Reasoning about Complexity – Software Models as External Representations

Simon C. Lynch

School of Computing
Teesside University
United Kingdom
s.c.lynch@tees.ac.uk

Joseph Ferguson

School of Education
Deakin University
Melbourne, Australia
jpfergus@deakin.edu.au

Keywords: POP-I.B. internal and external representations, POP-III.B. NetLogo, POP-V.A. mental models.

Abstract

Programming is traditionally considered to be an activity which aims only to produce a software artefact as its primary goal. With this view programming languages are simply the notations which define these artefacts. This paper examines the relationships between internal representations (mental models) and external representations (notations and other forms) arguing that program code behaves as an external representation in a similar way to mathematical or logical notations but with the added property that code can be executed and its notational consequences observed.

Furthermore some environments allow program operation to be manipulated at run-time; we propose that these systems also operate as external representations and that programming language statements and their run-time environments can thereby be utilised as reasoning systems to promote the exploration and discovery of new understandings. In this context we consider NetLogo as a framework for reasoning about complex and emergent systems, evaluating its suitability from a representational perspective.

Introduction

This work was originally motivated by the desire to find a notation suitable for representing and reasoning about complex and emergent systems. We require this representational framework to be capable of describing executable computer models of these complex systems. The target audience for using this framework is broad, ranging from senior secondary school pupils to postgraduates. While we assume basic numeracy and some ability to reason logically, we do not rely on prior exposure to specific programming concepts from our users.

The application area, which forms the focus of our investigation, is in describing natural and/or ecological systems, e.g. interaction between bacteria and their environments; predator-prey relationships; flocking; genetic drift. We use the term "complexity" in its common usage to describe systems with many parts or interactions. Reasoning about these complex systems is challenging because, although we may be able to predict the next system state given some current state, it becomes increasingly difficult as we attempt to predict states further into the future. Our meaning of complexity may sometimes partially correlate with algorithmic complexity and “Big O” notation but we are not fundamentally concerned with the performance of algorithms here.

In this paper we consider emergent systems to be a subset of complex systems. Emergent systems are those that exhibit some "radical novelty" (Goldstein 1999, p. 50) or whose macroscopic behaviours are not predictably defined by the behaviours of their parts. In preliminary discussions with target users we found a tendency to assume that emergence (and to a lesser extent complexity) can only be exhibited by large-scale and/or non-deterministic systems. However, we will later illustrate that this may not be the case (see the "Vants" example below).

We do not enter a philosophical debate about determinism, choosing instead to consider determinism from an observational perspective – so if a normal observer could not be expected to have the level of information required to accurately predict all future states/events then a system is observationally non-deterministic. In terms of the notations and reasoning presented in this paper, we consider systems to be non-deterministic if the descriptions of them (formal or verbal) draw on probabilistic behaviour.

Representations and Reasoning

For the purpose of this investigation, we view our representational framework from two different perspectives; as a notation suitable for specifying computer models and also as an "external representation" to support reasoning. Discussion about internal and external representations has described internal representations as the "knowledge and structure in individuals' minds" and external representations as those in the external environment (Zhang & Patel 2006, p. 334). In our case this external representation is the notation used to describe systems, i.e. the program code (though we form a less restricted view as we develop the ideas in this paper). Traditional approaches to cognition consider external representations as "peripheral aids" typically operating as notations used only to aid memory (Wang, Johnson, Sun & Zhang 2005; Zhang & Wang 2009) and in computing, code is regarded as a product – the result of some problem solving activity – but this presents too narrow a view of the potential value of external representations.

Cox (1999) and Lehrer (2006) argue that external representations, models in particular, enable students to map the natural world to the representational world. It is this externalisation, where the demands of reasoning are distributed across the external representations (forming a distributed cognition system) which enables students to work with their ideas and to engage in reasoning (Zhang & Norman 1994; Zhang 1997a, 1997b; Zhang 1998; Xu, Tytler, Clarke & Rodriguez 2012) – the external and internal representations interact to enable reasoning and exploration. Prain and Tytler (2012) suggest that external representations can be considered as affordances; they provide individuals with opportunities to achieve certain reasoning or problem solving outcomes. Consider, for example, the mathematical system for writing numbers as an external representation and the meaning of numbers and symbols as an internal representation. Some tasks only become possible (long division is an example for most people) when both representations are used. Moreover, many mathematical proofs and formulas have only been discovered with the aid of such external representations. Thus representations are not only products of inquiry that reflect current understanding, but the use of representations is a process that promotes the development of new understanding (Carolan, Prain, Waldrip 2008; Tytler & Prain 2010). Representations afford reasoning.

