Some thoughts and demos, on ways of using computing for deep education on many topics

As a change from teaching:

- "useful" skills (of various kinds),
- uses of computing,
- computer science
- computer/software engineering.

How to really stretch young minds as evolution designed them to be stretched — by the environment, by teachers and by themselves.

(Illustrated using teaching materials in Poplog/Pop-11.)

Aaron Sloman, School of Computer Science, UoB.
http://www.cs.bham.ac.uk/~axs

An updated version of these slides will (eventually) be made available in my talks directory
http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#ncsl

Towards a liberal education for the 21st Century

Some things should form part of a general educational system because they are worth knowing about in their own right for many different reasons, including all the main academic disciplines:

- physical sciences (physics, chemistry, astronomy, earth sciences, ...)
- biological sciences, including neuroscience, psychology,
- mathematics
- linguistics
- philosophy
- music and other art forms
- history, geography, literature, social sciences
- application fields: engineering, medicine, ...

A subset of worthwhile competences are specially important because they provide the tools for learning, thinking, reasoning and communicating about some, or all, of the others – it is normally assumed that these include the "three-Rs": Reading, wRiting, aRithmetic.

(Actually arithmetic is a small subset of mathematics, the real third competence.)

I suggest we need Five Rs, i.e. including programming broadly construed as learning to think about and use tools for thinking about:

structures, processes and how structures can produce and be produced by processes.

Computers have been able to support such education for nearly 40 years – but alas the opportunity has been mostly ignored, and it is now very difficult to use it: but perhaps not impossible. **That's the background to this presentation.**

Two inspiring little books

The Child's Discovery of Space: From hopscotch to mazes – an introduction to intuitive topology

Smaller J. Sauvy and S. Suavy,
Penguin Education,
Harmondsworth, 1974,
Translated from the French by Pam Wells,

How to Solve It

George Pólya

Princeton University Press

1945

`http://en.wikipedia.org/wiki/How_to_Solve_It`

Both of those books inspired me many years ago.

Also

John Holt *How Children Learn*

Ivan Illich *Deschooling Society*

(Being inspired by something does not necessarily involve agreeing with everything in it.)

The Three Main Points

1. Cognitive learning and development involve growing and populating an **information-processing architecture.**

Driven to a large extent by playful exploration – including things going wrong

Bugs/errors/mistakes are not bad things to be avoided: they are a deep resource for learners.

(Compare ideas by John Holt, Ivan Illich, Alan Kay, Seymour Papert, Gerald Sussman, Mitchell Resnick, Marvin Minsky, Jean Piaget, and others. Also philosophers of science: K. Popper revised by I. Lakatos)

2. Computers can provide powerful provocations and support for the process e.g. playing with AI programs and AI programming concepts.

Scratch, Alice, Greenfoot, Roblox and other new graphics-based tools are powerful engagers for young learners – but can be contrasted with simpler, smaller, and more flexible and extendable learning environments, some more focused on AI and some less graphics-based.

We need especially to understand the importance of textual manipulation to support abstraction and recombination of ideas. A Case study: Poplog/Pop-11 teaching materials (free & open source).

3. One possible way to get more of this into schools.

Make remotely-usable systems (Poplog isn't the only one) available remotely, using Xming and SSH clients on Windows PCs to connect to well-managed, shared linux machines, supporting strong collaborative learning and teaching, especially introducing simple forms of AI programming.

NOTE: Computers make powerful new forms of learning possible, but other things must not be forgotten ...

A major part of my education was playing with meccano out of school hours between the ages of about 5 and 10, over 60 years ago.

Not everything that is important in human learning can be done with computers:

The 3-D world can be a powerful teacher: it drove much of our evolution.

Education – aims, and tools

Some ideas about what education is for and how it can work: especially using opportunities provided by computing.

Successive well-meaning governments have got it badly wrong: especially in recent years in the UK!
(Both content and management – e.g. not allowing enough diversity in developmental trajectories.)

- School education, especially in early years, should primarily be about learning how the world (including ourselves) works, plus learning to think, learn, and communicate.

  Skills that serve those ends are deeply important – much more important (for children) than learning to use the latest gadgets – especially for children with high academic potential.

- Learning to use tools currently used in industry is no preparation for the real future.

  E.g. We should not teach physics by teaching children to drive tanks or fly airliners.

  Good ways to think have much longer-lasting value.

- Not much is known about the mechanisms and forms of representation involved in the processes through which humans learn and develop –

  but there are some hunches that look promising (educationally, if not politically – see below).

  We need better theories about how processes involving deep and effective learning can occur:
  what can happen in a learner's mind? E.g. How do ontologies develop?

- It is often assumed that the motivation that drives learning must always involve rewards – e.g.through teaching that provides fun and enjoyment.
- But evolution has produced, in humans (and some other animals), powerful learning mechanisms that do not need rewards: **what they really need is opportunities!**
- An idea from Vygotsky provides a useful concept for thinking about all this. (ZPD)
  (I use my own version of his idea: he can't be blamed for what follows.)

A Learner's Zone of Proximal Development–ZPD

Educators need to understand and make good use of the learner's ZPD.
`http://en.wikipedia.org/wiki/Zone_of_proximal_development` (Originally Vygotsky's idea.)

Learning should include **meeting challenges:** if all challenges are too close to what the learner can do easily, there will be little learning (though perhaps much enjoyment).

If the challenge is too far beyond what the learner can do at all, there may be no learning.

In between those extremes is the learner's ZPD, with "near" and "far" boundaries:

- actions close to the "near" boundary accumulate information and make skills faster/more fluent. (How?)
- actions beyond the "far" boundary produce failure and frustration.
  (though sometimes useful seeds are planted for delayed learning effects).
- In between there can be piecemeal accumulation of information (factual, procedural, and conceptual): increments within a largely unchanging framework, including training cognitive reactions.
  (Think of examples: new phrases learnt, new kinds of process observed, new action combinations...)
- Closer to the "far" boundary of the ZPD, learning can be (temporarily) more disruptive, producing
  - awareness of not understanding,
  - a need for substantially new concepts,
  - pressure for a new type or layer of competence,
  - pressure for a new explanatory theory,
  - pressure for a new form of representation, with new modes of reasoning, predicting, explaining,
  - pressure for a new organisation for old material (e.g. with new abstractions) ...
  - new productive confusion, that eventually drives conceptual extension or reorganisation.

How does this happen? We don't yet have detailed working models – only hunches.

INTERLUDE: reasoning quirks during development:
reasoning about potentially colliding cars

Here is an example of a child who is nearly, but not quite, ready to reorganise and systematise some of the things he has learnt.

