HW06

Section02

21600069 Minji Kim

1. **Describe your understanding about continuations. Provide examples and references you read and found.**

“Continuations” is the type of flow control structures. It is where a function “continues on” to. Lambda(x) is create a function. It appears to the entire stack-frame of a process immediately after it has returned. Continuations is bookmarking place your code, place your stack. Save both code location, stack frames.

Continuations have a mechanism to (1) save the overall your current stack, (2) re-instantiate it. It can save the whole stack for later.

For example, multithreading, try-catch use this continuation concept.

This is a code example about continuations.

(define the-continuation #f)

(define (test)

(let ((i 0))

(call/cc (lambda (k) (set! the-continuation k)))

(set! i (+ i 1))

i))

call/cc calls first function. And pass a variable. In this code the-continuation is variable. When the function argument assigns the continuation the-continuation will be variable.

1. **Explain about call/cc in detail. Provide examples and references you read and found.**

call/cc is call-with-current-continuation. It used as a flow control, like continuations. It is same as takes a “snapshot” of current control. And the object of continuation is a first-class value. When this continuation object is applied, before continuation is deleted and applied continuation is restored. So the program continue run at that point that continuation was captured. And then the argument of the continuation becomes the return value of this call-with-current-continuation invocation.

This is a code example about call/cc.

(let (

(my-val

(call/cc

(lambda (the-continuation)

The-continuation))))

(if (procedure? my-val)

(begin

(display “First time through\n”)

(display “my-val is a continuation object\n”)

(my-val 5))

(begin

(display “Second time through\n”)

(display “my-val is “)

(display my-val)

(newline))))

procedure? is same as “is there a continuation?”

so, first, In (if (procedure? my-val), my-val is a continuation so it will implement.

And then, (my-val 5) invokes the continuation, and then occur re-instating in the stack. Then, returning “5”. Second, (if (procedure? my-val) is run false branch,

So the Result is

(First time through

my-val is a continuation object

second time through

my-val is 5)

1. **Explain about Continuation Passing Style (CPS) with real code examples from the internet or any references. You must explain the example code in your own words.**

Continuation-passing style is a style of programming that pass the control in the form of a continuation. It is contrasted with direct passing style. CPS function have an extra argument. That is continuation. When we invoke a CPS function, the calling function required to supply a procedure to be invoked with the “return” value.

; continuation-passing style

(define fact-cps

(lambda (n k)

(cond

[(zero? n) (k 1)]

[else (\*(fact-cps (sub1 n)

(lambda (v)

(k (\* v n))))])))

(define fact (lambda (n) (fact-cps n (lambda (v) v))))

In this code, k is procedure of one argument like lambda. When n is zero, k (continuation) is 1, turning the value. Fact-cps has two argument. Second argument is continuation. And then when we input (fact 5) it can occur

| (fact 5)

| (fact-cps 5 #<procedure>)

| (fact-cps 4 #<procedure>)

| (fact-cps 3 #<procedure>)

| (fact-cps 2 #<procedure>)

| (fact-cps 1 #<procedure>)

| (fact-cps 0 #<procedure>) ;; #<procedure> means continuation.

So, we get result 120.

1. **Recently, continuation is adopting in Java JDK. Please watch the following YouTube video about it.**

Why continuations are coming to Java:

Then, answer/discuss the following questions (in one page full, more than 500 words).

1. **After watching the video, what were the major issues adopting continuations in Java JDK as you think?**

I think it has a problem about memory leaks. Introducing continuations in java opens the gate to memory leaks, another gate. We can have a piece of code that starts executing, holds on to some object and never terminates. And it mentioned that there is memory usage when using it. Since it can’t keep calling it in the future, do it need to load the executed stack to continue running? Is also issues adopting continuations in Java JDK. In the past, there are cases where coroutines are names different from sequential names separated by on-shot. However, the nomenclature here in a continuous world is a problem.

1. **Discuss and explain Why Oracle wanted to adopt continuations in Java JDK?**

Reactive programming is referred to as asynchronous programming. It has some problems. According to ted talker, when we suppose, original method was not simple, but we actually want to do some control flow. We want to branch the result of the first compute and then loop over the second core and we simply cannot use existing Java mechanism if we use ‘Completable Future’. Another problem is exception. When we use ‘Completable Future’ executed in a completely different thread. This makes debugging very hard. Biggest problem is return type. Stack traces do not give us the actual context of a computation. So there are two separate walls that can’t interoperate probably the one the async world and the synchronous blocking world. The language that we know not only handle input and nondeterminism, they have this built-in notion of threads of processes and blocking and it is a good abstraction from a programming perspective. Because the kernel threads is too heavy weight. So, I think oracle want to change how we program, continuation adopt in java JDK that as part of Project Loon in Oracle. This project is Java intended to add continuations and fibers to the JDK.

1. **Discuss why adding new features such as continuations and first-class functions in the existing languages is important and what could be the pros and cons in general.**

When we add new features such as continuations and first-class functions in the existing languages is important because when we add these features it allows developer to create a wider variety of expressions for use in the language. It can also allow to express things that cannot be represented by existing expressions alone. Pros of these new features. It allows developers to control flow of instructions. It can provide useful abstraction.

One area that has seen very useful this continuation is web programming. The use of continuations shields the developer from stateless nature of the HTTP protocol. However, it has cons about these new features. It can use in specialized languages. In other language that none specialized languages, implementing continuations are quite hard. Stack-bases languages basically have to copy the whole stack every time we create a continuation. Those languages can implement certain continuation-like features, don’t break the basic stack-based model, a lot more efficiently than the general case, but implementing generalized continuations is quite a bit harder and not worth it. And adding these new features might be cons because if developer not familiar enough this feature, it is a wasting time behave. And they have to go through the trouble of re-write of compilers, interpreters.