Following other research (Lemke 2003, 2004; Waldrip, Prain & Carolan 2010; Prain & Tytler 2012), we use Peirce's (1998a; 1998b) novel conceptualisation of representations and reasoning to explore software models as external representations that afford reasoning. Peirce proposes a specific relationship between representations and reasoning, encapsulated in his notion of *logic as semiotic*. Peirce (1998b) argues that "logic is the art of reasoning" (p. 11) and that reasoning is the process "to find out, from the consideration of what we already know, something else which we do not know" (1992a, p. 111). Peirce (1992c, 260) describes semiotic as "the science of the general law of signs" with a sign consisting of the triad of the *object* (the thing being represented), the *representamen* (the thing doing the representing), and the *interpretant* (the effect of the relationship between the object and the representamen on those involved in the communicative act) (1998a). The maxim "logic as semiotic" thus defines reasoning as: "the process by which representations operate to make meaning" and reasoning becomes a representational process; one cannot reason without using representations.

Peirce (1992a) delineates a number of different ways by which inquiry can be conducted, identifying the scientific method (with its direct reference to an external reality) as the approach which supports the most effective types of reasoning. Peirce argues that reasoning can be in the form of deduction, induction or abduction (1992b) and considers deduction to be the least productive as it is only analytical in nature, while induction and abduction can synthesize new inference and are thus more productive. Abduction (the process through which hypotheses about reality are generated and then tested through scientific means) is most highly valued, for while induction tends to be classificatory in nature, abduction is explanatory resulting in the production of truly new understanding for the individual. For this reason, representations that afford abduction are the most valuable in developing understandings of complexity.

For our work we are interested in finding an existing representation that will enhance users' mental models (internal representations) and reasoning processes as much as possible. So it is important to find a representation that aids exploration and understanding of the phenomena we wish to study but which also provides a basis for developing computer models. That is: we want some written, external representation to complement users' internal representations and form a partially distributed framework for cognition, producing an external cognitive artefact which facilitates exploration and discovery (i.e. abductive reasoning) while also offering the ability to be executed/interpreted by some kind of computational system. Consequently our representation will behave in a similar way to mathematical/logical representations but will also have the added property that it can be executed, its results observed and that this observation will further enhance its utility as a reasoning system.

In order to evaluate suitable representations we examined the nature of user-constructed representations, since we accept the findings of Prain and Tytler (2012) which indicate "strong conceptual gains and a high level of ... engagement" (p. 2752) for students using self-constructed representations. It is through constructing their own representations, and in so doing making abstractions about scientific observations, as well as engaging with the canonical representations of science that students undertake meaningful learning. In our case we cannot simply adopt any user constructed representation, no matter how semantically valid, because we require a formal representation which can be executed to investigate system properties and to produce observable results. Nonetheless we support a representation-construction pedagogy for teaching and learning science. This is a form of guided inquiry in which tutors support representational development by presenting challenges to students, while also constraining the production of representations to that which is both semantically unambiguous and in fine enough detail to describe the relevant features of a system (Tytler, Haslam, Prain & Hubber 2009; Hubber, Tytler & Haslam 2010). The tutor then channels users towards constructing representations that appropriately model specific phenomena and which begin to approach the efficacy of canonical representations. It is also important for tutors to guide students in the discussion of the adequacy of both user-constructed representations and existing representations and to promote discussion about which representations are suitable for which purposes (diSessa & Sherin 2000; diSessa 2004; Ainsworth, Prain & Tytler 2011). The choice of representation is important; different representations make different features explicit and facilitate different reasoning. Consequently, students are also encouraged to switch between representations and in doing so re-represent their understandings to experience the value of different representations and to further develop their knowledge (Ainsworth 1999; Prain & Waldrip 2006; Prain, Tytler & Peterson 2009).

We adopted a similar approach to the development of tutor-led, user-constructed representations while also requiring our tutors to steer students towards existing representations if this was possible within the context of their problem solving. While we recognise that there is some potential conflict of interest here (the tutor aims to facilitate free construction of representations while also hoping to steer users towards some exiting system) the process is highly informative to the final choice of representation. We also notice that users have a feeling of *ownership* in the choice of representation by participating in this process. And additionally, as argued by Prain and Tytler (2012), when students engage in the process of building their representations they carry out "authentic scientific knowledge-building" (p.2753) and discover new phenomena of the systems they intend to model. In this way their process of engagement has genuine benefit in addition to any product they may develop.