A child can learn to use a mixture of simulation and symbolic/verbal reasoning to draw conclusions and to justify conclusions.

That learning takes time and the process can go through buggy stages as illustrated by this example.

A racing car is on the left, and a van on the right.

The two vehicles start moving towards each other at the same time.

The racing car on the left moves much faster than the truck on the right.

Whereabouts will they meet – more to the left or to the right, or in the middle?

Where do you think a five year old will say they meet?

One five year old's spatial reasoning

The two vehicles start moving towards each other at the same time.

The racing car on the left moves much faster than the truck on the right.

Whereabouts will they meet – more to the left or to the right, or in the middle?

One five year old was asked this question with two toy vehicles placed on opposite ends of a window-sill, facing each other.

He answered by pointing to a location near 'b'

One five year old's spatial reasoning

The two vehicles start moving towards each other at the same time.

The racing car on the left moves much faster than the truck on the right.

Whereabouts will they meet – more to the left or to the right, or in the middle?

One five year old answered by pointing to a location near 'b'

Me: Why?

Child: It's going faster so it will get there sooner.

What produces this answer:

- Missing knowledge?
- Inappropriate representations?
- Missing information-processing procedures?
- An inadequate information-processing architecture?
- Inappropriate control mechanisms in the architecture?
- A buggy neural mechanism for simulating objects moving at different speeds?

Partly integrated competences in a five year old

Perhaps the child had not yet learnt that there can be constraints on ways in which items of information can be combined, like these:

- If two objects in a race start moving at the same time to the same target, the faster one will get there first
- Arriving earlier implies travelling for a shorter time.
- The shorter the time of travel, the shorter the distance traversed.
- So the racing car will travel a shorter distance!

The first premiss is a buggy generalisation: it does not allow for different kinds of 'race'.

The others have conditions of applicability that need to be checked.

Perhaps the child had not taken in the fact that the problem required the racing car and the truck to be travelling for the same length of time, or had not remembered to make use of that information.

Perhaps the child had the information (as could be tested by probing), but lacked the information-processing architecture required to make full and consistent use of it, and to control the derivation of consequences properly.

This is a simple example of a type of confusion I have often seen in students trying to think about processes: many features of a process are understood, but the learner has not developed a way of thinking about them simultaneously so as to ensure consistency. (Inadequate short-term memory mechanisms?)

(Compare being unable to see the impossibility of a Penrose triangle – or more complex Escher picture.)

At any age the ZPD can differ for different learners

Apart from genetic differences, children differ in many ways:

the kinds of toys, familiar actions, well understood household objects, social patterns, vocabulary and syntax, historical and political references, degrees and kinds of domestic harmony and disharmony...

all of which influence and limit the forms of learning available to the child at any time.

- New forms of learning often have to be based on familiar examples, and prior competences, so "the zone of proximal development" at each age will differ for different children for environmental as well as genetic reasons.
- Possible cognitive transitions at any age will differ between learners:

  the full set of possible routes to any major development **will not form a linear sequence** that all can follow, but a partially ordered network, through which different learners must take different routes.
- **The processes of teaching and assessment must allow for individual variation**

  including differences in both trajectories and speeds of traversal.
- This makes teaching very demanding – especially in large mixed-ability classes.
- The need to stretch all while discouraging none can make production of teaching materials tailored to the students very difficult.
- But humans and other animals can learn in environments that adjust themselves to the playing, learning, individual – e.g. like a part-built sandcastle, or meccano model.
- Computers can also help support personalised trajectories (e.g. using mini projects)
- However, only trivial things can be taught without generating (temporary) confusion.

An important way in which computing can help

A partial solution is to provide opportunities for learners to select and follow their own routes – in a well designed space of possibilities for learning.

How can a learner become a good driver? By using suitable roads and vehicles!

- Divide a course into packages, where each package starts as simply as possible and then leads on to increasingly difficult challenges,
- Each package should include
  - some tasks everyone in the group can cope with
  - practice with varied sub-types of whatever is being learnt
  - some tasks all average students can cope with
  - clear specifications of goals all students should aim at, and indications of possible targets for high fliers (HFs).
  - some increasingly hard problems and challenges that can drive further deep learning in the HFs: learning to explore, hypothesise, test, revise, debug, reconsider, analyse, explain, document ...
- Make sure the HFs have opportunities to go on learning to their limit.

  Time will inevitably be a constraint: they also have to learn when to move to another package.
- Tools should give teachers great flexibility:

  Packages may need to be re-ordered, or broken up into smaller packages for younger learners, or disadvantaged learners.

  Teachers need to be able easily to change the examples to suit their context.
- Some illustrations of these teaching ideas are presented here (needing improvement!)
  `http://www.cs.bham.ac.uk/research/projects/poplog/freepoplog.html#teaching`

Example: A challenge for older learners

A pop-11-based file on using lists of ‘x’s to represent numbers,

e.g. the first few numbers are

[]

[x]

[x x]

[x x x]

[x x x x]

starting with easy(?) exercises for doing addition, subtraction and multiplication, using list-processing algorithms,

then progressing to deep and difficult challenges, e.g. about negative numbers:

`http://www.cs.bham.ac.uk/research/projects/poplog/teach/teachnums`

(An introduction to “unary arithmetic”.)

(Can **you** explain why multiplying two negative numbers should give a positive number?)

Learning, enjoyment, emotions...

"Affective support" may need to be part of the solution: E.g.

helping learners understand why there are different rates of development, helping them

- to cope with the fact
  (including learning that some others are struggling just as much as they are)
- both to provide and to benefit from mutual help.

Some may have to unlearn "learnt helplessness" and "learnt self-condemnation" for poor performance.

Enjoyment does not have to be the main driver: a talented learner will be driven by **opportunities** to learn.

Don't confuse finding tasks absorbing and finding them enjoyable.
Being absorbed can drive deep learning.

Of course, a good learner will enjoy learning some of the time,
– but different learners will enjoy different things.

A good learning environment should cater for that.

For a sketchy theory contrasting reward based motivation with architecture-based motivation, see:

`http://www.cs.bham.ac.uk/research/projects/cogaff/misc/architecture-based-motivation.html`

Examples presented at the conference

My conference presentation demonstrated a number of features of Poplog: a powerful, free, open source toolkit that shows how teachers and students can learn together –

building, exploring, playing, discovering, learning, and teaching,

especially using its extremely versatile core language Pop-11.

(It's not the only usable system: just an unusually flexible example.)

In a PDF presentation I cannot repeat the demos.

Examples of things that can be done with young learners of various ages are here:

`http://www.cs.bham.ac.uk/research/projects/poplog/freepoplog.html#teaching`

Those are just examples of what can be done with the teaching resources in Pop-11: creative teachers can modify, extend, or replace those examples.

