SAMFYB / FP-150-Notebook

Branch: master ▾ FP-150-Notebook / Lecture_14.md Find file Copy path

SAMFYB update lec 14 6716360 on Mar 6

1 contributor

85 lines (67 sloc) 5.53 KB

Lecture 14 Regular Expressions

Basic Concepts

- Languages
- Acceptor Machines
- Grammars
- Content Free
- Regular Languages
- Regular Expressions
- Finite Automata
- Nondeterministic Pushdown Automata

Regular Expressions Definition

- "a" in the alphabet is a regular expression
- 0 is a regular expression
- 1 is a regular expression
- If `r1, r2` are regular expressions, so is `r1r2` (concatenation).
- If `r1, r2` are regular expressions, so is `r1 + r2` (alternation).
- If `r` is a regular expression, so is `r*` (Kleene Star).

Languages Associated with Regular Expressions

Regular Expression	Language
a	$L(a) = \{a\}$
0	$L(0) = \{\}$
1	$L(1) = \{\epsilon\}$

Regular Expression	Language
$r_1 r_2$	$\{s_1 s_2 : s_1 \in L(r_1), s_2 \in L(r_2)\}$
$r_1 + r_2$	$\{s : s \in L(r_1) ors \in L(r_2)\}$
r^*	$\{s_1 s_2 ... s_n : n \in \mathbb{N}, n \geq 0, s_i \in L(r)\}$

Some Examples

- $L(aa) = \{aa\}$
- $L(ab) = \{ab\}$
- $L((a+1)(a+1)) = \{\epsilon, a, aa\}$
- $L(1 + a + aa) = \{\epsilon, a, aa\}$
- $L((a+b)^*)$ is the set of all strings formed from the alphabet
- $L((a+b)^* aa (a+b)^*)$

Theorem. L is regular iff L^C is regular.

- $L((a+1)(b+ba)^*)$ is the compliment of the last language above.

Time to Write Some Code

```
datatype regexp = Char of char
                | Zero
                | One
                | Times of regexp * regexp
                | Plus of regexp * regexp
                | Star of regexp

(* accept : regexp -> string -> bool
 * req : true
 * ens : accept r s => true if s in L(r)
 *                     false otherwise
 *
 * (helper) match : regexp -> char list -> (char list -> bool) -> bool
 * req : k is total
 * ens : match r cs k => true if cs = p @ s s.t. p in L(r) & k s = true
 *                       false otherwise
 *)
fun accept r s = match r (String.explode s) List.null
fun match (Char a) cs k = case cs of
                            [] => false
                          | (c::cs') => (a = c) andalso (k cs')
  | match Zero cs k = false
  | match One cs k = k cs
  | match (Times (r1, r2)) cs k = match r1 cs (fn cs' => match r2 cs' k)
  | match (Plus (r1, r2)) cs k = (match r1 cs k) orelse (match r2 cs k)
  | match (Star r) cs k = (k cs) orelse (match r cs (fn cs' => match (Star k) cs' k))
```

Issue. The last clause may not necessarily terminate considering `Star 1`.

Fixes

- Require a stronger spec
- Instantly check `cs' <> cs`