

Syntonicity and the psychology of programming

Stuart Watt

Knowledge Media Institute and

Department of Psychology

Open University

Overview of the session

- Theory and background
- Syntonicity and the psychology of programming
- The ‘Syntonicity Hypothesis’
- Case study 1: Prolog
- Case study 2: Logo, StarLogo, Playground, KidSim, and Cocoa
- Implications, thoughts, and future work

Background

- Common-sense psychology
 - Piaget, Carey, Wellman
 - Faculties for physical and psychological reasoning
- Logo: identifying with the turtle
- The big analogy:
 - Kay's use of enactive reasoning
 - Our use of common-sense psychological reasoning
- More than mental models

Disclaimer

- This seminar is about work that is not yet 'finished' and the ideas are not yet mature. Comments, thoughts, and suggestions will be most welcome. But please bear in mind that these suggestions are tentative, and further work and discussion is needed
- Also, this work puts a different interpretation on the success and failure of some systems. The designers of these systems should not be taken as necessarily agreeing with these interpretations

Syntonicity

- Introduced by Papert (but derived from Freud)
 - “body syntonic” ideas are those which are compatible with one’s own feeling of being in a body (Logo)
 - “ego syntonic” ideas are those which are compatible with one’s own experience of having a mind (Logo, too)
- Syntonicity means we take something as a psychological system rather than a physical one

Identifying with computers and programs

- Aspects of common-sense psychology in programming
 - Identifying with the computer
 - Identifying with the operating system
 - Identifying with the interpreter
 - Identifying with the compiler
 - Identifying with the language
 - Identifying with a program statement or expression (imperative)
 - Identifying with a program procedure or function (imperative)
 - Identifying with an object (object-oriented)
 - Identifying with a clause (declarative)
 - Identifying with a process (data flow)

The Syntonicity Hypothesis

- Can people identify with computers?
- The Syntonicity Hypothesis. Programs and execution models are at least partly psychological, rather than being purely computational or physical. That is, people think about the behaviour of a program in mentalistic terms, rather than formal, logical, mathematical, or physical ones
- Corrolary 1. The more syntonic the programming language, the easier it will be to learn
- Corrolary 2. Execution models will be easier to grasp if people can identify with them

A brief digression: metaphor and syntonicity

- Syntonicity enables a special kind of metaphor
 - Between others and ourselves
 - Metaphors that are “image schematic”
- Importance of similarity to syntonicity
- Importance of embodiedness to metaphor (Johnson, 1988)
- Some metaphors are better than others
 - Metaphors that people can identify with
- Important to grasping skills (apprenticeship)

Another brief digression: learning skills

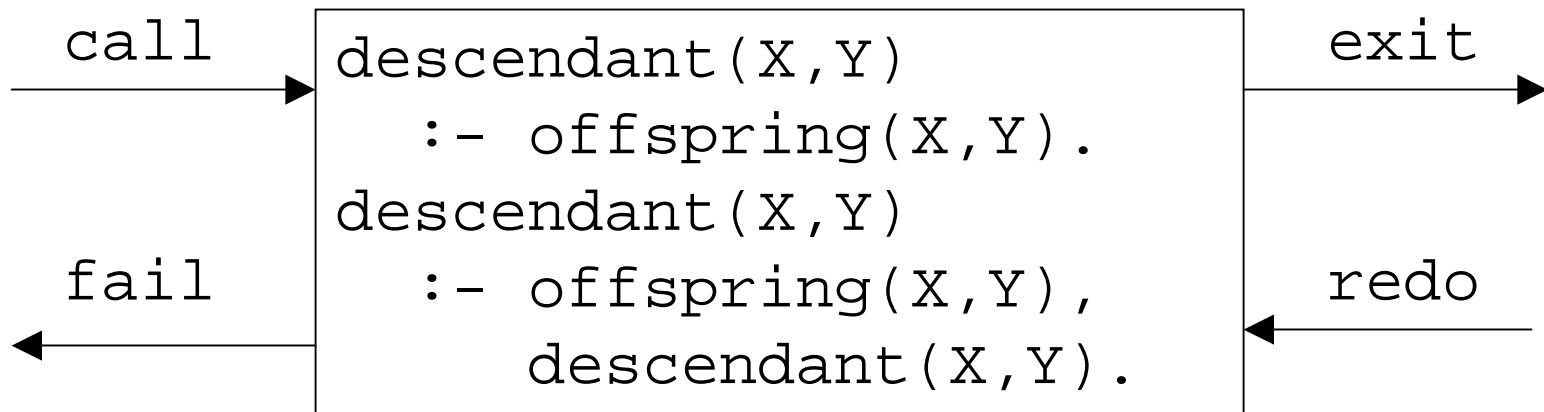
- How do people learn skills in practice
- Declarative and procedural encoding (Anderson, 1982)
 - How do people acquire a declarative encoding in the first place?
- Skills don't begin with passively receiving instruction
- Skills begin with imitation
 - Other objects (e.g. experts)
 - Other objects (e.g. multimedia systems)
 - Models of skill acquisition need to include aspects of syntonicity