(Much, though not all of it, could be done using other languages.)

More on this below.

What changes during childhood and after?

Humans go on learning from infancy to old age, but that is not a steady accretion of knowledge or skills: changes at different stages are very different, and new learning can build on old via radical changes. E.g.:

- Acquiring a new piece of information using pre-existing concepts and formalisms.
- Acquiring a new skill by combining old skills sequentially, or concurrently.
- Making existing skills faster, more fluent (including cognitive skills)
- Discovering that there is something you don't know/haven't understood.
  (Learning a new question!)
- Finding a new generalisation to unify previously learnt special cases.
  (requires appropriate concepts and formalisms)
- Finding a counter-example to an accepted generalisation.
- Acquiring/constructing a new concept (possibly an indefinable concept)
- Altering/clarifying an old concept
  (e.g. from "getting faster" to "accelerating at X cm/sec/sec")
- Learning the layout of some confusing building, city, terrain, set of concepts.
- Learning a new form of reasoning, explaining, e.g. logical, geometric, algorithmic ...
- Extending the information-processing architecture
  (e.g. adding a new kind of self-monitoring)

and more ....

Only trivial things can be taught without generating (temporary) confusion.

Living things are informed control systems

Multiple designs are found in living things.

Not only physically varied designs – but also varied information-processing designs.

Living things are information processors: informed control systems

The world contains: matter, energy, information

Organisms are control systems:

They acquire and use information (external and internal),
in order to control how they use matter and energy

(in order to acquire more matter, energy and information,

and also reproduce, repair, defend against intruders, dispose of waste products...).

Evolution produced more and more sophisticated information processors,
driven partly by changes in environments, partly by results of prior evolutionary history.

All organisms are information-processors but the information to be processed has changed and so have the means

Types of environment with different information-processing requirements

- Chemical soup
- Soup with detectable gradients
- Soup plus some stable structures (places with good stuff, bad stuff, obstacles, supports, shelters)
- Things that have to be manipulated to be eaten (e.g. disassembled)
- Controllable manipulators
- Things that try to eat you
- Food that tries to escape
- Mates with preferences
- Competitors for food and mates
- Collaborators that need, or can supply, information.
- and so on .....

Tradeoffs-between evolution and development

Biologists distinguish

Precocial species: individuals born physiologically advanced and competent
e.g. chicks follow hen, and peck for food; new born deer run with the herd

Altricial species: individuals born under-developed and incompetent – very dependent on parents.

Jackie Chappell convinced me in 2005 that:

It's not species but competences that should be distinguished.

Even humans have some "precocial" competences (e.g. sucking is highly developed very early).

Evolution produced very dramatic changes – over millions of years.

But it also produced some species whose individual members can make dramatic changes in their own information-processing, in only a few years –

huge changes, not in their physical shape, etc., but in their information-processing powers, and their knowledge about the world they live in.

Learning to talk, learning to read music and play musical instruments, learning to build skyscrapers, learning to make advances in theoretical physics. How?

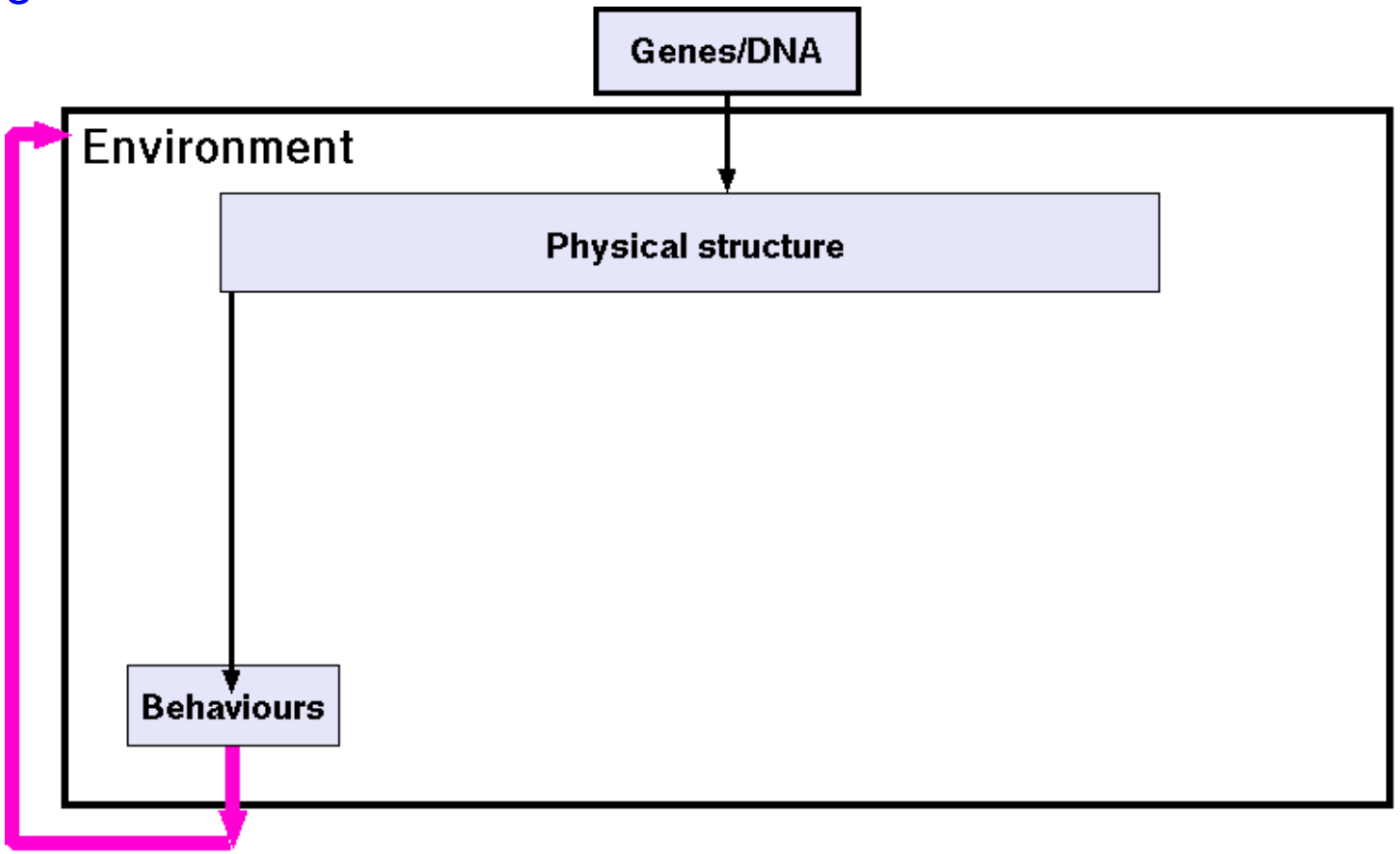
The world is now very different from the world in which that versatility evolved: but that's why deep and rapid learning capabilities were needed.