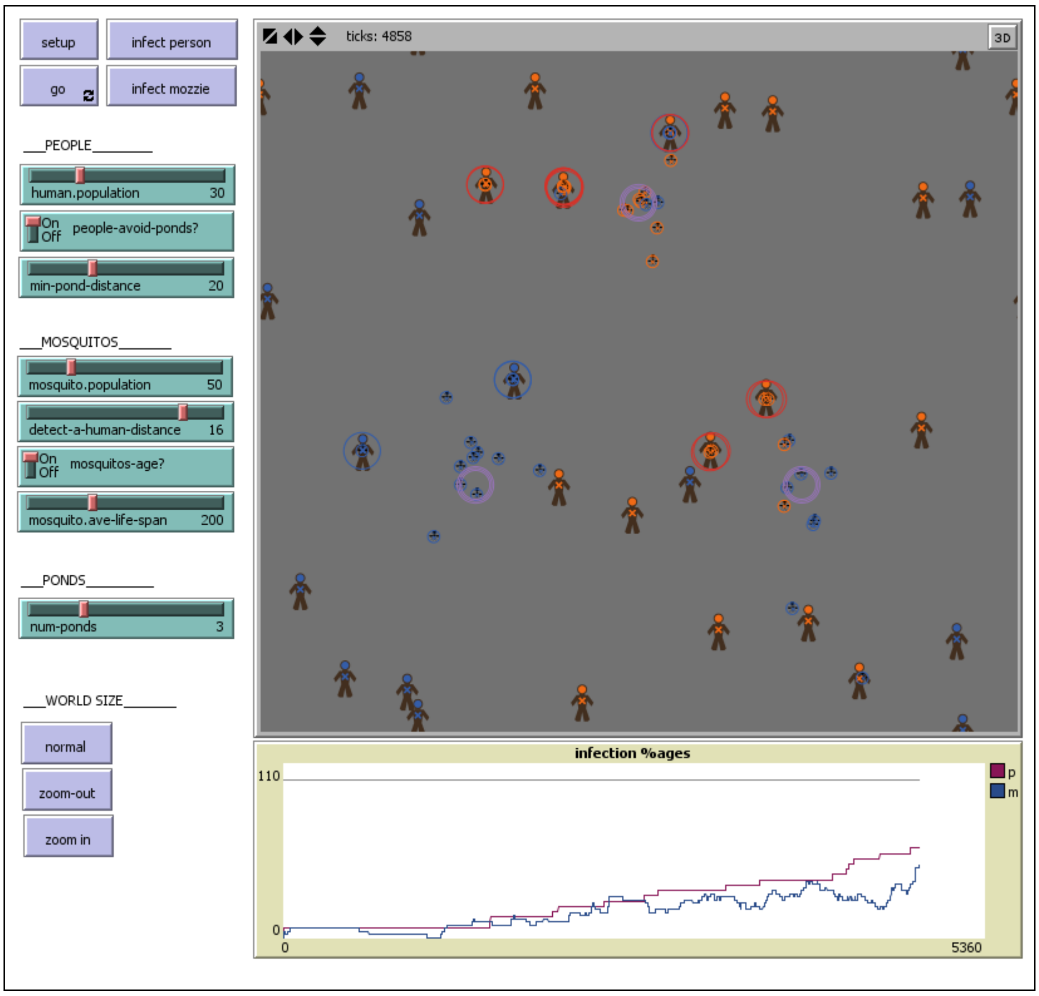
NetLogo as a Candidate Representation

We evaluated various notations and programming languages as candidate representations, these included Java and Netlogo as well as mathematical and logic based notations. NetLogo was included in the study because of the volume of research reporting its successful use in conceptualising complex systems, natural systems in particular (Jacobson & Wilensky 2006; Wilensky & Reisman 2006; Wilenksy & Novak 2010; Levy & Wilensky 2011; Dickes & Sengupta 2013). Our findings concur with this research but we approach the study from a new perspective, that of external representations.

In this paper we concentrate more on the evaluation of the suitability of NetLogo as a representation than a thorough assessment of the relative merits of different representations. We nevertheless make comparisons where appropriate. We assume readers are familiar with the features of Java and mathematical notations, but provide a brief outline of NetLogo since this is less widely used.

NetLogo is an agent-based modelling and programming environment based on M.I.T.'s StarLogo system. Designed for a wide range of audiences, it aims to present a low learning threshold for new users but a high ceiling for advanced experimentation. Specifically it aims to be accessible for novice programmers and users who are not necessarily from a scientific background. NetLogo programs are called "models."

NetLogo provides a programming language where concepts are expressed in terms of agents and their environments (2-dimensional worlds of tiles/patches) and presents an animated graphics panel, the *world*, which shows the actions of agents and their states at run-time. It provides tools for producing control panels (buttons, scrollbars, etc.) and graphs (see Fig 1). Typically, some controls allow the operation of models to be manipulated at run-time thereby modifying the behaviour of running systems and providing an additional level of experimentation. Standard controls allow models to be paused or to have their running speed slowed or increased.



ticks: 4858

human.population 30
On Off people-avoid-ponds?
min-pond-distance 20
mosquito.population 50
detect-a-human-distance 16
On Off mosquitos-age?
mosquito.ave-life-span 200
num-ponds 3

Fig. 1. A NetLogo model showing malaria transmission between humans and mosquitoes.

