Linear ML
Efficient linear types for human beings

Iain Lane

School of Computer Science
University of Nottingham

9 July, 2011
Outline

Introduction

Linear logic

Linear ML

Demo!

Fin
Introduction

- Internship at the beginning of the year (January–April)
- Funded by Horizon project (www.horizon.ac.uk)
Project

- Work on a new programming language: Linear ML
- Got some coverage on the internets
  - Reddit
    (http://www.reddit.com/r/types/comments/g118j/linearml_is_a_programming_language_designed_to/)
  - Ycombinator
    (http://news.ycombinator.com/item?id=2310587)
This talk

- Introduction to linear logic & linear type systems
- Introduction to Linear ML
- Demo, if time
Outline

Introduction

Linear logic

Linear ML

Demo!

Fin
First proposed by Girard in 1987 (Girard is also the inventor of the mustard watch)
The connectives of linear logic have meanings in terms of parallel and concurrent computation
Traditional logic encourages casual use of assumptions. Duplication is free. In linear logic it is restricted via alterations to the weakening and contraction rules
Beginning to be used in OS research: Microsoft “Singularity” project uses linear typing for message passing
Linear logic
New connectives

- & "choose from A and B"
- ⊕ "either A or B"
- $A \otimes B$ "both A and B"
- $A \multimap B$ "A linearly implies B" or "consuming A yields B"
- !A "bang A"
Variables

Two kinds of variables

- Linear: $A$
- Nonlinear (or sometimes “intuitionistic”): $A$
Some inference rules

both

\[
\frac{\Delta \vdash A \quad \Delta' \vdash B}{\Delta, \Delta' \vdash A \otimes B} \quad \otimes I \quad \frac{\Delta \vdash A \otimes B \quad \Delta', \iota A, \iota B \vdash C}{\Delta, \Delta' \vdash C} \quad \otimes E
\]

choose

\[
\frac{\Delta \vdash A \quad \Delta \vdash B}{\Delta \vdash A \& B} \quad \& I \quad \frac{\Delta \vdash A \& B}{\Delta \vdash A} \quad \& E_1 \quad \frac{\Delta \vdash A \& B}{\Delta \vdash B} \quad \& E_2
\]
either

\[
\begin{align*}
\Delta \vdash A & \quad \oplus l_1 \\
\Delta \vdash A \oplus B & \\
\end{align*}
\]

\[
\begin{align*}
\Delta \vdash B & \quad \oplus l_2 \\
\Delta \vdash A \oplus B & \\
\end{align*}
\]

wand

\[
\begin{align*}
\Delta, i.A \vdash B & \quad \Diamond l \\
\Delta \vdash A \rightarrow B & \\
\end{align*}
\]
structural

\[
\frac{\Delta \vdash B}{\Delta, ?A \vdash B} \quad \text{Weakening} \quad \frac{\Delta, ?A, ?A \vdash B}{\Delta, ?A \vdash B} \quad \text{Contraction}
\]

bang

\[
\frac{?\Delta \vdash A}{?\Delta \vdash !A} \quad \frac{\Delta \vdash !A}{\Delta, ?A \vdash B} \quad \frac{\Delta, \Delta' \vdash B}{!E}
\]
Example
A packet-response system

Given three types: Packet, Response and Command.
A packet can either elicit a response or issue a command

\[ \text{Packet} \vdash \text{Response} \quad \text{Packet} \vdash \text{Command} \]
Given two distinct packets, we can produce both a response and a command

\[ \mathbb{Packet}, \mathbb{Packet} \vdash \text{Response} \otimes \text{Command} \]

Given one packet, the server can decide whether to send a response or elicit a command

\[ \mathbb{Packet} \vdash \text{Response} \& \text{Command} \]
Linear types?

- Variables must be used exactly once
- Enforced by going out of scope after they are used
- Useful for expressing ownership
Example

The code

```java
write "Enter your name:"
string name = readLine
writeLine (sprintf "Hello %s" name)
```

Works just as you’d expect.
Example

BUT, the code

```java
write "Enter your name:"
string name = readLine
writeLine (sprintf "Hello %s" name)
writeFile "e.v.l.hackers.log" name
```

Fails. Line 4: Variable 'name' not in scope. name was consumed by the first writeLine.
Outline

Introduction

Linear logic

Linear ML

Demo!

Fin
Linear ML

- Linearly typed ‘dialect’ of ML (uses OCaml-like syntax)
- Compiled to LLVM
- Statically typed
- Strict evaluation order
Linear ML
Linear types imply

- Messages passed between threads without copying
- Majority of operations performed in-place
- No need for GC
Introduction

Linear logic

Linear ML

Demo!

Fin
Outline

Introduction

Linear logic

Linear ML

Demo!

Fin
The end

https://github.com/pikatchu/LinearML