Educators need to understand those biological mechanisms – yet no known theory is adequate: e.g. all currently implemented models of learning are simplistic by comparison.

But we can say something about the variety of routes from genome to behaviours.

Individual developmental trajectories

Routes from genome to behaviour : the direct model.

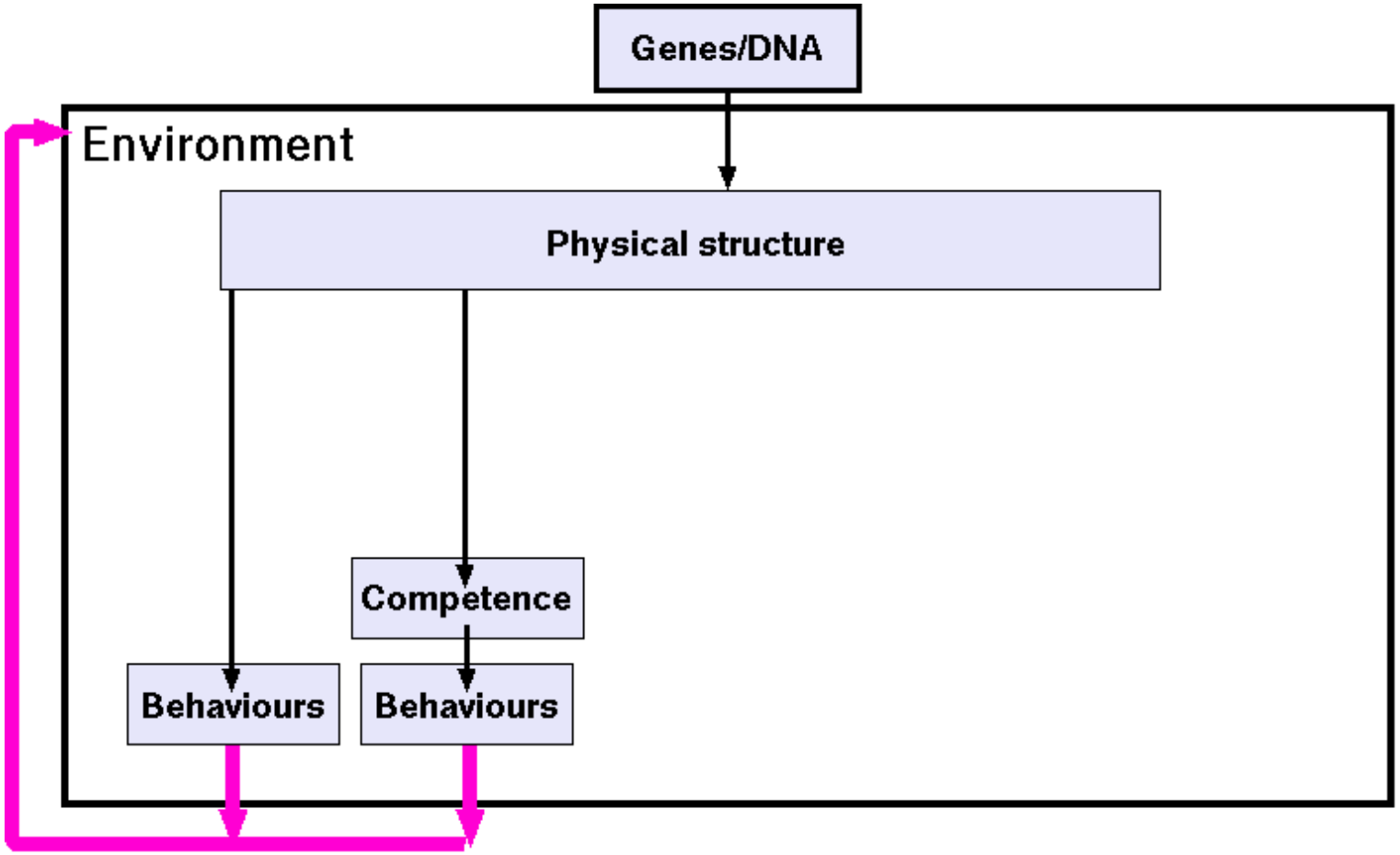


The vast majority of organisms (including micro-organisms) are like this.

Many don't live long enough to learn much – they have to make do with innate reflexes. Other organisms have more "inside the box".

Individual developmental trajectories

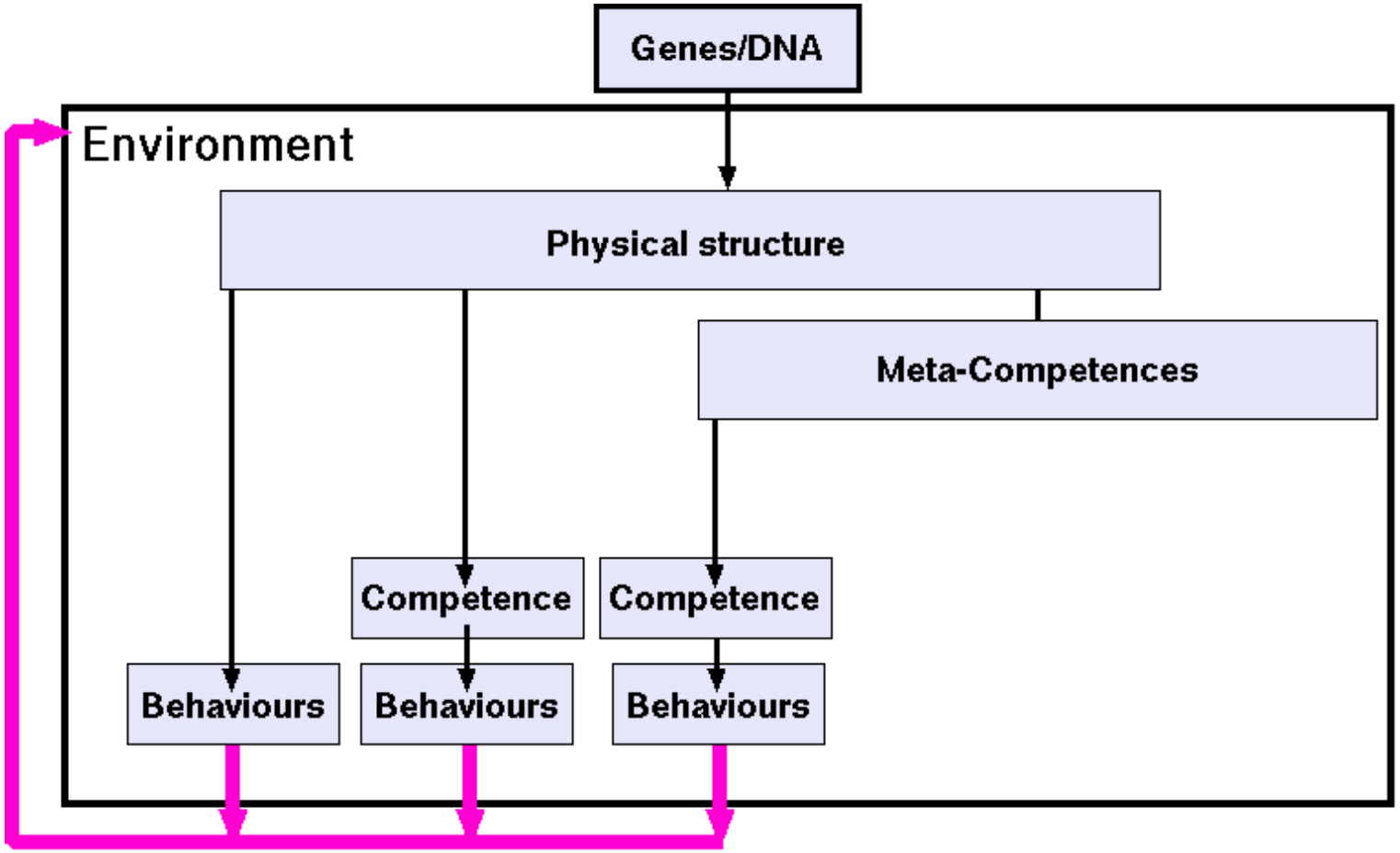
Routes from genome to behaviour : the two-stage model.