The figure below (see Fig 2) shows example code for a predator / prey model containing "foxes" and "rabbits". The code shown defines the set-up and activity of rabbits. Code for the foxes is similar. In this model both species start with a low preference for activity (a random percentage between 0% and 10%). The model runs multiple decision-action cycles for foxes and rabbits. When a fox "lands on" a rabbit, the rabbit dies and a new rabbit is cloned. The clone inherits its preference for activity from its parent but randomly adjusts this by +/- 5%. A similar approach is taken with foxes – those who "eat" the fewest rabbits are replaced by the clones of other foxes and these clones also randomly adjust their activity levels when they are created. The model shows how the populations become increasingly active over time as rabbit and fox populations “co-evolve” larger values for their "active%" variables.

```
breed [rabbits rabbit]      ;; rabbits are one type of agent...
rabbits-own [active%]       ;; ...they have a probability of being active

to setup-rabbits            ;; this procedure creates a population of rabbits
  create-rabbits 70                    ;; create 70 rabbits
  ask rabbits
  [ set shape "rabbit"                 ;; set their appearance
    set color white
    setxy random-xcor random-ycor      ;; set random x,y coordinates...
    set active% (random 10)            ;; and a 0%-10% activity probability
  ]
end

to move-rabbits                        ;; this procedure describes the behaviour
  ask rabbits                          ;; of rabbits for each cycle.
  [ if (trigger-probability active%)   ;; if this rabbit is active
    [ face nearest-of foxes            ;; find the nearest fox
      right 180                        ;; turn around 180 deg
      wiggle                           ;; randomly turn a bit left or right
      forward 0.5                      ;; move forward 1/2 step
    ]
  ]
end

to clone-a-rabbit
  ask one-of rabbits      ;; ask a rabbit to clone itself...
  [ hatch 1               ;; ...then mutate the activity of the clone
    [ set active% (randomly-adjust active%) ]
  ]
end
```

Fig 2. Sample code for a co-evolving predator / prey model.

Evaluating NetLogo as an External Representation – Users' Experiences

We evaluate the use of NetLogo as an external representation in two ways. First we examine examples of NetLogo models, how they are specified, how they illustrate complexity and the response of user groups. Secondly (see later section "addressing key criteria") we consider how NetLogo meets the properties of external representations summarised by Zhang & Patel (2006).

Example 1 – Vants

The first example we use here serves to illustrate how complexity and emergence can exist in, and arise from, systems that are deterministic and simple to specify. This example also highlights the use of the NetLogo world as an external representation, which both informs and supports reasoning. The example is a simplified version of Langton's virtual ants (Langton 1986) and based on a similar NetLogo model from Wilensky (2005).

The virtual ant world consists of a 2-dimensional grid of tiles (patches), these (logically) have a black side and a white side and are initialised to be white-side up (they have their colour set to white). In our simplified version there is only one ant. This ant repeatedly flips the tile it is standing on, moves forward one tile, then turns right or left depending on the colour of the new tile it is on. In NetLogo the behaviour of this ant is specified below.

```
ask ants
[ flip-tile
  forward 1
  ifelse (pcolor = white)
  [ right 90 ]
  [ left 90  ]
]
```

The ant starts in a world of white tiles but after 5 moves starts to encounter tiles that it has flipped to black. As the model progresses, the ant wanders chaotically, flipping and re-flipping tiles. After 10,000 cycles the ant has created a figure of tiles with no visibly discernible pattern (see Fig. 3). The virtual ant model is deterministic and its complexity is unnoticed by many users who assume complex systems require multiple agents (or causative entities) and that these agents must interact. In fact the ant world behaves like a 2-dimensional Turing Machine with the colours of the tiles defining a set of binary-encoded instructions. In a generalised model, or models containing multiple ants, various patterns of self-organised, emergent behaviour may occur. Even with our simple model we can observe organised structures emerging after 10,500 cycles (see Fig. 4) and witness other larger scale behaviour patterns (our ant can sometimes be seen "running" up prebuilt columns or "undoing" – reversing patterns of tiles created earlier).

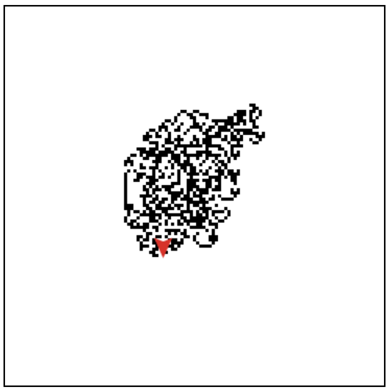


Fig. 3. Vant at 10,000 cycles.

