ACM SIGPLAN Continuation Workshop

Saturday, September 24, 2011
Tokyo, Japan (co-located with ICFP)

Session 1 (chair: Yukiyoshi Kameyama)

9:00–10:00 **Continuations and classical logic: using continuations as a tool for logic**
(invited talk)
Koji Nakazawa (Kyoto University)

It is well known as the Curry-Howard isomorphism that there is a neat correspondence between logical systems and typed calculi, in particular, the intuitionistic natural deduction and the simply typed lambda calculus. In his paper in 1989, Griffin showed that the correspondence is extended to classical logic and calculi with control operators, and then some typed calculi based on classical logic have been proposed and studied from viewpoints of both logic and programming languages.

In this talk, I show how continuations relate to classical logic, and that we can use ideas from continuations to prove a fundamental property of logic, that is, normalization theorem for some proof systems of classical logic.

10:00–10:30 Tea break

Session 2 (chair: Tiark Rompf)

10:30–10:45 **Visualizing continuations**
Naoki Takashima, *Yukiyoshi* Kameyama

Direct manipulation of delimited continuations allows one to write elegant and modular programs. However, it is often hard for beginners to understand their behavior due to their semantical difficulty. To ease such a burden, we have designed a new language Redex for *visualizing* delimited continuations. It has nested (multi-prompt) delimited-control operators and a serialization mechanism. The latter gives the source-term representation, rather than binary representation, of any represented values in the language so that one can see the delimited continuations at any time of execution of a program. We believe that such a feature is very useful for learning delimited continuations.

10:45–11:00 **Demonstration of Continuation based C on GCC**
Shinji Kono

We have implemented a C-like Continuation based programming language. Continuation based C, CbC was implemented using micro-C on various architectures, and we have tried several CbC programming experiments. Here we report a new implementation of CbC compiler based on GCC 4.5. Since it contains full C capability, we can use both CbC and C.

11:10–11:35 **Using delimited continuations for distributed computing with the CIEL engine**
Derek G. Murray, *Malte* Schwarzkopf, *Christopher* Smowton, *Steven* Smith, *Anil* Madhavapeddy, *Steven* Hand

CIEL is a universal execution engine for distributed computation, designed to achieve high scalability and reliability when run on a commodity cluster. CIEL supports the full range of MapReduce-style computations, and additionally Turing-powerful data-dependent control-flow that permits efficient, fault-tolerant evaluation of iterative and dynamic programming problems that are difficult to express in a pure MapReduce framework. CIEL also has a clear separation between the execution engine and the programming language interfaces, and so in this talk I will describe the integration of delimited continuations (Scala, OCaml), monadic workflow (Haskell), pure continuations (Stackless Python), and manual callbacks (Java).

11:35–12:00 **Swarm: transparent scalability through portable continuations**
James Douglas

Transparent scalability is an elusive characteristic sought for successful software projects which inevitably outgrow themselves. A common way to approach the design of such applications is with the MapReduce pattern, which requires considerable foresight into how the application can be broken down into the functional map and reduce operations. A problem with this and similar approaches is the investment required at the beginning of development; the problem domain must be carefully analyzed and a solution crafted to support the predicted scalability needs. It would be preferable if applications could be developed simply and cheaply, then later, when necessary, made scalable without reworking the existing source code. We present an approach to building transparently scalable applications using Swarm, a framework which enables code execution to "follow the data" within Scala's serializable delimited continuations. Swarm abstracts the location of data across a distributed system from the developer, eliminating costly architectural and modeling requirements of popular distributed computing patterns and frameworks. We explain the design of an example implementation of a Twitter-like Web application which uses Swarm's continuation-passing style collections, and show how the developer is unburdened by the complexity of scalability. We demonstrate how this Swarm-based application can be transparently scaled without requiring changes to the code or accommodation by the architecture.

12:00–13:30 Lunch break

Session 3 (chair: Chung-chieh Shan)

13:30–14:30 **Continuation semantics in linguistics** (invited talk)
Mats Rooth (Cornell University)

For decades, typed lambda calculus has been an essential part of the toolkit for work on semantics in theoretical linguistics. While practitoners in linguistics have been aware that the same logical and type-theoretic methods are used in theoretical computer science, the advantages of this in importing ideas into linguistics are only starting to be cashed out. The chief success so far is the "continuation semantics" for linguistic phenomena including scope and coreference. In this talk, I will explain the linguistic intuition about continuation semantics for scope, and look at some of my own research on intonational focus and ellipsis. I will also talk about using lambda calculators in teaching, and anticipated benifits of using CS-derived ideas in linguistic semantics.