Some more complex organisms, instead of having only rigid (reflex) behaviours, also have competences that allow them to respond in fairly flexible ways to the environment: adapting behaviours to contexts.

Individual developmental trajectories

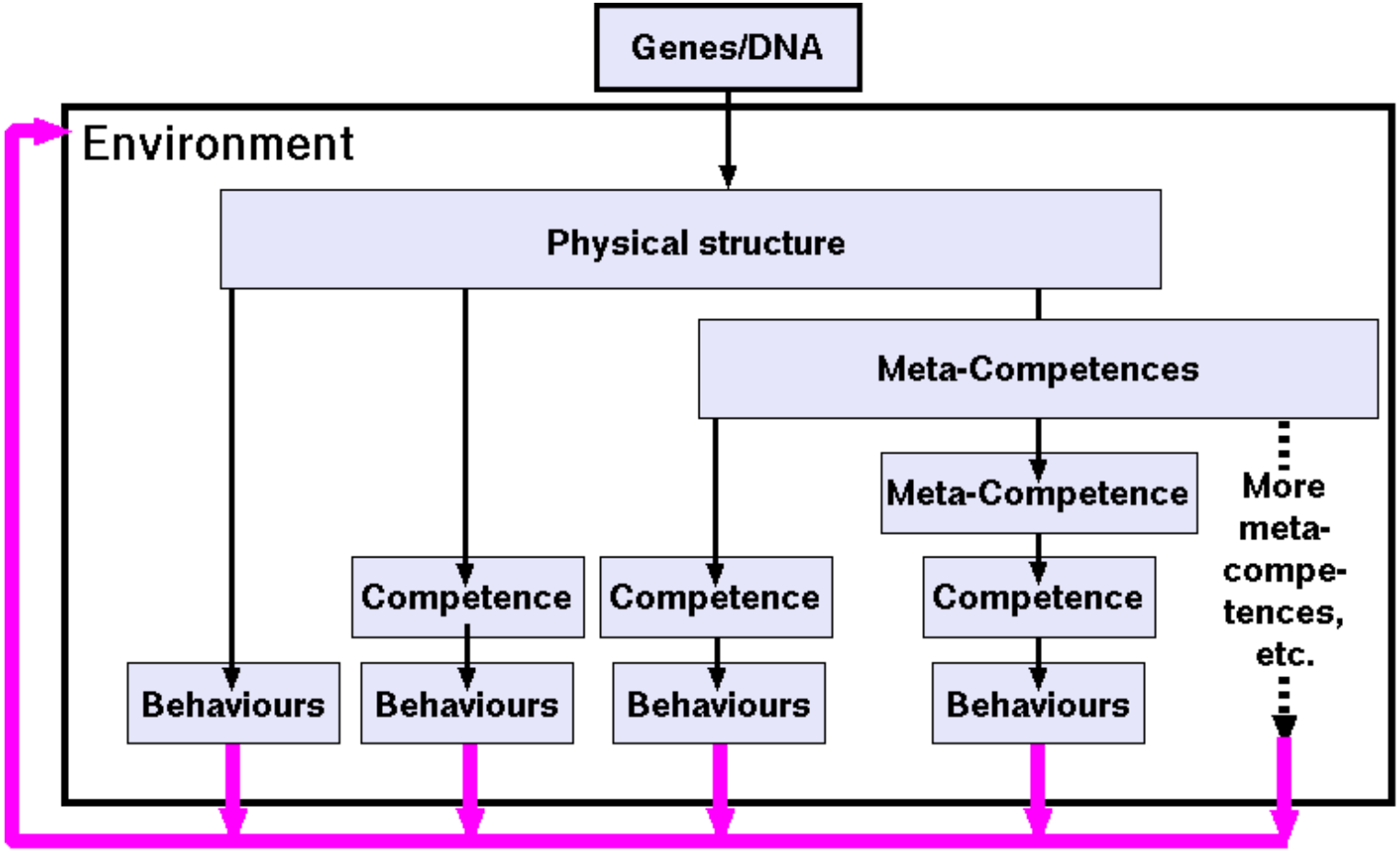
Routes from genome to behaviour : stages added by learning.



Genetically determined meta-competences allow individuals to respond to the environment by producing new types of competence, increasing flexibility and generality.

Individual developmental trajectories

Routes from genome to behaviour : the multi-stage model.



Some can also develop new meta-competences, on the basis of meta-meta competences.

Humans seem to be able to go on developing meta-meta-competences until late in life.

