under an $\exists$ quantifier, AND in the restriction.
- If you want variables to be different, you have to explicitly require them to be not equal.

(a) Domain: Positive integers; Predicates: `Even`, `Prime`, `Equal`
"There is only one positive integer that is prime and even." **Solution:**

$\exists x(\texttt{Prime}(x) \wedge \texttt{Even}(x) \wedge \forall y[\neg\, \texttt{Equal}(x, y) \rightarrow \neg(\texttt{Even}(y) \wedge \texttt{Prime}(y))])$

(b) Domain: Real numbers; Predicates: `Even`, `Prime`, `Equal`
"There are two different prime numbers that sum to an even number." **Solution:**

$\exists x \exists y(\texttt{Prime}(x) \wedge \texttt{Prime}(y) \wedge \neg \mathit{Equal}(x, y) \wedge \texttt{Even}(x + y))$

(c) Domain: Real numbers; Predicates: `Even`, `Prime`, `Equal`
"The product of two distinct prime numbers is not prime." **Solution:**

$\forall x \forall y([\texttt{Prime}(x) \wedge \texttt{Prime}(y) \wedge \neg\, \texttt{Equal}(x, y)] \rightarrow \neg\, \texttt{Prime}(xy))$

(d) Domain: Real numbers; Predicates: `Even`, `Prime`, `Equal`, `Postivite`, `Greater`, `Integer`
"For every positive integer, there is a greater even integer" **Solution:**

$\forall x(\texttt{Positive}(x) \wedge \texttt{Integer}(x) \rightarrow [\exists y(\texttt{Integer}(y) \wedge \texttt{Even}(y) \wedge \texttt{Greater}(y, x))])$
Or equivalently: $\forall x \exists y(\texttt{Positive}(x) \wedge \texttt{Integer}(x) \rightarrow (\texttt{Integer}(y) \wedge \texttt{Even}(y) \wedge \texttt{Greater}(y, x)))$

5. Quantifier Switch

Consider the following pairs of sentences. For each pair, determine if one implies the other, if they are equivalent, or neither.

(a) $\forall x\ \forall y\ P(x, y)$ $\quad$ $\forall y\ \forall x\ P(x, y)$ **Solution:**

These sentences are the same; switching universal quantifiers makes no difference.

(b) $\exists x\ \exists y\ P(x, y)$ $\quad$ $\exists y\ \exists x\ P(x, y)$ **Solution:**

These sentences are the same; switching existential quantifiers makes no difference.

(c) $\forall x\ \exists y\ P(x, y)$ $\quad$ $\forall y\ \exists x\ P(x, y)$ **Solution:**

These are only the same if P is symmetric (i.e., the order of the arguments doesn't matter). If the order of the arguments does matter, then these are different statements. For instance, if $P(x, y)$ is "$x < y$", then the first statement says "for every x, there is a corresponding y such that $x < y$", whereas the second says "for every y, there is a corresponding x such that $x < y$". In other words, in the first statement y is a function of x, and in the second x is a function of y.

If your domain of discourse is "positive integers", for example, the first is true and the second is false; but for "negative integers" the second is true while the first is false.

(d) $\forall x\ \exists y\ P(x, y)$ $\quad$ $\exists x\ \forall y\ P(x, y)$ **Solution:**

These two statements are usually different.

(e) $\forall x\ \exists y\ P(x, y)$ $\quad$ $\exists y\ \forall x\ P(x, y)$ **Solution:**

The second statement is "stronger" than the first (that is, the second implies the first). For the first, y is allowed to depend on x. For the second, one specific y must work for all x. Thus if the second is true, whatever value of y makes it true, can also be plugged in for y in the first statement for every x. On the other hand, if the first statement is true, it might be that different y's work for the different x's and no single value of y exists to make the latter true.

As an example, let you domain of discourse be positive real numbers, and let $P(x, y)$ be $xy = 1$. The first statement is true (always take y to be $1/x$, which is another positive real number). The second statement is not true; it asks for a single number that always makes the product 1.

6. Quantifier Ordering

Let your domain of discourse be a set of `Element` objects given in a list called `Domain`. Imagine you have a predicate $\texttt{pred}(x, y)$, which is encoded in the java method `public boolean pred(int x, int y)`. That is you call your predicate `pred` true if and only if the java method returns `true`.

(a) Consider the following Java method:

```
public boolean Mystery(Domain D){
    for(Element x : D) {
        for(Element y : D) {
            if(pred(x,y))
                return true;
        }
    }
}
```

`Mystery` corresponds to a quantified formula (for `D` being the domain of discourse), what is that formula?
Solution:

$\exists x \exists y(\texttt{pred}(x, y))$. If any combination of x and y causes pred to evaluate to true, we return true; that is we just want x, y to exist.

(b) What formula does `mystery2` correspond to

```
public boolean Mystery2(Domain D){
    for(Element x : D) {
        boolean thisXPass = false;
        for(Element y : D) {
            if(pred(x,y))
                thisXPass = true;
        }
        if(!thisXPass)
            return false;
    }
    return true;
}
```

Solution:

$\forall x \exists y(\texttt{pred}(x, y))$.