14:35–15:00 **'Focus movement' by delimited continuations**
Daisuke Bekki*, Kenichi* Asai

In the past 10 years, the application of the notion of continuations to natural language semantics has been pursued in order to capture non-local aspects of semantic composition. The phenomenon of "focus" is an example of such aspects, in which the "focused element" induces a universal quantification which refers to the meaning of the whole sentence. In this talk, we will first introduce a theory of the meta-lambda calculus, a kind of two-level typed lambda calculus, to define a monadic translation by which shift/reset operators are defined via continuation monad. We then introduce Bekki and Asai's analysis of focus in which focus contains a shift operator and the adverbial "only" denotes a reset operator. Such a compositional encoding of focus becomes possible through the clear semantics of the meta-lambda calculus based on category theory.

15:00–15:30 Tea break

Session 4 first half (chair: Oleg Kiselyov)

15:30–15:55 **Modular rollback through free monads**
Conor McBride*, Olin* Shivers*, Aaron* Turon

Control operators prove to be an excellent tool for decoupling concerns, and in particular for separating error repair or user interaction from the processing of correct input. In a paper to appear in this year's ICFP, Shivers and Turon give one such use case, a programming pattern for "modular rollback through control logging." Using this pattern, an input processor can be written in a direct way, without any knowledge of a user's ability to back up and alter the input, or a repair module's ability to fix errors. In this talk, we will explore the theoretical underpinnings of the programming pattern, using free monads and Filinski's reify/reflect to factor the implementation. We will show, in particular, that the pattern can be seen as a particular mode of use of monadic representation.

15:55–16:20 **Yield, the control operator: applications and a conjecture**
Roshan P. James*, Amr* Sabry

In previous work, "Yield: Mainstream delimited continuations" (TPDC 2011), we presented a generalized version of the yield control operator that was distilled from studying yield operators of various programming languages. In this brief abstract,

1. we extend that presentation to establish the connection of yield with dynamic binding, dynamic scope and generalized stack inspection in the spirit of Kiselyov et al (SIGPLAN Not. 2006),
2. we outline a lightweight workflow infrastructure in the spirit of Lu and Gannon (eScience 2008) and
3. we provide a yield monad transformer that allows yield to be composed with other effects.

Finally, we pose a question of considerable theoretical interest: do delimited continuations expressed using yield in combination with session types shed light on answer-type polymorphism?

16:20–16:45 **Correctness of functions with shift and reset**
Noriko Hirota, *Kenichi* Asai

Although shift and reset have become used to write various interesting functions, the understanding of those functions is not always simple. As an attempt to better understand their behavior, we formalize and prove correct some functions written with shift and reset in Coq. Building on Sozeau and Kiselyov's formalization of shift and reset using Generalized Continuation Monad, we first formalize Kameyama and Hasegawa's axioms for shift and reset in Coq. We then write a few functions in monadic style and prove them correct using Kameyama and Hasegawa's axioms and the standard monad laws. By carefully not unfolding the definition of monadic operators, we can effectively prove correctness of functions in direct style. We report on two case studies of this approach: reverse and times, and mention that non-trivial generalization of hypothesis is required to properly characterize the behavior of continuations.

16:45–16:55 Short break without tea

Session 4 second half (chair: Amr Sabry)

16:55–17:20 **The limit of the CPS hierarchy**
Josef Svenningsson

We present a language which we refer to as *the limit of the CPS hierarchy*. It allows for an unbounded number of levels of continuations. We present a semantics in the form of an abstract machine.

17:20–17:45 **Non-deterministic search library**
Kenichi Asai, *Chihiro* Kaneko

Non-deterministic programming has been used as a non-trivial application of (delimited) continuations. We report on our experience of using OchaCaml, an extension of Caml Light with (polymorphically typed) shift and reset, to write a search problem using non-deterministic operators. We provide a library for non-deterministic operations implemented using shift and reset and show how it enables us to write a search problem in direct style, using party puzzles as a concrete example.

Thanks to the Continuation Workshop program committee:

- Kenichi Asai (Ochanomizu University, Japan)
- Małgorzata Biernacka (University of Wroclaw, Poland)
- Hugo Herbelin (PPS-πr^2, INRIA, France)
- Oleg Kiselyov
- Julia Lawall (University of Copenhagen, Denmark)
- Tiark Rompf (EPFL, Switzerland)
- Hayo Thielecke (University of Birmingham, UK)

Thanks to the Continuation Workshop organizers: Yukiyoshi Kameyama (University of Tsukuba) and Oleg Kiselyov. Thanks to the ICFP organizers, especially

- Manuel Chakravarty (University of New South Wales)
- Zhenjiang Hu (National Institute of Informatics)
- Soichiro Hidaka (National Institute of Informatics)
- Gabriele Keller (University of New South Wales)
- Derek Dreyer (Max Planck Institute for Software Systems)

Welcome and enjoy!
Chung-chieh Shan (Cornell University), program chair