Implications of the Multi-stage Model for education

- **Children build their own information processing architectures**
  (Every now and again adding major new layers or subsystems.)
- Teachers merely help (but sometimes hinder) the process
  by providing building materials – along with challenges and opportunities.
  (Ideas from Jean Piaget, Ivan Illich, John Holt, Seymour Papert, Mitchel Resnick and others...)
- Learning by exploring and playing is a crucial aspect of human development.
  (And development in some other species.)
- Previous individual history limits what's learnable (e.g. when is a child ready for a new layer?)
- There cannot be a single learning trajectory followed by all children.
- We must find ways to let children (possibly in collaboration) build their own trajectories.
- **Vygotsky: Zone of proximal development:** what a child can learn at any stage is limited – but deep learning happens near the limits.
- Bruner and others: "Scaffolding" may be needed to support (and limit) the process.
- Only trivial things can be taught without generating confusion.
  Like building a map of terrain as you explore it – you will **inevitably** get things wrong at first and have to correct them later – as happens in the development of science and engineering.
  We need to understand that that's a feature of all deep education (e.g. learning a first language).
- Computers provide new opportunities for children and teachers to learn together.
  (Use of networks can extend the collaboration.)

Having fun versus being stretched

We must be careful not to assume that what children (or even teachers) **like** best **is** best.

- Some of what both children and teachers like best may be too easy – it doesn't push learners towards the limits of their "ZPD" (zone of proximal development).

  I recently heard a BBC radio 4 program "Word of mouth" about powerpoint presentations, which reported research showing that multi-media presentations that students said they liked best actually did not produce as much learning as duller, more old fashioned, presentations.

  `http://www.bbc.co.uk/programmes/b006qtnz`

  (after about 14 minutes 28secs for about 4 minutes: Prof. Rich Mayer `http://www.psych.ucsb.edu/people/faculty/mayer`)

- Alongside all the wonderful new graphical tools that make it easy for young learners to assemble impressive demonstrations and games, and learn to work together, we also need some other kinds of "more traditional" challenge –

  especially for the children with the highest long term potential.

  (Yes I am elitist about never holding learners back for the sake of the majority!
  Compare athletes, musicians, poets, ...
  There's good and bad elitism, as explained here:
  `http://www.cs.bham.ac.uk/research/projects/cogaff/misc/elitism.html`)

Merits of text-based programming exercises

Some things that can be done with text-based programming cannot easily be done using graphical formalisms and "direct manipulation".

See Andrew Downey's preface to *How to think like a computer scientist*:
"The problem is that many of the more exciting features involve lots of details and not much concept. Pedagogically, that means a lot of effort for not much payoff. So there is a tradeoff between the material that students enjoy and the material that is most intellectually rich. I leave it to individual teachers to find the balance that is best for their classes." `http://www.greenteapress.com/thinkapjava/preface.html`

I am not arguing against using graphical programming interfaces for teaching:
I have even developed some myself, e.g.
`http://www.cs.bham.ac.uk/research/projects/poplog/figs/rclib/rc_graphic.html#panel`

However, doing **everything** that way can deprive students of

- opportunities to learn about kinds of abstraction that are best expressed textually (e.g. the abstractions that are important in logic, in grammars, in inheritance systems, in pattern-driven rule-based systems, ...),
- benefits of being able to study static records of textual output produced by a series of commands,
  looking for patterns in relationships between commands and effects, which are not available if all the program-outputs are transient changes on dynamic displays.

Several examples of the use of AI programming languages and techniques in stretching learners' minds are available on the poplog site:

`http://www.cs.bham.ac.uk/research/projects/poplog/freepoplog.html#teaching`

Many were developed before graphical terminals for teaching were affordable – and still retain their potential educational value, e.g. in teaching concepts related to the structure of language, planning, reasoning, using analogies, etc.

Beyond cumulative learning

The addition of new meta-meta-competences, new forms of representation, new explanatory theories, etc. can be disruptive processes, including discarding old temporary skills, ideas, and theories:

Deep learning is not continuous and cumulative

- One of the important features of human learning is that some knowledge originally acquired empirically (by observation, exploration, being told, etc.) is later reorganised so that there are new core principles from which other things can be derived.
- Thus old knowledge that was initially empirical is reconstrued as non-empirical, like theorems in topology, geometry, arithmetic, logic, ...
- The new system allows things to be predicted and planned without prior experience of the particular cases – crucial for dealing with novel problems.
- The learner need not be aware of this happening!
- An example from language is the development of syntax:

  e.g. discovering that there are very useful rules for forming plurals, past tense, etc., so that new cases can be dealt with, without having to be learnt.
- I believe this kind of transformation is the basis of human mathematical abilities: but the architecture has to grow before it can do this.
- This is crucial to learning to program also: it is empirical at first, then later understood!

Even toddlers transform empirical discoveries

Generalisations are learnt,

then counter-examples discovered,

causing the generalisations to be abandoned or de-bugged,

and in the process ontologies are extended,

e.g. length of a curved string, knots, loops, forces being transmitted indirectly, new "kinds of stuff", new properties of kinds of stuff, rigidity, elasticity, etc.

Some of the learnt generalisations then acquire the status of **toddler theorems**.

This does not mean

(a) that they are explicitly labelled as such, only that they may be treated as if incapable of refutation – e.g. being used with total confidence on novel situations, rejecting alternatives.

(b) that they are derived from some **prior** explicitly formulated theory: the theory may have to be grown.

(c) that the derivations use logical reasoning, or any familiar mathematical formalism.

Mathematical competences and biological evolution

Most people think that learning mathematics requires being taught by someone who has already learnt mathematics

A QUESTION:

Who taught the first mathematicians?

ANSWER

Biological evolution and the environment we evolved in, working together.

That's the process educators are contributing to.

The bootstrapping is still continuing – and we have opportunities to push it forward.

But the opportunities are not being taken.

We need a better understanding of the problems evolution solved – in order to understand the solutions

It's hard: many problems are invisible.

Some incomplete ideas about these processes can be found in:

`http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#toddler`
Why (and how) did biological evolution produce mathematicians?

`http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#babystuff`
Ontologies for baby animals and robots

Some infant and toddler examples

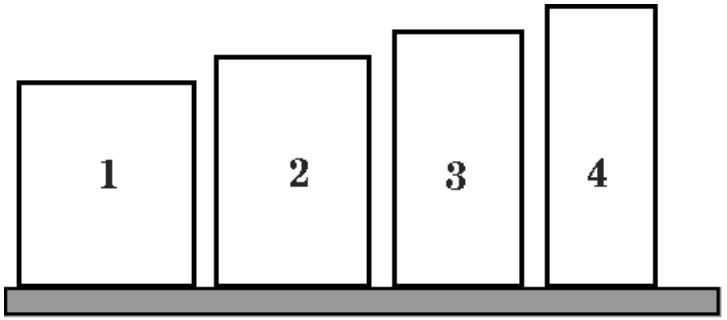
It takes a child some time to learn how to align a cup so that it will go into or stack on another cup.

