CSU290 Lecture Notes   Lecture 10   2 Oct 2008

Peter Dillinger

More Boolean Logic

We looked at simplifying some boolean formulas (see Boolean Identities sheet). You should practice so that you know all of those simplifications reasonably quickly and reliably.

How many rows in a truth table?   (Two raised to the power of the number of variables. 1 var = 2 rows, 2 vars = 4 rows, 3 vars = 8 rows.)

Generating all possible truth values over a set of boolean variables is like counting in binary:

```
  F F F    000
  F F T    001
  F T F    010
  F T T    011
  T F F    100
  T F T    101
  T T F    110
  T T T    111
```

We constructed a truth table to verify that

```
 (p /\ (q \/ r)) <-> ((p /\ q) \/ (p /\ r))
```

(That is the property that AND distributes over OR. It is also true that OR distributes over AND.)

We also looked at Modus Ponens and Modus Tollens (see sheet).

More Functions on Conses/Lists

We are going to write a function somewhat related to integral calculus. If we treat a list of numbers as the values a mathematical function takes on inputs 0 through the length-1, then the discrete integral of that function would be, using the same representation, a list of numbers each of which is the sum of all the elements up to including the corresponding element from the input:

```
; DINT: rational-list -> rational-list
; Returns the "discrete integral" of the input list.  That is, the returned
; list should be the same length and the nth element should be the sum of
; all elements of the input list up to and including the nth element.
; Examples:
;   (dint '(6 5 3 1)) = '(6 11 14 15)
;   (dint   '(5 3 1)) =   '(5  8  9)
;   (dint     '(3 1)) =      '(3  4)
;   (dint       '(1)) =        '(1)
;   (dint        '()) =         '()
```

Now the question seems to be, how does solving the problem on the CDR of the list help us to solve it overall? In the specific case of '(6 5 3 1), how does (dint '(5 3 1)) = '(5 8 9) help us get '(6 11 14 15)?

The answer is that it helps us get the ’(11 14 15) part if we could add the CAR of the list, 6, to each element of the DINT of ’(5 3 1), ’(5 8 9). Then we just cons the CAR, 6, onto that, ’(11 14 15), to get our result, ’(6 11 14 15).

How do we add some value to each element of a list? We need a helper function. In fact, I chose this example exactly because it requires use of a helper function. Here’s how we might write the helper:

```
; ADD-TO-ALL: rational rational-list -> rational-list
; Adds first parameter to each element of second parameter list.
; Examples:
;   (add-to-all 6 ’(5 8 9)) = ’(11 14 15)
;   (add-to-all 3 ’())      = ’()

(defun add-to-all (v l)
  (if (endp l)
    nil
    (cons (+ v (car l))
          (add-to-all v (cdr l)))))
```

If we test this function, we will see that it works.

Now to use that in writing DINT:

```
(defun dint (l)
  (if (endp l)
    nil
    (cons (car l)
          (add-to-all (car l)
                      (dint (cdr l))))))
```

We can also solve DINT by writing a helper that uses an accumulator that keeps track of the sum so far. Think about that for next lecture.