Discussion Problem

CS 173: Discrete Structures

Problem 1.5. In Discussion Section Booklet

1. Yes, the set of operators $\{\vee, \neg\}$ is functionally complete, as we can express $\wedge$ in terms of these operators:

   $$\alpha \wedge \beta \equiv \neg(\neg\alpha \vee \neg\beta)$$

   So in any formula involving $\wedge$, we can replace subformulas involving $\wedge$ systematically using the above equivalence to get an equivalent formula involving only $\neg$ and $\vee$. For example,

   $$((p \vee q) \wedge r) \vee (\neg s) \equiv \neg(\neg(p \vee q) \vee \neg r) \vee (\neg s)$$

   Since any Boolean function (any truth table) can be expressed using the set of operators $\{\vee, \neg\}$ (by expressing them first using $\{\vee, \wedge, \neg\}$ and removing $\wedge$ using the above conversion), the set of operators $\{\vee, \neg\}$ is functionally complete.

2. $$\neg p \equiv p \uparrow p$$

   When $p = T$, $p \uparrow p = F = \neg p$.
   When $p = F$, $p \uparrow p = T = \neg p$.
   Hence the above equivalence holds. The following truth table also verifies equivalence:

p	$\neg p$	$p \uparrow p$
T	F	F
F	T	T

   Alternate solution: $p \uparrow T$ also is equivalent to $\neg p$,

3. Notice that the truth table entries for $p \uparrow q$ are precisely the negation of what we need, i.e, for $p \vee q$, provided p and q are negated before combining with $\uparrow$.

   So we conjecture:

   $$p \vee q \equiv (\neg p) \uparrow (\neg q)$$

   Since we know how to express negation using $\uparrow$ (as argued above), we conjecture:

   $$p \vee q \equiv (p \uparrow p) \uparrow (q \uparrow q)$$

   The truth table below verifies this:

p	q	$p \uparrow p$	$q \uparrow q$	$(p \uparrow p) \uparrow (q \uparrow q)$	$(p \vee q)$
T	T	F	F	T	T
T	F	F	T	T	T
F	T	T	F	T	T
F	F	T	T	F	F

Alternate Solution: There are many solutions to this problem, of course. Here's an alternative that uses the alternative encoding of $\neg$, as mentioned above, which uses the *constant* T (true).

$$p \vee q \equiv (p \uparrow T) \uparrow (q \uparrow T)$$

This too can be verified using a truth table.

4. The set of operators $\{\uparrow\}$ is functionally complete since we can take any formula involving operators in $\{\neg, \vee\}$ and replace subformulas systematically using equivalent formulas that use only the $\uparrow$ operator. Since the set of operators $\{\neg, \vee\}$ is functionally complete, so is $\{\uparrow\}$.

5. Let us try to express $\uparrow$ using $\downarrow$.

   First, notice that $p \uparrow q \equiv \neg(p \wedge q)$.
   And $p \downarrow q \equiv \neg(p \vee q)$.

   So $p \uparrow q$
   $\equiv \neg(p \wedge q)$
   $\equiv \neg p \vee \neg q$
   $\equiv \neg(\neg(\neg p \vee \neg q))$
   $\equiv \neg((\neg p) \downarrow (\neg q))$ So what's left is to express negation using $\downarrow$. Notice that $\neg r \equiv (r \downarrow r)$. Using this to rewrite negation gives: $p \uparrow q$
   $\equiv \neg((p \downarrow p) \downarrow (q \downarrow q))$
   $\equiv ((p \downarrow p) \downarrow (q \downarrow q)) \downarrow ((p \downarrow p) \downarrow (q \downarrow q))$ (Verify the above using truth tables!) Since we have expressed $\uparrow$ using $\downarrow$ operator only, and given that $\{\uparrow\}$ is functionally complete, it follows that $\{\downarrow\}$ is functionally complete as well.

   Alternate Solution: Another way to express $\uparrow$ using $\downarrow$ is:

$$((p \downarrow F) \downarrow (q \downarrow F)) \downarrow F$$

Note: If you have a different solution that is hard to check, you could use the following Z3 script. Skip this if you find this too complex. Z3 is a SAT solver and is available online at https://rise4fun.com/z3 or at https://compsys-tools.ens-lyon.fr/z3/index.php. In order to check an equivalence, we assert the negation of the equivalence, and expect the SAT solver to say "unsat". If SAT solver says "sat", the answer is not correct. I've put in the solution above in the script below (as well as the alternate solution, commented out). Note that operators are in prefix format rather than infix.

```
(declare-const p Bool)
(declare-const q Bool)
(define-fun stroke ((p Bool) (q Bool)) Bool
 (or (and p (not q))
     (and (not p) q)
     (and (not p) (not q))
 )
)
(define-fun arrow ((p Bool) (q Bool)) Bool
```

```
 (and (not p) (not q))
)
(define-fun conjecture () Bool
(= (stroke p q)
   ; type in solution below to check if it's correct
   (arrow
     (arrow (arrow p p) (arrow q q))
     (arrow (arrow p p) (arrow q q))
   )
   ; here's the alternate solution (commented out)
   ; (arrow
   ;    (arrow  (arrow p false) (arrow q false))
   ;    false
   ; )
  )
)
(assert (not conjecture))
(check-sat)
(get-model)
```