Case study 1: Prolog

- Big in the psychology of programming community
- Several stories about of execution, e.g.
 - Byrd box
 - Choice point
 - Which is more syntonic?
- Close (perhaps too close) to common-sense psychology
 - ‘Simple desire’ psychology (Wellman, 1990)
- Control flow versus data flow
- Why is Basic relatively easy?

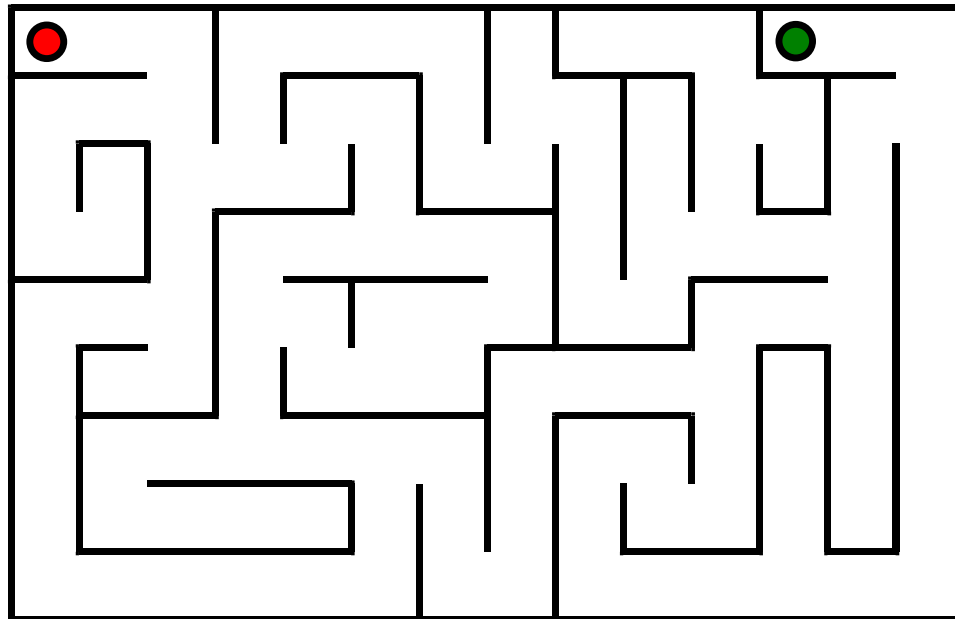
The Byrd box model

- “The ‘box’ idea delineates the procedure as the prime focus of attention” (Byrd, 1980)



The choice point model

- The myth of Theseus (Mulholland, 1995)
- Prolog itself becomes the prime focus of attention