For a given x, when we come across a y that makes $\texttt{pred}(x, y)$ true, we set the given x to pass (so one y suffices for a given x) but we require **every** x to pass, so x is universally quantified. Since y is allowed to depend on x, we have x as the outermost variable.

7. All for 1 and One $\forall$

Let the domain of discourse contain only the two object a and b. *For this problem only*, you are allowed to use the following fake equivalence rules

$$\forall x\, P(x) \equiv P(a) \wedge P(b) \qquad \forall \to \wedge$$
$$\exists x\, P(x) \equiv P(a) \vee P(b) \qquad \exists \to \vee$$

(a) Use a chain of equivalences to show that $Q \wedge (\exists x\, P(x)) \equiv \exists x\, Q \wedge P(x)$.

Solution:

$$
\begin{aligned}
Q \wedge (\exists x\, P(x)) &\equiv Q \wedge (P(a) \vee P(b)) && \exists \to \vee \\
&\equiv (Q \wedge P(a)) \vee (Q \wedge P(b)) && \text{Distributivity} \\
&\equiv \exists x\, Q \wedge P(x) && \exists \to \vee
\end{aligned}
$$

(b) Likewise show that $Q \vee (\exists x\, P(x)) \equiv \exists x\, Q \vee P(x)$.

Solution:

$$
\begin{aligned}
Q \vee (\exists x\, P(x)) &\equiv Q \vee (P(a) \vee P(b)) && \exists \to \vee \\
&\equiv (Q \vee Q) \vee (P(a) \vee P(b)) && \text{Idempotence} \\
&\equiv Q \vee (Q \vee (P(a) \vee P(b))) && \text{Associativity} \\
&\equiv Q \vee ((Q \vee P(a)) \vee P(b)) && \text{Associativity} \\
&\equiv (Q \vee (Q \vee P(a)) \vee P(b) && \text{Associativity} \\
&\equiv ((Q \vee P(a)) \vee Q) \vee P(b) && \text{Commutativity} \\
&\equiv (Q \vee P(a)) \vee (Q \vee P(b)) && \text{Associativity} \\
&\equiv \exists x\, Q \vee P(x) && \exists \to \vee
\end{aligned}
$$

(c) Are each of these equivalences also true assuming our fake equivalences? Yes or no.

i $Q \wedge (\forall x\, P(x)) \equiv \forall x\, Q \wedge P(x)$

ii $Q \vee (\forall x\, P(x)) \equiv \forall x\, Q \vee P(x)$.

Solution:

Yes. All three are easy adaptations of the proofs above.

(d) Do the equivalences proven in (a)-(b) hold in every other domain of discourse? Briefly explain why or why not. **Solution:**

Yes. The fake equivalences can become infinitely long, but the proven equivalences remain true.

8. Find the Bug

Each of these inference proofs is incorrect. Identify the line (or lines) which incorrectly apply a law, and explain the error. Then, if the claim is false, give concrete examples of propositions to show it is false. If it is true, write a correct proof.

(a) This proof claims to show that given $a \to (b \vee c)$, we can conclude $a \to c$.

1. $a \to (b \vee c)$ [Given]
   - 2.1. a [Assumption]
   - 2.2. $\neg b$ [Assumption]
   - 2.3. $b \vee c$ [Modus Ponens, from 1 and 2.1]
   - 2.4. c [$\vee$ elimination, from 2.2 and 2.3]
2. $a \to c$ [Direct Proof Rule, from 2.1-2.4]

Solution:

The error here is in lines 2.2 and 2. When beginning a subproof for the direct proof rule, only one assumption may be introduced. If the author of this proof wanted to assume both a and $\neg b$, they should have used the assumption $a \wedge \neg b$, which would make line 3 be $(a \wedge \neg b) \to c$ instead.

And the claim is false in general. Consider:
a: "I am outside"
b: "I am walking my dog"
c: "I am swimming"
If we assert "If I am outside, I am walking my dog or swimming," we cannot reasonably conclude that "If I am outside, I am swimming" $(a \to c)$.

(b) This proof claims to show that given $p \to q$ and r, we can conclude $p \to (q \vee r)$.

1.$p \to q$	[Given]
2.r	[Given]
3.$p \to (q \vee r)$	[Intro $\vee$ (1,2)]

Solution:

Bug is in step 3, we're applying the rule to only a subexpression.

The statement is true though. A correct proof introduces p as an assumption, uses MP to get q, introduces $\vee$ to get $q \vee r$, and the direct proof rule to complete the argument.

(c) This proof claims to show that given $p \to q$ and q that we can conclude p

1.$p \to q$	[Given]
2.q	[Given]
3.$\neg p \vee q$	[Law of Implication (1)]
4.q	[Eliminate $\vee$ (2,3)

Solution:

The bug is in step 4. Eliminate $\vee$ from 3 would let us conclude $\neg p$ if we had $\neg q$ or q if we had p. It doesn't tell us anything with knowing q.

Indeed, the claim is false. We could have p: "it is raining"
q: "I have my umbrella"

And be a person who always carries their umbella with them (even on sunny days). The information is not sufficient to conclude p.