The University of Nottingham

Finally, a Simple Semantics

Joel Wright and Graham Hutton
University of Nottingham

This Talk

Reasoning about exceptions is a tricky but important topic

- A simple language to explore the semantics of exceptions
- Define a 'finally' operator

Exceptions & Interrupts

The University of Nottingham

- What are exceptions?
  - Events that occur during the execution of a program that disrupt the normal flow of instructions
  - Arise as a direct result of the current operation
- What are interrupts?
  - Also events that disrupt the normal flow of instructions
  - Arise as a result of the program's environment

Our Approach

- A simple language
- High level semantics
- Compiler
- Low level machine
- Compiler Correctness
- Behaviour Proofs

The Language

```
Expr = Val Int
     | Throw
     | Expr + Expr
     | Catch Expr Expr
     | Expr ; Expr
```

Simplest setting in which to study exceptions

Finally

The University of Nottingham

"Do x, then whatever happens do y"

- Not used for handling exceptions
- Performs "clean up" operations
- Propagates the exception

Finally (Attempt 1)

```
x ; y
```

- If x completes successfully then y is run
- If x raises an exception y is NOT run

Finally (Attempt 2)

```
(Catch x y) ; y
```

- If x executes successfully y is run
- If x raises an exception y runs TWICE
- If x raises an exception it is not propagated

Finally (Attempt 3)

```
(Catch x (y ; Throw)) ; y
```

- If x executes successfully y is run
- If x raises an exception y is run and the exception is re-raised

Adding Interrupts

- Worst Case Scenario
  - Any part of an expression may evaluate successfully or fail
  - Demonic Environment

Is finally still correct?

```
(catch x (y ; throw)) ; y
```

- NO!
- We can show evaluations which never execute y
  - Can this problem be solved by redefining finally?
  - Can we modify our language to be “interrupt safe”?

Interrupt Scoping

```
Expr = …
     | Block Expr
     | Unblock Expr
```

- Two scoping operators
- Control the points at which interrupts may be delivered

Finally with Interrupts

```
Block ((Catch (Unblock x) (y ; Throw)) ; y)
```

- Only the x is executed unblocked
- Exception handling mechanism protected

Further Details

This talk has been a very high level overview of work ranging over the past 3 years. All the details will be available soon in our upcoming paper

"Finally, a Simple Semantics"

http://www.cs.nott.ac.uk/~jjw
http://www.cs.nott.ac.uk/~gmh