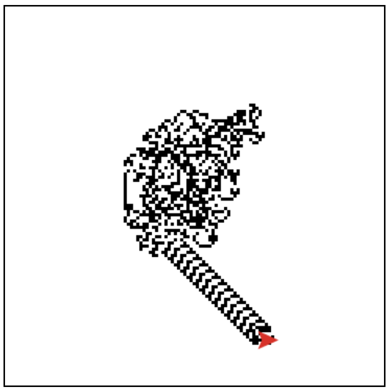


Fig. 4. Vant at 11,500 cycles.

From the perspective of our examination of NetLogo as an external representation there are a number of points of interest:

- the first evidence of emergence, self-organisation and, by implication, complexity arises after 10,000 cycles and is only evident by visual observation, with the NetLogo world providing a conceptualisation of a top-down view of a physical/geographical environment. The evidence of emergence is explicit and self-evident, no analysis is required and the structures are clearly perceived;
- before examining the system using NetLogo as a reasoning tool we found no students (including those who are experienced programmers or mathematicians) to correctly predict the emergent properties of this system from its rules;
- the NetLogo code describing activity of the ant(s) is understood by the youngest and least experienced of our user group who are able (i) to act out the ant behaviour (and other behaviours coded in a similar style) and (ii) to understand (and then explain) the cause of the emergent behaviour when it is investigated using the NetLogo agent inspector;
- before exposure to the NetLogo Vants model, a group of programming students were given the system's rules and asked to code the ant behaviour using a programming language of their choice and report on their findings. Of the 12 respondents, none identified any complex or emergent behaviour, primarily because they either (i) terminated the testing too soon, (ii) failed to visualise behaviour at all or (iii) failed to visualise behaviour from the perspective of the world.

Example 2 – Genetic Drift

Genetic drift describes the process where, over time, random effects cause changes in the frequency of gene variations within a population. In small populations genetic drift can cause some gene types to die out completely, though in the absence of other evolutionary pressures gene frequencies will tend to remain approximately constant in larger populations. Genetic drift is considered to be one of the processes which can influence speciation. It can be modelled mathematically (see later comments) or by using various types of algorithm (Hartl & Clark 2007; Wilensky 1997; Tian 2008). In our implementation (developed independently of Wilensky (1997) but resulting in a similar specification) we model genetic variations using colour on a population of static individuals. At each cycle, one individual has its genetic type replaced by the type of one of its neighbours – this is achieved simply by setting its colour to that of one of its neighbours.

The NetLogo code is specified as follows (note that in this case the agents are named "bugs"):

```
to go
  ask one-of bugs
  [ set color [color] of one-of neighbors ]
end
```

Modelling genetic drift in this way is of interest for various reasons: (i) it operates like a type of cellular automata; (ii) the model can be extended to investigate types of speciation; (iii) of relevance here, the model involves a high level of simple agent-agent interaction which causes it to exhibit emergent properties. See Fig. 5 and Fig. 6 for initial and later states of low population models and Fig. 7 for a later state of a large population model where the visual effect is similar to that of cloud formation.

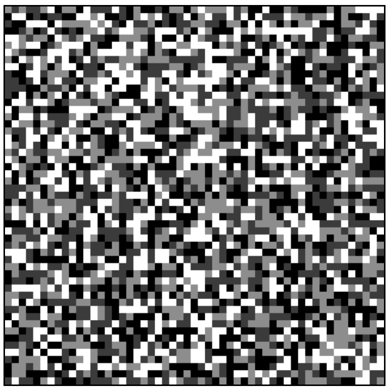


Fig. 5. Drift – initial

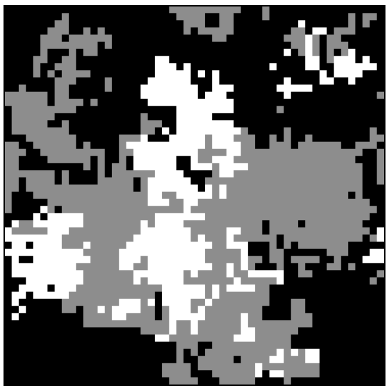


Fig. 6. Drift – later

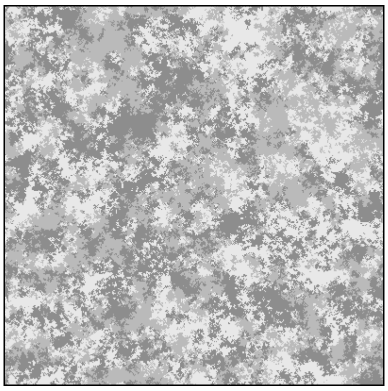


Fig. 7. Drift – large scale

Note that an important attribute for external representations is that they facilitate experimentation (i.e. abduction). As an example of this consider adapting the model to extend the neighbourhood for selection. Two steps are required: (i) adding a slider/scrollbar to the interface controlling a numeric variable *neighbourhood*; and (ii) making a small change to the code used earlier, as shown below.

```
to go
  ask one-of bugs
  [ set color [color] of one-of bugs in-radius neighbourhood ]
end
```