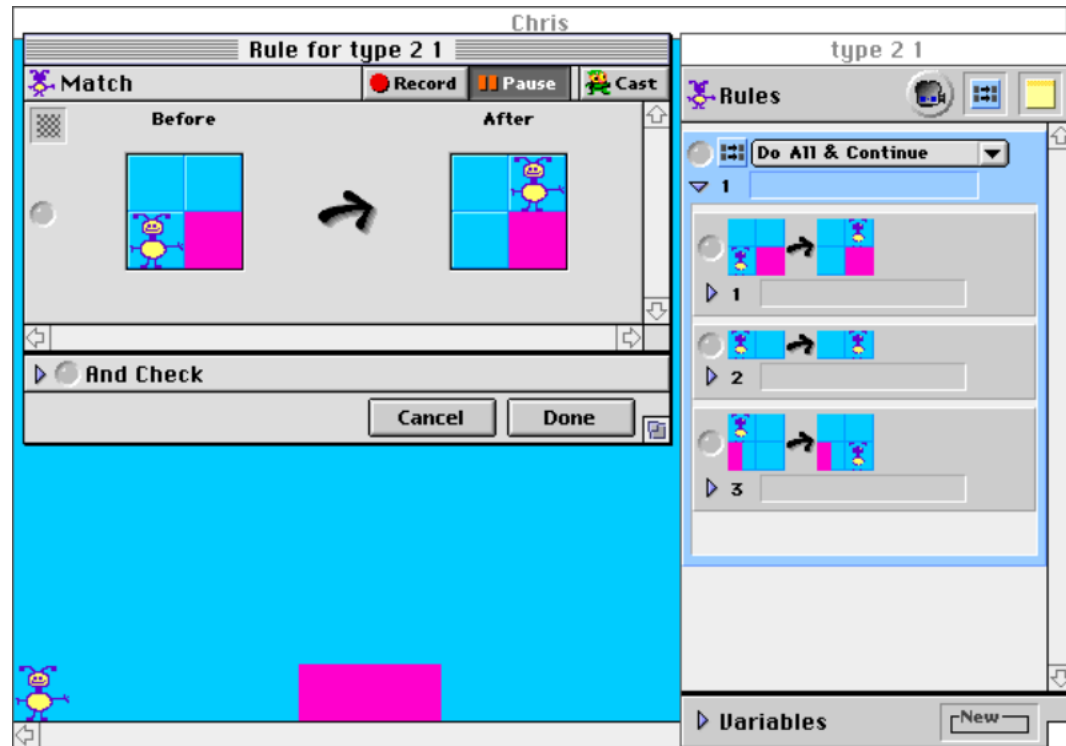
Case study 2: Logo, StarLogo, Playground, KidSim, and Cocoa

- Simulation environments
- Intended for children
- Two kinds of scripting
 - Textual languages (Logo, StarLogo, Playground)
 - Graphical rewrite rules (KidSim, Cocoa)
 - Procedural and declarative languages
- Syntonicity
 - Identify with the turtles

Comparing StarLogo and Cocoa

```
to find-food-demon
if (not carrying-food?)
and ask patch-here [food > 0]
[set-carrying-food? true
ask-patch-here [set-food food - 1]
set-drop-size 35
right 180 forward 1]
end
```

The ant 'wants' food



The monster 'wants' to climb over obstacles

Issues raised by simulation environments

- Logo, StarLogo, Playground
 - Textual rules, written like instruction sheets
 - Agent's eye view
 - Both ego and body syntonic
- KidSim, Cocoa
 - Graphical rewrite rules, written like situations
 - God's eye view
 - Harder to identify with agents?
- Which really is better, and what for?

'So What?'

- Does syntonicity really matter?
 - Logo?
 - Is syntonicity endemic to all languages and paradigms?
 - Who is going to use your language, and what for?
- Syntonicity as a theory of 'graspability'
 - Beyond computation
 - Beyond formalisation
 - Back to common-sense
 - Is this right for novices?

Implications, thoughts, future work

- More detailed evaluation is needed
- What is the difference between body syntonicity and ego syntonicity?
- Syntonicity may be important for multimedia teaching of skills
 - Agents one can identify with
 - Agents one can imitate
 - Can we build experts into environments?

Summary

- People treat programs as psychological entities not physical ones
 - Common-sense psychology is important
 - Agents and objects
 - Don't expect people to manipulate agents
- Imitation is important to learning skills
- Syntonicity is important to imitation
- Syntonicity isn't the only factor, but it is an important one, or is it?