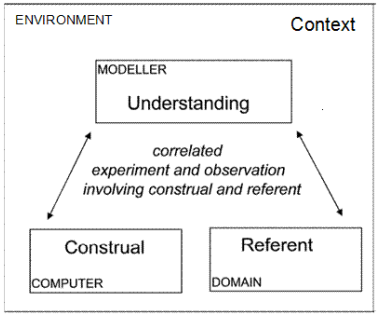
Agenda suggested by Lab 1 …

Making a multi-paradigm programming environment coherent is a fundamental unresolved research problem.

Dealing with the relationship between the operational state-based semantics and the declarative denotational semantics is a challenge for classical programming.

The emphasis in classical computer science is on how meanings can be made formal and non-negotiable ... in contrast modern practices demand ways of thinking about computing that acknowledge its dependence on specific environments and human interpreters.

Empirical Modelling as *Construction*



… contrast this with ….

"The classical answer" to ***What is a program?***

A program is a recipe for action that computes an input-output relationship

Turing's machine model

Machine models of a computer always have

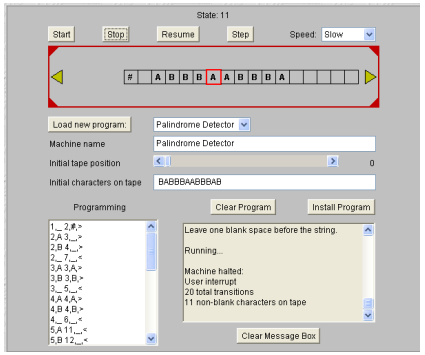
- means to store data
  - e.g objects in Java, files and variables in UNIX
- means to manipulate data
  - e.g. methods in Java, processes in UNIX
- ways to program data manipulation
  - e.g. JAVA programs, UNIX shell scripts

The Turing Machine model (1936)

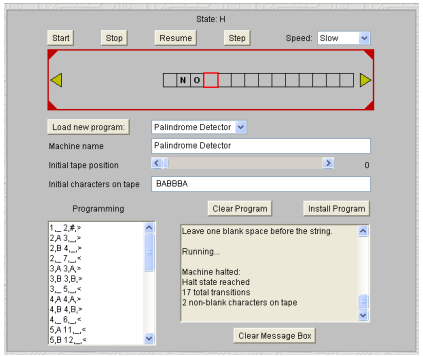
- store is represented by an unbounded tape
- processor is represented by a read/write head
- program is represented by a set of rules

Suzanne Skinner (1996) / Britton (2011)
Java applet simulator at:
http://ironphoenix.org/tril/tm/

A Turing computation in progress … recognising a odd length palindromes



A Turing computation in the halt state … rejecting a non-palindromic string



The Church-Turing thesis

There is no computational model that is in principle more powerful than the Turing machine …

… all algorithmic data processing is equivalent to Turing computation

… by this criterion, very simple notations can define "a full programming language"

A **procedural** program explicitly expresses a recipe as a sequence of actions …

A problem with procedural programs …

Procedural variable

- has a value that can be elusive e.g. when debugging

- always changing, possibly in ways that are hard to track

Procedural version of isprime

```
func factors {
    para n;
    auto r, result;
    result = [];
    for (r=1; r<=n/2; r++)
        if (n % r == 0) result = result // [r];
    return result;
}

func isprime {
    para n;
    return ((factors[n])# == 1);
}
```

Program (e.g.) by specifying the required input-output relation in a mathematical form:

```
out = f(in)
```

This is called **functional programming ("FP")**.

FP exploits a special-purpose interpreter that can compute the function f

FP uses very powerful operators ("λ - calculus") in order to frame the function f

Functional programming (FP)

A "functional" program to compute prime numbers:

```
factors n = [r | r<-[1..n div 2]; n mod r = 0]
isprime q = (# factors q) = 1
```

functional ≡ based on specifying functions
The functions in this context are
factors() and *isprime()*

The programming language is **Miranda**

Procedural version of isprime

```
func factors {                          factors n = [r | r<-[1..n div 2]; n mod r = 0]
    para n;
    auto r, result;
    result = [];
    for (r=1; r<=n/2; r++)
        if (n % r == 0) result = result // [r];
    return result;
}

func isprime {                          isprime q = (# factors q) = 1
    para n;
    return ((factors[n])# == 1);
}
```

Key virtues of declarative programming …

it hides internal states of the computation

have **referential transparency**

frame computational problems in terms of the external domain, not the computer

BUT issues for declarative programming …

Makes interaction tricky

'lazy evaluation' / dataflow as potential solutions

Supporting rich input-output challenging cf. oxo.m

Legacy of the TM concept of computation:
a highly abstract conception of programming

Not well-suited to "emerging computing"

- diverse and rich contexts for computer use
  - non-standard devices, modes of interaction
  - reactive systems
  - real-time, distributed computing, concurrency

- new challenges for software development …

Legacy of the TM concept of computation:
a highly abstract conception of programming

Not well-suited to "emerging computing"

- new challenges for software development …
  - computer + devices + human
  - team work, user participation in design
  - computer as instrument

  Need software that is comprehensible and manipulable even by the non-specialist / even whilst its being constructed

Techniques to help address these goals …

object-orientation
agent-based analysis and conception of systems
design patterns
service-oriented architecture
spreadsheet principles

Going beyond classical programming ….

Characteristics of tools to be introduced in the module ... they are concerned with modelling in which we

- *observe meaningful things*
- *adopt a **constructivist** stance*
- *exploit an **empirical** approach*

that we wish to reconcile / can be reconciled with the more abstract, rationalist, theoretical framework that characterises classical computer science

Reconceptualise by **introducing the human dimension** ... key shift in emphasis towards questions such as:

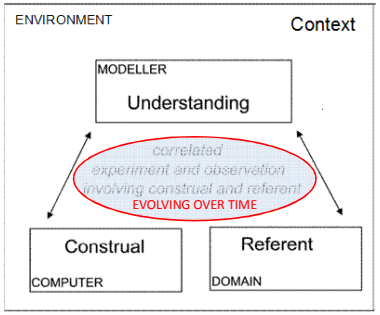
*? what is the **experience** of the people engaging with Turing computation, procedural programs, functional programs etc.*

Consider people's experience ('programmers', 'users', 'modellers' or 'analysts' etc.) with reference to

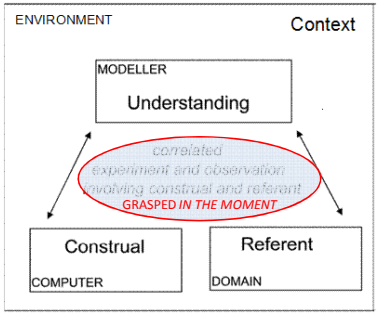
* What are the significant things that they observe?
* How are they able to interact and manipulate?
* What is the context for their interaction and interpretation?

when they are engaged in some variety of programming / model-building activity.

EM as *an experience of construction*



EM as *construction of an experience*



The word **experience** is a key word in Empirical Modelling, and is being used in a very distinctive way e.g. the financial modeller

experiences the current financial situation
"Phwor! I might be about to make a lot of £££s here …"

experiences the numerical patterns in the cells of the spreadsheet
" … that number looks a bit big for that cell"

experiences the current context
"… now I can buy my boyfriend a Ferrari … "

experiences the connection between all these experiences
"figures on screen / money in world / boyfriend ecstatic"