The genetic drift model appears deceptively simple in NetLogo suggesting that genetic drift may be fundamentally simple, however this is not supported by our investigation. One of our advanced user groups (final year computing undergraduates) was shown two alternative mathematical equations (Fig. 8) describing the probabilistic behaviour of a simple genetic drift process (in which selection occurs globally rather than with near neighbours). They were unable to recognise the equations as relating to genetic drift or reason with them to predict systems behaviour.

$$\text{(i)} \quad Y_n = \prod_{i=1}^{n} \frac{g(X_i)}{f(X_i)} \qquad \text{(ii)} \quad \frac{(2N)!}{k!(2N-k)!} p^k q^{2N-k}$$

Fig. 8. Two probabilistic equations describing aspects of genetic drift.

Our advanced group were split into 8 development teams (of 3-4 students in each) and tasked to produce their own models of genetic drift using Java, C# or C++. Some teams failed to produce working models or incorrectly specified their models; in one model, for example, a “bug” would reset itself to the most commonly occurring type/colour rather than to a random choice of its neighbouring types, causing a significant change in the emergent behaviour. Of those teams who had correctly specified their models some still failed to observe aspects of behaviour including: (i) localised convergence (clustering) of types; and (ii) extinction of types with smaller populations. Only 3 of the 8 teams correctly observed behaviour, this contrasts with the level of success of teams using NetLogo (who all made relevant observations).

Evaluating NetLogo as an External Representation – Addressing Key Criteria

An external representation acts as no more than a memory aid, having limited utility, if it simply mirrors an internal representation. In order to form a distributed cognitive environment, external representations need to augment and expand the capacity of internal representations (Ainsworth 1999, 2006; Cox 1999). It is important that there is a conceptual correspondence between the internal and external representations otherwise the burden of translating between them outweighs any benefit offered.

Our observations show that NetLogo functions well as an external representation with users in our problem domain; its different facets (code, world and UI controls) allow it to interact with users' internal representations in a variety of ways (see Fig. 9).

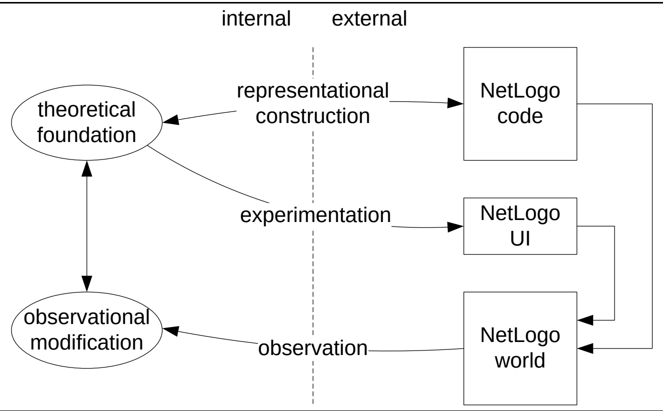


Fig. 9. Interaction between internal representation and NetLogo.

Zhang & Patel (2006) summarise properties of external representations, specifically that they provide the following (list edited from Zhang & Patel):

1 memory aids (to reduce memory load);
2 directly perceived information;
3 knowledge/skills unavailable internally;
4 support for easy recognition & direct inference;
5 effortless support for cognitive behaviour;
6 generation of more efficient action sequences;
7 facilities to stop time, supporting perceptual rehearsal with visible & sustainable information;
8 reduced need for abstractions;
9 aids to decision making (accuracy & effort).

We observe that the process of producing NetLogo code is itself constructive for reasoning and the development of understanding. Though more investigation is required in this area of study, initial findings suggest that this effect is more prevalent with NetLogo code than with other programming languages. We propose possible reasons for this:

- tutors emphasise the importance of the *process* of code production rather than the model as an artefact. The primary criterion for success is identified as understanding a system; the production of a robust, fully functioning model is a secondary goal. Clearly this approach can also be taken with other languages, but the mind-set of our students tends to consider the use of other languages as an exercise in program specification not learning;
- NetLogo code describes activity in terms of the actions of individual agents, this produces a subtle shift in the conceptualisation of code elements from "objects having things done to them" to "agents interacting in their world". This influences the way students discuss their code, causing them to personalise (or anthropomorphise) their agents' behaviour and to more easily rationalise the macroscopic behaviours of their systems as resulting from the collective actions of a population of individuals;
- for the types of population-based multi-agent systems we have used in our study, the level of abstraction provided by NetLogo code matches the style of students' verbal description of agent activity. Its primitives and constructs lend themselves to these types of system, limiting the need for students to create additional abstractions. This is in contrast to development in languages like Java, C#, etc. where programmers need to specify the lower-level mechanics of interactions between agents and the worlds they inhabit;
- NetLogo code production is closely coupled with model experimentation, so phases of code production are shorter and more informed by model observation (the details of our data collection and analysis are beyond the scope of this paper).

