CS 70 Spring 2005 — Discrete Mathematics for CS — Clancy/Wagner — HW 2

Due Thursday, February 3

Coverage: This assignment involves topics from the January 25 and 27 lectures and from sections 3.1 through 3.3 of Rosen.

Administrative reminders: We will accept only unformatted text files or PDF files for homework submission. Your homework submission should include the following information:

Your full name
Your login name
`Homework assignment 2`
Your section number
Your list of partners

To submit your answers to this homework assignment, create a directory named `hw2`, copy your answer file to that directory, `cd` to that directory, and then give the command

```
submit hw2
```

Homework exercises:

1. **(6 pts.) Propositional proof practice**
Prove the proposition

$$(P \Rightarrow (Q \Rightarrow (P \wedge Q))) \Rightarrow (\neg(P \wedge Q) \Rightarrow (P \Rightarrow \neg Q))$$

Your proof may involve subsidiary proofs, or *lemmas* (similar to "helper functions" in Scheme). It may also involve two kinds of *assumptions*. If you're trying to show an implication $E_1 \Rightarrow E_2$, then one valid proof strategy is to assume E_1 and then use this assumption to prove E_2 (but you are only allowed to assume E_1 for purposes of proving the implication; the assumption E_1 is not allowed to survive past the body of the proof that $E_1 \Rightarrow E_2$). Also, if you're trying to show an expression E, you may assume $\neg E$ and then use this assumption to prove E (but again, this assumption must not survive past the body of the proof of E).

Your proof may use only the following inference rules.

* Modus ponens: from the expressions E_1 and $(E_1 \Rightarrow E_2)$, you can infer E_2.
* Modus tollens: from the expression $\neg E_2$ and $(E_1 \Rightarrow E_2)$, you can infer $\neg E_1$.
* Implication construction: from the expression E_2, you can infer $(E_1 \Rightarrow E_2)$ for any expression E_1.

* Contradiction: from the expressions E and $\neg E$, you can infer any expression.
* And-elimination: from the expression $E_1 \wedge E_2$, you can infer E_1 (or E_2).
* Logical equivalence: from the expression E_1, you can infer the expression E_2 if $E_1 \equiv E_2$ (if E_1 and E_2 are logically equivalent, as defined in Lecture Notes 1 or in Rosen).

Here are some examples.

Theorem: $P \Rightarrow (\neg P \Rightarrow R)$
Proof:
1. Assume P (allowed for the purposes of proving $P \Rightarrow (\neg P \Rightarrow R)$)
2. Lemma: $\neg P \Rightarrow R$
   Proof:
   2.1 Assume $\neg p$ (allowed for the purposes of proving $\neg P \Rightarrow R$)
   2.2 $\neg P \Rightarrow R$ (contradiction resulting from steps 1 and 2.1)
   QED.
3. $P \Rightarrow (\neg P \Rightarrow R)$ (implication construction applied to step 2)
QED.

Theorem: $(((P \Rightarrow Q) \Rightarrow Q) \Rightarrow ((Q \Rightarrow P) \Rightarrow P))$
1. Assume $((P \Rightarrow Q) \Rightarrow Q)$ (allowed for the purposes of proving $(((P \Rightarrow Q) \Rightarrow Q) \Rightarrow ((Q \Rightarrow P) \Rightarrow P)))$
2. Lemma: $(Q \Rightarrow P) \Rightarrow P$
   Proof:
   2.1 Assume $(Q \Rightarrow P)$ (allowed for the purposes of proving $(Q \Rightarrow P) \Rightarrow P$)
   2.2 Lemma: P
      Proof:
      2.2.1 Assume $\neg P$ (allowed for the purposes of proving P)
      2.2.2 $\neg Q$ (modus tollens applied to steps 2.2.1 and 2.1)
      2.2.3 $\not(P \Rightarrow Q)$ (modus tollens applied to steps 2.2.2 and 1)
      2.2.4 Lemma: $P \Rightarrow Q$
         Proof:
         2.2.4.1 Assume P (allowed for the purposes of proving $P \Rightarrow Q$)
         2.2.4.2 $P \Rightarrow Q$ (contradiction resulting from steps 2.2.4.1 and 2.2.1)
         QED.
      2.2.5 P (contradiction resulting from steps 2.2.4 and 2.2.3)
      QED.
   2.3 $(Q \Rightarrow P) \Rightarrow P$ (implication construction applied to step 2.2)
   QED.
3. $((P \Rightarrow Q) \Rightarrow Q) \Rightarrow ((Q \Rightarrow P) \Rightarrow P)$ (implication construction applied to step 2)

2. (4 pts.) Simple induction
Prove that $2^n < n!$ for all integers $n \geq 4$.

