Project 2B

Abhi Agarwal

Notes

I have divided this section into how I have ordered my program. I have broken this down into each subsection that is defined on the paper.

1.2 Types were already defined in the base that was given to us by the professor, and I have not made any changes to them.

1.3.1 I have defined Defined, Lookup, and Extend under the Type sort. Firstly, I defined the Name sort, which has a symbol for an Identifier. This symbol allows Name to be used as a key in a map. Next I defined a Map sort that contains a Map, which basically allows us to recursively define a key-value pair. This particular definition will allow us to do the k to value v mappings.

1.3.2 Each map should have these auxiliary operations. To solve the problem of having many maps I made a Maps sort, which works as a list. The list contains a single key-value map, and the Maps sort defines MoreMaps, and NoMaps. MoreMaps basically allows to traverse through the Map list, and NoMaps alerts us when we reach the end of the Map.

1.3.3 Defined is implemented by returning either a True or a False if that value exists. It never throws an error.

1.3.4 Lookup is implemented by returning either a True if it exists or a TypeError if it isn't. TypeError is the standard error notation I will be using throughout this program.

1.3.5 Extend is implemented by taking a key-value pair, appending it to the list, and then returning the new list. It uses the MoreMaps part of Maps to take in a Map, and the rest of the list and returns the new list.

1.4 The key-value pair for each Map sort is a mapping from Name to TD, which is a sort to allow us to set up a descriptor list. TD definition is in section 1.5.

1.5 I had a slight issue of not understanding how to implement this. I thought I doing the correct thing until I completed the Program section, and then realized that this

section was not working well. My initial implementation was to parse the Type, and Argument Signature for the function and Members for the class. Then if neither of these were valid then TypeError. I wasn't too sure how to use the keyword Call, and Class here.

1.6

1.7

1.8

1.9

1.10

1.11

1.11.1

1.11.2

1.11.3

1.11.4 LValue(E, id) should be the case here rather than the one defined in the documentation given by the professor. For a given environment

1.12 # 1 Extras

1. There is no need for Unif - we don't have floating types, and we only have integer types. To make sure they are both integers we can use the SameType scheme inside the Boolean sort.

2. Other functions I implemented under the Boolean sort were SingleIntType, SingleBooleanType, SameNoVoid, IntType, BooleanType, StingIntType, IsIdentifier, and IsAny.