The animated NetLogo world (the graphics panel) showing the movement, birth and death of agents, with facilities for pausing behaviour and slow-motion, addresses most of the representational facets identified by Zhang and Patel (2006) and listed earlier. In contrast to program code and other notation-based representations, the changing system state is directly perceivable in NetLogo and may be cross-referenced with graphs included in models. Agents of different types/breeds are easily recognised as they are typically represented using icons of different shapes and colours. Additionally agent-agent interaction is made observable and explicit with the use of other graphical tools.

NetLogo control panels allow running models to be changed dynamically, altering models as they run. This enhances experimentation by allowing users to immediately visualise the consequences of modifying the attributes and behaviours of agents – the agents' new behaviour will be immediately observed. In this way NetLogo clearly affords opportunities for abduction. These controls work in conjunction with other (default) controls that allow agents to be paused, their worlds to be increased/decreased in size, etc. and an agent "inspector" which provides facilities to examine the internal states of agents and "follow" their individual movements in the graphics world.

By following the discussions of student groups, using an ethnographic approach, we observe trends in the behaviour of students developing NetLogo models:

1 students engage in phases of observation and experimentation, which directly inform code modification and structure plans for further experimentation. We notice that experimentation tends to focus more on system behaviour than testing code (in contrast to the use of other languages) and a greater tendency to push experiments to the point at which behaviour "breaks" revealing the thresholds of system dynamics;

2 a continued tendency to anthropomorphise agents when discussing activity seen in the graphics world;

3 an ability to discuss modelling abstractions with increased clarity and understanding. In assessing this we used various models, which used either one-to-one or one-to-many relationships between NetLogo agents and their natural analogs.

The part of our study discussed above involved study teams of student programmers. These students were at intermediate or advanced levels of competence with Java, C# or C++ and had some additional competence with other languages. They were given a one-hour overview of NetLogo, which included the use of models and building control panels as well as NetLogo code. They were then asked to experiment with some pre-constructed models and were later asked to build software to investigate other systems. Most of our findings here have concentrated on this, but it is also interesting to note that subsequently:

- 16 groups investigated swarm dynamics (including bacterial activity, altruistic and parasitic behaviour, disease pandemics, etc.), all 16 groups chose to use NetLogo, self-learning the platform to the necessary level;
- 14 of 16 groups chose NetLogo to model evolution;
- 6 of 8 groups used NetLogo as an exploratory tool for designing agent deliberation for a collaborative problem solving system (the others described the system at a more abstract level using BDI style plans);
- the quality of student models as well as their investigations compared favourably with work they undertook in other languages (based on results for assessed work).

Conclusions

NetLogo has been successfully used by various researchers from education and science to model complex systems and teach programming. For our work, the models themselves are less important than the investigation into complex systems that modelling may afford. With this approach we are less interested in programmed computer models as artefacts and more interested in the processes that are used to formulate them. Consequently, while other work has examined NetLogo as a modelling tool or as a system to teach programming to novices, we have evaluated NetLogo from an alternative perspective: that of an external representation that affords reasoning.

Specifically, we have examined how well NetLogo serves to support the perception and analysis of complex systems, particularly those exhibiting some emergent property, and how it functions to provide opportunities for reasoning about complex systems. We have achieved this by following groups of students when they experimented with pre-constructed NetLogo models and also when they built their own models. We have also considered how NetLogo addresses the features of external representations identified by Zhang and Patel (2006) among others.

Our findings strongly support the use of NetLogo as a reasoning system to improve understanding of complexity with both novice and advanced users. We have observed that the different facets of NetLogo (code, control panel and graphics environment) interact with users’ internal representations, serving to inform their experimentation and complement their understanding. In this way Netlogo operates as an external representation which facilitates cycles of hypothesis and test, thereby promoting abductive enquiry and exploration. We find that this compares favourably to using other languages and notations that typically offer different types of language semantics and more restricted representational forms. In addition, we find that NetLogo fulfils the criteria for external representations as they are defined in related work, supporting users’ reasoning with tools to animate system behaviours, which makes the interaction of system entities explicit and the emergent properties of systems visually observable.

References

Ainsworth, S. (1999). The functions of multiple representations. *Computers & Education, 33* (2-3), 131-152.

Ainsworth, S. (2006). DeFT: A conceptual framework for considering learning with multiple representations. *Learning and Instruction, 16* (3), 183-198.

Ainsworth, S., Prain, V., & Tytler, V. (2011). Drawing to learn in science. *Science, 333* (6046), 1096-1097.

