Interactive Formal Verification (L21)

1 Regular Expressions

This assignment will be assessed to determine 50% of your final mark. Please complete the indicated tasks and write a brief document explaining your work. You may prepare this document using Isabelle’s theory presentation facility, but this is not required. (A very simple way to print a theory file legibly is to use the Proof General command Isabelle > Commands > Display draft. You can combine the resulting output with a document produced using your favourite word processing package.) A clear write-up describing elegant, clearly structured proofs of all tasks will receive maximum credit.

You must work on this assignment as an individual. Collaboration is not permitted.

Consider reading, e.g., http://en.wikipedia.org/wiki/Regular_expression to refresh your knowledge of regular expressions.

For this assignment, we define regular expressions (over an arbitrary type 'a of characters) as follows:

1. ∅ is a regular expression.
2. ε is a regular expression.
3. If c is of type 'a, then Atom(c) is a regular expression.
4. If x and y are regular expressions, then xy is a regular expression.
5. If x and y are regular expressions, then x + y is a regular expression.
6. If x is a regular expression, then x* is a regular expression.

Nothing else is a regular expression.

▷ Define a corresponding Isabelle/HOL data type. (Your concrete syntax may be different from that used above. For instance, you could write Star x for x*.)
1.1 Regular Languages

A word is a list of characters:

```plaintext
type_synonym 'a word = ''a list
```

Regular expressions denote formal languages, i.e., sets of words. For \( x \) a regular expression, we define its language \( L(x) \) as follows:

1. \( L(\emptyset) = \emptyset \).
2. \( L(\varepsilon) = \{[]\} \).
3. \( L(\text{Atom}(c)) = \{[c]\} \).
4. \( L(xy) = \{uv | u \in L(x) \land v \in L(y)\} \).
5. \( L(x + y) = L(x) \cup L(y) \).
6. \( L(x^*) \) is the smallest set that contains the empty word and is closed under concatenation with words in \( L(x) \). That is, (i) \([\] \in L(x^*) \), and (ii) if \( u \in L(x) \) and \( v \in L(x^*) \), then \( uv \in L(x^*) \).

Define a function \( L \) that maps regular expressions to their language.

```
L :: ''a regexp \Rightarrow ''a word set
```

Prove the following lemma.

```
lemma "L (Star (Star x)) = L (Star x)"
```

1.2 Matching via Derivatives

We now consider regular expression matching: the problem of determining whether a given word is in the language of a given regular expression. You are about to develop your own verified regular expression matcher. We need some auxiliary notions first.

A regular expression is called nullable iff its language contains the empty word.

Define a recursive function \( \text{nullable} \) \( x \) that computes (by recursion over \( x \), i.e., without explicit use of \( L \)) whether a regular expression is nullable.

```
nullable :: ''a regexp \Rightarrow bool
```

Prove the following lemma.

```
lemma "nullable x = ([] \in L x)"
```
The derivative of a language $L$ with respect to a word $u$ is defined to be $\delta_u L = \{ v \mid uv \in L \}$. For languages that are given by regular expressions, there is a natural algorithm to compute the derivative as another regular expression.

▷ Define a recursive function $\Delta c x$ that computes (by recursion over $x$) a regular expression whose language is the derivative of $L x$ with respect to the single-character word $[c]$.

$$\Delta :: \text{'a } \Rightarrow \text{'a regexp } \Rightarrow \text{'a regexp}$$

Hint: nullable might come in handy.

▷ Prove the following lemma.

**lemma** "$u \in L (\Delta c x) = (c#u \in L x)$"

Hint: see the *Tutorial on Isabelle/HOL* and the *Tutorial on Isar* for advanced induction techniques.

▷ Define a recursive function $\delta$ that lifts $\Delta$ from single characters to words, i.e., $\delta u x$ is a regular expression whose language is the derivative of $L x$ with respect to the word $u$.

$$\delta :: \text{'a word } \Rightarrow \text{'a regexp } \Rightarrow \text{'a regexp}$$

▷ Prove the following lemma.

**lemma** "$u \in L (\delta v x) = (v @ u \in L x)$"

To obtain a regular expression matcher, we finally observe that $u \in L x$ if and only if $\delta u x$ is nullable.

**definition** match :: "'a word $\Rightarrow$ 'a regexp $\Rightarrow$ bool" where

"match u x = nullable ($\delta u x$)"

▷ Prove correctness of match.

**theorem** "match u x = (u $\in$ L x)"

▷ Solutions are due on Friday, June 17, 2011, at 12 noon. Please deliver a printed copy of the completed assignment to student administration by that deadline, and also send the corresponding Isabelle theory file to tw333@cam.ac.uk.