Toddler theorem: If cup A is to be inserted into cup B, then the orientations of the cups must be aligned, A must be on the open side of B, and the axis of cup A must point into cup B, but rotation of A about that axis makes no difference.

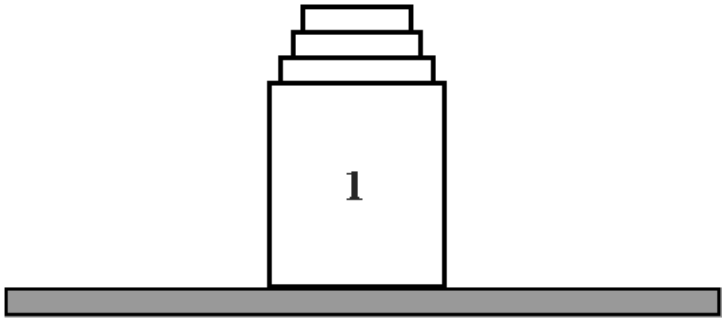
A Process theorem: Toddler Nesting

Understanding how to stack cylinders.

Consider being faced with a collection of cylindrical cans or mugs of different heights and widths all open at the top, as shown in the upper picture.



What do you have to do to get them all tucked together compactly as in the second picture?

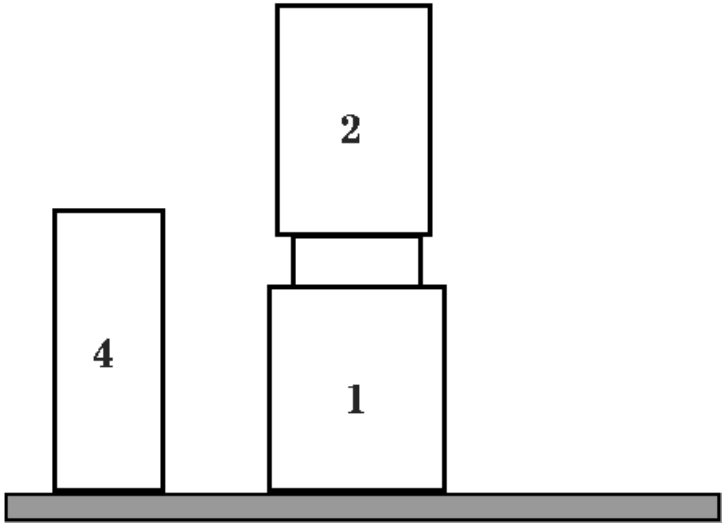


What steps should you go through?

A very young child may try putting one mug into another at random, and then get frustrated because situations result like the one depicted at the bottom, where the top mug cannot be pushed down into the mug below it, as required.

By trial and error a child may learn a sequence of actions that works, e.g.

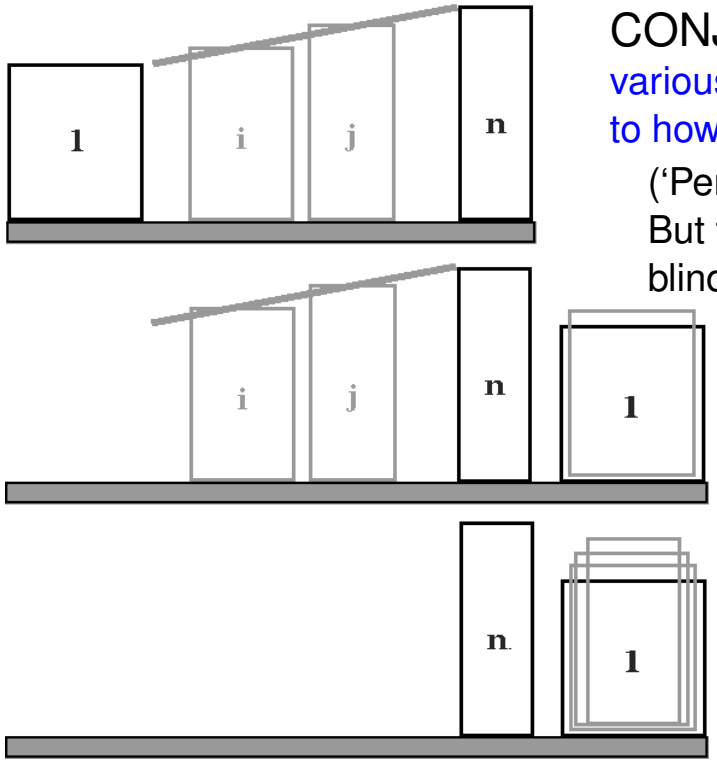
(i) put mug 2 into mug 1
(ii) put mug 4 into mug 3
(iii) lift mug 3 with mug 4 inside it and put both into mug 2



What if there are five mugs, or nine mugs?

A more "intuitive" mode of reasoning?

Children (and most adults) apparently cannot construct logical proofs, but I conjecture that they can grasp the necessity of the outcome by using a different form of representation based on "analogical representations" (Sloman 1971, Peirce's Icons?):



CONJECTURE:

various stages of the process are represented in a manner that is closer to how they are perceived visually than how they are described logically.

('Perceived visually' has to be qualified for people blind from birth. But they still benefit from having had ancestors with vision if the blindness is caused by a peripheral defect not something central.)

The reasoning is easiest in the special case where the containers are first all arranged in a row, decreasing in width.

That configuration can be thought of schematically as having a widest cup on the left and a thinnest one on the right, with every adjacent pair (i,j) having the wider (i) on the left, but without representing specifically how many cups there are, nor how big any of them is, nor exactly where they are:
This requires a "generic" or "schematic" analogical representation – using "visual ellipsis".

(Cf Kant vs. Berkeley, on generic triangle images.)

From that starting configuration (top image) you can envisage a first step of moving the widest cup to a new location (middle image) and then repeatedly taking the last cup on the left and putting it in the growing stack.

So there will be a generic situation that can be visualised, with the row of remaining cups, decreasing in width from left to right, and the already stacked cups somewhere separate – until one cup is left, and finally put inside the stack. (Alas we can't yet get computers to reason like this.)

More on the learning process

- The stages of development at first implicitly presume, then later explicitly articulate, theories about processes and structures, and later still organise them into a system. (Not all individuals do all of this.)
- Some aspects of those theories will be empirical and potentially refutable (such as theories about what rigid, solid material objects do not do if left alone, e.g. split or merge) and others may be intrinsic to any process of learning about what can and cannot happen in a 3-D world where things can move.
- I call the latter “framework theories”: A long term research goal is teasing them out, and separating them from the empirical theories.
- NB: the processes are not guaranteed to be bug-free: errors can occur, but can be detected and fixed

There is still much work to be done

I have tried to sketch, at least in outline, a theory of some of the ways evolution dealt with the need for learners to be able to find out about situations that never arose in their evolutionary history.