3. (8 pts.) Another way to multiply
Consider the following Scheme code.

```
(define (product a1 a2)
  (product-helper a1 a2 0) )
```

```
(define (product-helper n1 n2 so-far)
  (cond
    ((= n1 1) (+ so-far n2))
    ((odd? n1) (product-helper (- n1 1) n2 (+ so-far n2)))
    (else (product-helper (/ n1 2) (* 2 n2) so-far)) ) )
```

The result returned by `product`, given two natural numbers as arguments, is the product of the two numbers. Since it only involves addition, subtraction, doubling, and halving, the algorithm provides a practical procedure for multiplication of two integers[1] by hand.

An equivalent specification of the algorithm in pseudocode would be the following:

Algorithm `product`(a_1, a_2):
1. Return `product-helper`$(a_1, a_2, 0)$.

Algorithm `product-helper`(n_1, n_2, s):
1. If $n_1 = 1$, then return $n_2 + s$.
2. If n_1 is odd, then return `product-helper`$(n_1 - 1, n_2, n_2 + s)$.
3. Otherwise, return `product-helper`$(n_1/2, 2n_2, s)$.

(a) Find a formula that relates the values of `n1`, `n2`, and `so-far` at the start of each call to `product-helper` to the product of the `a1` and `a2`, the original arguments of the `product` procedure.

(b) Using your invariant relation, prove that `product` returns the product of its two nonnegative integer arguments.

4. (8 pts.) Reversal of fortune

Consider the following Scheme code. A call to `reverse` with a list as argument returns the result of reversing the list.

```
(define (reverse L)
  (reverse-helper '( ) L) )

(define (reverse-helper so-far L)
  (if (null? L) so-far
    (reverse-helper (cons (car L) so-far) (cdr L)) ) )
```

Prove that `reverse` works. You will need some formal definitions: if $L = (a_0\ a_1\ a_2\ ...)$, then

$$(\texttt{car L}) = a_0, (\texttt{cdr L}) = (a_1\ a_2\ ...), (\texttt{cons x L}) = (x\ a_0\ a_1\ a_2\ ...).$$

You then are to prove that if $L = (x_0\ x_1\ x_2\ ...\ x_N)$, `(reverse L)` returns $(x_N\ x_{N-1}\ ...\ x_1\ x_0)$.

5. (8 pts.) Sprouting forever?

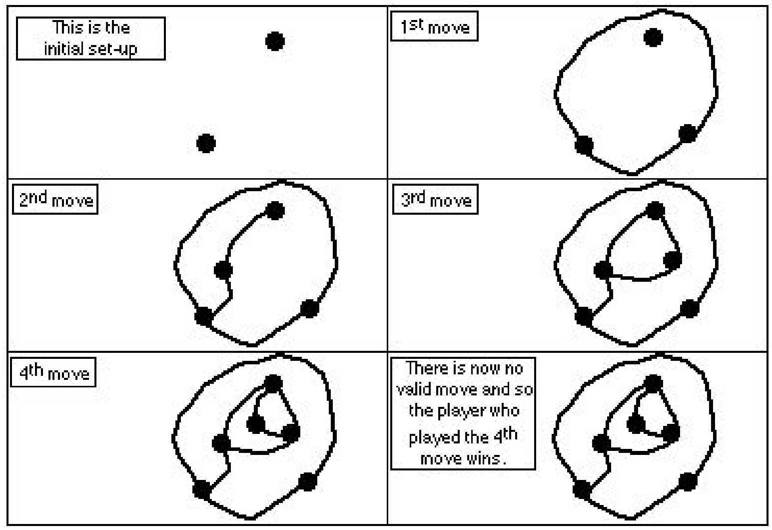
Sprouts is a two-player game played with paper and pencil. Several dots are drawn on the paper. Then the players take turns, each doing the following:

[1] Donald Knuth, in *The Art of Computer Programming*, volume 2, notes that this algorithm was used as early as 2000 BC in Egypt. It is described as "Russian peasant multiplication" since Western visitors to Russia in the nineteenth century found the method in wide use there.

* drawing a line that connects two dots or connects a dot to itself but doesn't touch or cross any other line;
* putting a new dot on this new line, thus separating it into two lines.

No dot can have more than three lines attached to it. The last player that can make a legal move wins.

The figure below shows a sample game.



(a) Prove that any Sprouts game consists of a finite number of moves before someone loses.

