331 Final

Fall 2010

Details

- 3:30-5:30 Friday December 17th
- LH4
- Comprehensive with more emphasis on material since the midterm
- Study example finals and midterm exams from Fall 2008 and Spring 2010
- Read all assigned material from book and schedule

Example Problem

For each of the following grammars, briefly describe the language it defines in a sentence or two. Assume that the start symbol is S for each and that any symbol found only on the right hand side of a production is a terminal symbol.

(1b) 5 points.

S -> B A

S -> a

B -> S A

A -> a

Example Problem

For each of the following grammars, briefly describe the language it defines in a sentence or two. Assume that the start symbol is S for each and that any symbol found only on the right hand side of a production is a terminal symbol.

(1c) 5 points.

S -> a | aa
S -> a X
S -> b S b
S -> b | bb
X -> S a

Example Problem

Problem five: EBNF to BNF (15)

Assume that you have already been given grammars for the non-terminals <iden> and <expr> which represent identifiers and expressions, respectively. Rewrite the following EBNF grammar in BNF. You may create new non-terminal symbols if you wish.

<statement> ::= <iden> '=' <expr>
<statement> ::= 'IF' <expr> 'THEN' <statement> ['ELSE' <statement>] 'ENDIF'
<statement> ::= 'WHILE' <expr> 'DO' <statement> 'ENDWHILE'
<statement> ::= 'BEGIN' <statement> {';' <statement>} 'END'

Example Problem

(a) Draw a DFA for a real number that satisfies the following description, using the conventions above.

A real number can start with an optional sign which can be "-" or "+" and consists of an integer part followed by a decimal point followed by a fractional part. The integer part can be a single zero or a non-empty sequence of digits that does not start with a zero. The fractional part is a non-empty sequence of digits. Positive examples include 0.0, +0.0, 0.12, 12.3 and -9.87 . Negative examples are: 0, 01.2, -01.2, 3. and 42 .

Identify the start state and all accepting states and label every arc. Since this is a deterministic and not a non-deterministic finite automaton, every arc must have a label which can not be epsilon.

(b) Write a regular expression that corresponds to the DFA, making it as simple as possible. Use parentheses to ensure proper scope or for clarity. (15 points)

Example Problem

Assuming that we've done (define x '((1 (2)) (3))) give a Scheme expression using only the functions car and cdr and variable x that returns the second symbol in the list

Example Problem

Common Lisp has a built-in function maplist. The Scheme counterpart

could be writen as follows:

```
(define (maplist f l)
  (if (null? l)
      null
      (append (f l)
              (maplist f (cdr l)))))
```

[10] What will (maplist list '(1 2 3)) return?

[10] What will (maplist (lambda (x) x) '(1 2 3)) return?

[10] What will (maplist (lambda (x) (list (length x))) '(1 2 3))

Example Problem

Consider a function insert with three arguments: an arbitrary s-expression, a proper list, and a posi- tive integer. The function returns a new list that is the result of inserting the expression into the list at the position specified by the third argument. Note that positions begin with zero. For example,

```
> > (insert 'X '(a b c) 3)
(a b c X d)
> > (insert '(X) '(a b c) 1)
(a (X) b c)
> > insert 'X '(a b c) 0)
(X a b c)
```

Here is an incomplete definition of the function. Give code expressions for <S1>, <S2> and <S3> that will complete it.

```
(define (insert expr lst pos)
   ;; Returns a list like proper list lst but with expr inserted at
   ;; the position given by positive integer pos. e.g.: (insert 'X
   ;; '(a b c) 2) => (a b X c)
   (cond (<S1> (cons expr lst))
        ((null? lst) <S2> )
        (else <S3>)))
```

Python

Write a function weight that takes a single argument and returns its weight, where the weight of a non-list is 1 and the weight of a list of N elements is N plus the sum of the weights of the elements.

```
>>> type(l)
<type 'list'>
>>> type(1)
<type 'int'>
>>> type([])
<type 'list'>
>>> type(1)
<type 'int'>
>>> def is_list(x): return type(x) == type([])
...
>>> is_list([1,2,3])
True
>>> is_list("foo")
False
>>>
```