Kenneth Hwang

John Meeker

Final Project Write-up Draft – A Whole New World

The notion of worlds is to make a new language construct that redefines the role of a program state. All computation takes place within a world (changes to global and local variables, arrays, objects, etc.), and the world also keeps track of all these changes. Worlds are first class values that can be handled as variables, passed as arguments to a function, and can even be garbage-collected like any other object. A new world can be "sprouted" from an existing world at any given time, and the state of a child world is derived from the state of its parent. However, changes and side effects that occur inside the child do not affect the parent. At the same time, the side effects captured in the child world can be propagated to its parent with a commit operation.

We originally intended our final version of the project to contain the functionality of sprout new worlds from existing worlds, and being able to commit changes from a world into its parent world. Committing would check for changes in the parent world, and if changes were present, the commit would fail. We initially planned the projected to be based off notC.

The final state of the project instead is based off notJs, and the functionality of running code inside a world has been added. Commits now will never fail, and will overwrite everything inside the parent world, even if the parent world contains changes.

Our language extension has the following syntax:

b = thisWorld.sprout();

Above is the syntax for sprouting a world from an existing world. “thisWorld” refers to the world the expression currently resides in. Assuming “a” and “b” are both worlds, the following is also possible:

b = a.sprout();

The above makes a copy of the current state of world “a”, and allows access to the copy through the variable “b”.

b.commit()

The above syntax is for committing a world. It will commit world “b” back into the parent of world “b”.

inside b {

a:=20

}

The above is an example of code execution inside of a world. “b” refers to a world, and the variable “a” is being assigned a value of 20.

The semantics of the extension is has the same semantics as notJs. Additional properties include: a world is simply a new environment with variables mapped to different addresses than that of the parent world. Committing a world replaces the parent’s environment with the child’s environment. Before code is run inside a world, the configuration for the world is loaded.

Our satisfaction with the outcome of this project is minimal. We are unhappy with the amount of work needed for seemly simple tasks. The language extension does not seem to provide any real world applications and is terrible to write and comprehend due to its visually unappealing syntax. Also, much time was wasted on syntax parsing.

If we had the opportunity to redo things differently, we would have tried to obtain significantly more instructor feedback consistently throughout the duration of the project. We would also pick a project that could be based off a simpler language so the complexity of existing features in the language would not interfere with the project requirements; this also would have allowed us to have access to a working solution of the base language earlier.