Those abilities to think and reason about what has not yet been experienced is crucial to programming skill, as well as designing new physical machines, and building new explanatory theories in the sciences.

To some extent those abilities can be encouraged in the context of building amd testing working models that have some generality.

Teaching programming (using powerful tools and highly expressive languages) can develop the learner's ability to think about and understand relationships between structures and processes, including

- processes produced by structures
- processes limited by structures
- structures produced or modified by processes
- different structures and processes interacting at the same time

Related online papers and presentations

On learning about numbers: Problems and speculations.
http://www.cs.bham.ac.uk/research/projects/cogaff/crp/chap8.html
Chapter 8 of The Computer Revolution in Philosophy:
Philosophy, science and models of Mind

Teaching AI and Philosophy at School?
http://www.cs.bham.ac.uk/~axs/courses/ai-syllabus.pdf

Computational Cognitive Epigenetics (With J. Chappell in BBS 2007):
http://www.cs.bham.ac.uk/research/projects/cosy/papers/#tr0703
Commentary on Jablonka and Lamb (2005)

Evolution of minds and languages. What evolved first and develops first in children: Languages for communicating, or languages for thinking (Generalised Languages: GLs)
http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#glang

Diversity of Developmental Trajectories in Natural and Artificial Intelligence (AAAI Fall symposium 2007):
http://www.cs.bham.ac.uk/research/projects/cosy/papers/#tr0704

What Cognitive Scientists need to know about virtual machines
http://www.cs.bham.ac.uk/research/projects/cogaff/talks/#cogsci09

Additional papers and presentations:
http://www.cs.bham.ac.uk/research/projects/cogaff/talks/
http://www.cs.bham.ac.uk/research/projects/cosy/papers/
http://www.cs.bham.ac.uk/research/projects/cogaff/

More on Pop-11 and Poplog

Poplog: Developed at Sussex University and marketed by Systems Designers & ISL:

once an expensive commercial product e.g. used to develop Clementine, a world-beating data-mining system (bought by SPSS in 1998).

Poplog is now free and open source, and fairly small – also very robust.

For Poplog on linux see
`http://www.cs.bham.ac.uk/research/projects/poplog/latest-poplog`

For Poplog on Windows (no graphics) see
`http://www.cs.bham.ac.uk/research/projects/poplog/winpop`

Poplog runs on Windows, but without the graphical tools. A windows user with network access to a Linux machine with Poplog can use it remotely e.g. using Xming and a Secure Shell client on a Windows PC.

Instructions and pointers are here:

`http://www.mail-archive.com/pop-forum@cs.bham.ac.uk/msg00019.html`

We have offered to let UK school teachers experiment with remote use of our linux system and Poplog at Birmingham university.

Anyone interested should contact me.

At the "unconference" I had hoped to show some benefits of using Linux for presentations – e.g. using multiple virtual desktops, with rapid flipping between them, as needed. But there was not enough time in a 30 Minute session.

AI-related examples of Pop-11 'TEACH' files

From beginners, as long as they can read and type, to more advanced learners.

- TEACH RIVER
  About the problem of getting man, fox, chicken grain and boat across the river.
  `http://www.cs.bham.ac.uk/research/projects/poplog/teach/river`
- TEACH ARITH
  about syntax for defining simple arithmetical procedures that can also process lists of words.
  `http://www.cs.bham.ac.uk/research/projects/poplog/teach/arith`
- TEACH respond
  A brief introduction to designing, implementing, and testing a simple chatbot (Eliza), using the pop-11 pattern matcher, with a simple but powerful control architecture.
  `http://www.cs.bham.ac.uk/research/projects/poplog/teach/respond`
  A working example online: `http://www.cs.bham.ac.uk/research/projects/cogaff/eliza`
- TEACH grammar
  How to use a formal context free grammar to overcome restrictions of pattern-matching – and how trying to find a good grammar to do what you want can teach you things about your own language.
  `http://www.cs.bham.ac.uk/research/projects/poplog/teach/grammar`
- TEACH rc_graphic
  Intro to Pop-11's "relative coordinates" turtle graphic system, the basis of a host of user extendable 2-D graphical tools
  `http://www.cs.bham.ac.uk/research/projects/poplog/figs/rclib/rc_graphic.html`
  `http://www.cs.bham.ac.uk/research/projects/poplog/figs/rclib/`
- TEACH rulebase
  Introduction to rule-based programming (using the Poprulebase extension to Pop-11)
  `http://www.cs.bham.ac.uk/research/projects/poplog/packages/current/prb/teach/rulebase`

More general programming in Pop-11

A sample of introductory materials related to various types/aspects of programming.

- TEACH loops

  `http://www.cs.bham.ac.uk/research/projects/poplog/doc/pophelp/loops`

- TEACH lists

  `http://www.cs.bham.ac.uk/research/projects/poplog/doc/popteach/lists`

- TEACH sets (based on lists)

  `http://www.cs.bham.ac.uk/research/projects/poplog/doc/popteach/sets`

- TEACH recursion

  `http://www.cs.bham.ac.uk/research/projects/poplog/doc/bhamteach/recursion`

- TEACH the functional style

  `http://www.cs.bham.ac.uk/research/projects/poplog/doc/bhamteach/functional.style`

- TEACH stack

  Use of the open stack (as in FORTH and many calculators)

  `http://www.cs.bham.ac.uk/research/projects/poplog/doc/popteach/stack`

- TEACH objectclass

  An introduction to object oriented programming with multiple inheritance

  `http://www.cs.bham.ac.uk/research/projects/poplog/doc/objectclassteach/objectclass_example`

- TEACH super

  Using the Super database package to learn logic programming

  `http://www.cs.bham.ac.uk/research/projects/poplog/teach/super_example`

- TEACH teachnums (implementing numbers using lists of ‘x’s).

  Learning to think unusually deeply about what numbers are and how they work.

  `http://www.cs.bham.ac.uk/research/projects/poplog/teach/teachnums`

The teach files are all plain text

Because the teach files are plain text files

(supported by program libraries to be invoked, modified, or extended by students)

it is simple for knowledgeable teachers

- to change the wording of a group of teach files
- to suit
  - a particular group of students
  - the teacher's preferred way of communicating,
  - a particular timetable constraint (e.g. lesson times)

In particular, it will often be desirable to change the examples in the teach files to refer to things a particular group of students knows about and can relate to.

Teaching materials should always be changeable or extendable by gifted teachers.

Pop-11 and the Poplog system