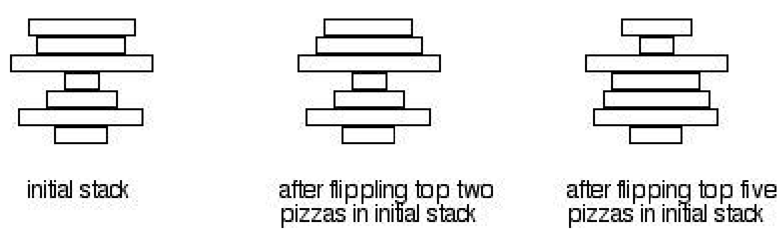
(b) Give as good an upper bound as you can on the worst-case number of moves in a Sprouts game that starts with n dots, and prove your answer.

6. (8 pts.) A floor show

Prove by induction that $\sum_{i=1}^{n} \lfloor i/2 \rfloor = \lfloor n^2/4 \rfloor$. (Recall that $\lfloor x \rfloor$ is the largest integer less than or equal to x.)

7. (6 pts.) A pizza proof

Working at the local pizza parlor, I have a stack of unbaked pizza doughs. For a most pleasing presentation, I wish to arrange them in order of size, with the largest pizza on the bottom. I'm able to place my spatula under one of the pizzas and flip over the whole stack above the spatula (reversing their order). The figure below shows two sample flips.



This is the only move I can do to change the order of the stack; however, I am willing to keep repeating this move until I get the stack in order. Is it always possible for me to get the pizzas in order? Prove your answer.

8. (12 pts.) You be the grader

Assign a grade of A (correct) or F (failure) to each of the following proofs. If you give a F, please explain exactly everything that is wrong with the structure or the reasoning in the *proof*. You should justify all your answers (remember, saying that the claim is false is *not* a justification).

(a) **Theorem 0.1**: *For every $n \in N$, n^2+n is odd.*

Proof: The proof will be by induction.
Base case: The natural number 1 is odd.
Inductive step: Suppose $k \in \mathbf{N}$ and k^2+k is odd. Then,

$$(k+1)^2+(k+1) = k^2+2k+1+k+1 = (k^2+k)+(2k+2)$$

is the sum of an odd and an even integer. Therefore, $(k+1)^2+(k+1)$ is odd. By the Principle of Mathematical Induction, the property that n^2+n is odd is true for all natural numbers n. □

(b) **Theorem 0.2**: *For all $x,n \in N$, if $nx = 0$ and $n > 0$, then $x = 0$.*

Proof: The proof will be by induction.
Base case: If $n = 1$, then the equation $nx = 0$ implies $x = 0$, since $nx = 1 \cdot x = x$ in this case.
Induction step: Fix $k > 0$, and assume that $kx = 0$ implies $x = 0$. Suppose that $(k+1)x = 0$. Note that $(k+1)x = kx+x$, hence we can conclude that $kx+x = 0$, or in other words, $kx = -x$. Now there are two cases:

Case 1: $x = 0$. In this case, $kx = -x = -0 = 0$, so $kx = 0$. Consequently, the inductive hypothesis tells us that $x = 0$.

Case 2: $x > 0$. In this case, $-x < 0$ (since $x > 0$). At the same time, $kx \geq 0$ (since $k,x \geq 0$). But this is impossible, since we know $kx = -x$. We have a contradiction, and therefore Case 2 cannot happen.

In either case, we can conclude that $x = 0$. This completes the proof of the induction step. □

(c) **Theorem 0.3**: *For all $x,y,n \in N$, if $\max(x,y) = n$, then $x = y$.*

Proof: The proof will be by induction.
Base case: Suppose that $n = 0$. If $\max(x,y) = 0$ and $x,y \in N$, then $x = 0$ and $y = 0$, hence $x = y$.
Induction step: Assume that, whenever we have $\max(x,y) = k$, then $x = y$ must follow. Next suppose x,y are such that $\max(x,y) = k+1$. Then it follows that $\max(x-1,y-1) = k$, so by the inductive hypothesis, $x-1 = y-1$. In this case, we have $x = y$, completing the induction step. □

(d) **Theorem 0.4**: $\forall n \in N.\ n^2 \leq n$.

Proof: The proof will be by induction.
Base case: When $n = 0$, the statement is $0^2 \leq 0$ which is true.
Induction step: Now suppose that $k \in \mathbf{N}$, and $k^2 \leq k$. We need to show that

$$(k+1)^2 \leq k+1$$

Working backwards we see that:

$$\begin{aligned} (k+1)^2 &\leq k+1 \\ k^2+2k+1 &\leq k+1 \\ k^2+2k &\leq k \\ k^2 &\leq k \end{aligned}$$

So we get back to our original hypothesis which is assumed to be true. Hence, for every $n \in \mathbf{N}$ we know that $n^2 \leq n$. □