CMPSC 420
Introduction to Compiler Design
Fall 2014

Useless Stack Language 1.0 Specification

Contents

1 Introduction

2 Grammar

Listing 1 Backus–Naur Form for the USL grammar

$\langle program \rangle$ ::= $\langle data \rangle$
  | $\langle data \rangle$ $\langle program \rangle$

$\langle data \rangle$ ::= $\langle number \rangle$
  | $\langle string \rangle$
  | $\langle ident \rangle$
  | $\langle obj \rangle$

$\langle obj \rangle$ ::= '{' $\langle program \rangle$ '}'

USL features only 3 primitive types: numbers, identifiers, and Objects (where an Object is a stack containing more primitive types). Strings, as given by the grammar spec, are syntactic sugar for an object filled with single character identifiers.

3 Execution Model

3.1 Data Structures

USLs execution model involves three structures: the Dictionary, Stack A, and Stack B. Stack A and Stack B are both stacks of primitives, to be used during program execution. The Dictionary is a mapping of identifiers a stack of primitives, such that identifiers may be bound to other symbols in the language, or to Objects, to serve as compound structures. The binding is a stack such that multiple definitions may be scoped and observed; original meanings of identifiers are restored once an accompanying undefine is executed.

3.2 Execution Flow

In preparing for program execution, USL first empties its environment. The Dictionary is set to an empty mapping, and Stack A and Stack B are both emptied. Instructions to execute are loaded into Stack A before execution begins.

During the course of execution, the following process is observed:

1. A primitive is popped off of Stack A. If that primitive is a Number or an Object, it is passed directly to Stack B, otherwise if that primitive is an identifier, that identifier is evaluated.
2. When an identifier is evaluated, if it has a special implementation meaning, that meaning is then executed. Otherwise, if that identifier both does not have a special meaning and is not

given in the Dictionary, it is passed to Stack B. Else, in the case that the identifier can be found in the Dictionary, a Dictionary Evaluation occurs.

3. A Dictionary Evaluation happens as the following:
   - First, look up the identifier in the dictionary.
   - If the result is a Number, place it on Stack B.
   - If the result is another Identifier, evaluate it as per above, unless that Identifier is itself, in which case place that identifier on Stack B.
   - If the result is an object, explode the object.
4. When exploding an object, one reads out its contents in reverse order and pushes each one in turn onto Stack A.

4 Language Keywords

The following identifiers are known to have special meaning in USL:

macro Pop and hold an Object from Stack B. Rotate Stack A with Stack B, and then explode.

+ Pop two Numbers from Stack B. Push the result of adding those Numbers back to Stack B.

- Pop two Numbers from Stack B. Push the result of subtracting the second by the first back to Stack B

***** Pop two numbers from Stack B. Push the result of multiplying the numbers.

/ Pop two numbers from Stack B. Push the result of dividing the second by the first back to Stack B.

def Pop an Identifier and then any primitive from Stack B. Push the primitive as a binding to the Identifier.

undef Pop an Identifier from Stack B. Pop a binding from that Identifier in the Dictionary. If the binding does not exist, or is empty, destroy the entry from the Dictionary entirely.

$ Pop a primitive from Stack B, and push this primitive onto Stack A.

yank Pop an Object from Stack B. Pop any primitive from the Object. Then, push the Object back to Stack B, and then the primitive which was popped.

smush Pop a primitive from Stack B, and then an Object. Push the primitive onto the Object, and then push the Object back to Stack B.

Pop a primitive from Stack B. If that primitive is an Identifier with a Dictionary entry, push back the result of that Identifiers binding. Otherwise, if the primitive was an Object, assemble a new Identifier by gluing all of that Objects contained identifiers together, and push that result back. Otherwise, if the primitive was a Number, push that Number back unchanged.