LED Tutorial
Dr. Nelson Rushton
9/3/2014

LED is short for the *Language of Effective Definitions.* Essentially, LED is a programming language that interprets certain expressions and programs written in the language of mathematics. LED was designed by Dr. Nelson Rushton of Texas Tech University, and has been implemented by Qianji Zheng.

This document explains how to install and start the LED interpreter, and to evaluate expressions written using natural numbers, tuples, and finite sets, with the operators of equality, addition, multiplication, tuple subscripting, set membership, union, and subset.

1. Installing and starting LED

To install the LED interpreter,

1. Download and install Python 3.4.
2. Download the file LED.zip at http://www.nelsonrushton.net/LED.zip
3. Unzip the contents of LED.zip. This should create a folder called LEDParser-Master, containing two subfolders: Version1 and Version2.

To start the LED interpreter folder Version2 (a subfolder of LEDParser-Master) and double-click the file IDE.py.

2. Syntax of expressions

- A natural number can be written in LED as a sequence of digits. For example, `12` and `6098` are numerals in LED.
- An *n*-tuple, where $n \geq 2$, can be written in LED by writing its respective corordinates, separated by commas and enclosed in parentheses. For example `(10,20,(30,40))` is an LED expression for the the 3-tuple whose first coordinate is 10, whose second coordinate is 20, and whose third coordinate is the pair (30,40).
- A finite set can written in LED by writing expressions for its members, separated by commas and enclosed in braces. For example, `{10,(20,{30})}` is an LED expression for the set whose members are the natural number 10 and the ordered pair (20,{30}). The empty set can be written in LED as `{}`.
- $x + y$ is written in LED as *x* `+` *y*. For example, the value of the LED expression `3+9` is 12.
- $x \times y$ is written in LED as *x* `*` *y*. For example, the value of the LED expression `3*2` is 6.
- $x = y$ is written in LED as *x* `=` *y*. For example, the LED expression `(10,30)=(10,20)` is true, and the LED expression `(10,20)=(20,10)` is false.
- u_i is written in LED as *u*`[`*i*`]`. For example, the value of the LED expression `(10,20,30)[2]` is 20.
- $x \in A$ is written in LED as *x* `in` *A*. For example, the LED expression `1 in {1,2,3}` is true, and the LED expression `4 in {1,2,3}` is false.

- *A* U *B* is written in LED as *A* `U` *B*. For example, the value of the LED expression `{1,2} U {2,3}` is {1,2,3}.
- *A* ⊆ *B* is written in LED as *A* `subeq` *B*. For example, the LED expression `{1} subeq {1,2,3}` is true, and the LED expression `{1} subeq {2,3}` is false.

3. Evaluating expressions using the LED interpreter

Upon starting the LED interpreter (see Section 1 above), a console window should appear with the following message.

```
Enter an expression and hit [return] to get its value.
Enter quit() at the prompt to exit.

>
```

The LED prompt is a greater-than sign (`>`). Expressions can be evaluated by following the instructions in the window. A transcript of a sample session is shown below, giving the answers to the first few odd numbered exercises in Chapter 4.

```
Enter an expression and hit [return] to get its value.
Enter quit() at the prompt to exit.

> (88,99,10)[2] = 99
True

> 1=(1)
True

> (10,20,30)=(10,(20,30))
False

> (10,(20,30)) = ((10,20),30)
False

> ((10,(20,30),40)[2])[1] * 3 = (60,79)[1]
True

>
```