Homework 3
CS420 Spring 1999

이 광근

Due: April 3 (토요일), 24:00

1999년 3월 29일

Exercise 1 Write an ML function

```
dfa: string -> bool
```

for deterministic finite automata of regular expression $a(b|c)*(d|e)$. The function when given a string returns true if the string is a sentence of the regular expression, false otherwise.

Here are some hints:

1. derive a determinitic automata for the regular expression
2. define a function $\mathtt{s}_n$: `string -> bool` for each state n
3. state transition $n \xrightarrow{a} m$ is implemented by a call.

□

Exercise 2 Using `ml-lex`, construct a program

```
spelldum: string -> unit
```

that `spelldum("file")` translates a text in the file such that character exchange rules are applied after the rule is mentioned inside the text. The translated output is printed out in the screen.

For example,

```
[e->i] is a good example. But [n->m] is not.
```

is translated into

```
[e->i] is a good ixampli. But [n->m] is mot.
```

The exchange rule is a term in `\[[a-z]"->"[a-z]\]`. The input text is a term in `[a-zA-Z,\-\ \t\n\.\"]+`.

Note that

- Your program can assume that the text has no cyclic exchange rules (like `[a->b]`,`[b->c]` and `[c->a]`).
- No repetitive rules, either (like `[a->b]` and `[a->c]`).
- Rule's transitivity, however, is applied. Thus, if the rules are `[a->b]` and `[b->c]`, then `a` as well as `b` must be replaced by `c` after the second rule.
- Rules are not case-sensitive. `[a->b]` implies both `[a->b]` and `[A->B]`.
- Exchange rules are not effective for other rules. That is, if

  $$\cdots \texttt{[a->b]} \cdots \texttt{[c->a]} \cdots$$

  the `[a->b]` rule does not change the `[c->a]` rule into `[c->b]`.

An input example is in `cain:/home/kwang/cs420/spell.example`. □