Carolan, J., Prain, V., & Waldrip, B. (2008). Using representations for teaching and learning science. *Teaching Science: The Journal of the Australian Science Teachers Association*, *54* (1), 18-23.

Cox, R. (1999). Representation construction, externalised cognition and individual differences. *Learning & Instruction, 9* (4), 343-363.

Dickes, A., & Sengupta, P. (2013). Learning natural selection in 4th grade with multi-agent-based computational models. *Research in Science Education*, *43* (3), 921-953.

diSessa, A. A., & Sherin, B. L. (2000). Meta-representation: An introduction. *Journal of Mathematical Behavior*, *19* (4), 385-398.

diSessa, A. A. (2004). Metarepresentation: Native competence and targets for instruction. *Cognition and Instruction*, *22* (3), 293-331.

Goldstein, J. (1999), Emergence as a construct: History and issues. *Emergence: Complexity and Organization*, *1* (1): 49–72.

Hartl, D., & Clark, A. (2007). Principles of Population Genetics (4th ed.). Sinauer Associates. p. 112.

Hubber, P., Tytler, R., & Haslam, F. (2010). Teaching & learning about force with a representational focus: Pedagogy and teacher change. *Research in Science Education, 40* (1), 5-28.

Jacobson, M. J., & Wilensky, U. (2006). Complex systems in education: Scientific and educational importance & implications for learning sciences. *Journal of the Learning Sciences*, *15* (1), 11-34.

Langton, C. G. (1986). Studying artificial life with cellular automata. *Physica D: Nonlinear Phenomena*, *22* (1-3): 120–149.

Lehrer, R., & Schauble, L. (2006). Cultivating model-based reasoning in science education. In R. K. Sawyer (Ed.), *The Cambridge handbook of the learning sciences* (pp. 371-387).

Lemke, J. (2003). Mathematics in the middle: Measure, picture, gesture, sign, and word. In M. Anderson, A. Saenz-Ludlow, & V. V. Cifarelli (Eds.), *Educational perspectives on mathematics as semiosis: From thinking to interpreting to knowing* (pp. 215-234). Ottawa: Legas Publishing

Lemke. (2004). The literacies of science. In E. W. Saul (Ed.), *Crossing borders in literacy and science instruction: Perspectives on theory and practice* (pp. 33-47). Newark: International Reading Association/National Science Teachers Association

Levy, S. T., & Wilensky, U. (2011). Mining students' inquiry actions for understanding of complex systems. *Computers & Education, 56* (3), 556-573.

Peirce, C. S. (1998a). What is a sign? In N. Houser, A. De Tienne, J. R. Eller, C. L. Clarke, C. Lewis, & D. B. Davis (Eds.), *The essential Peirce – Selected philosophical writings – Volume 2 (1893 – 1913)* (Vol. 2, pp. 4-10). Bloomington: Indiana University Press.

Peirce, C. S. (1998b). Of reasoning in general. In N. Houser, A. De Tienne, J. R. Eller, C. L. Clarke, C. Lewis, & D. B. Davis (Eds.), *The essential Peirce – Selected philosophical writings – Volume 2 (1893 – 1913)* (Vol. 2, pp. 11-26). Bloomington: Indiana University Press.

Peirce, C. S. (1992a). The fixation of belief. In N. Houser, & C. Kloesel (Eds.), *The essential Peirce – Selected philosophical writings – Volume 1 (1867 – 1893)* (Vol. 1, pp. 109-123). Bloomington: Indiana University Press.

Peirce, C. S. (1992b). Deduction, induction and hypothesis. In N. Houser, & C. Kloesel (Eds.), *The essential Peirce – Selected philosophical writings – Volume 1 (1867 – 1893)* (Vol. 1, pp. 186-199). Bloomington: Indiana University Press.

Peirce, C. S. (1992c). An outline classification of the sciences. In N. Houser, A. De Tienne, J. R. Eller, C. L. Clarke, C. Lewis, & D. B. Davis (Eds.), *The essential Peirce – Selected philosophical writings – Volume 2 (1893 – 1913)* (Vol. 2, pp. 258-262). Bloomington: Indiana University Press.

Prain, V., Tytler, R., & Peterson, S. (2009). Multiple representations in learning about evaporation. *International Journal of Science Education, 31* (6), 787-808.

Prain, V., & Tytler, R. (2012). Learning through constructing representations in science: A framework of representational construction affordances. *International Journal of Science Education, 34* (17), 2751-2773.

Prain, V., & Waldrip, B. (2006). An exploratory study of teachers' and students' use of multi-modal representations in primary science. *International Journal of Science Education, 28* (15), 1843-1866.

Tian, J. P. (2008). Evolution algebras and their applications. Lecture Notes in Mathematics 1921. Berlin: Springer-Verlag. p. 11.

Tytler, R., Haslam, F., Prain, V., & Hubber, P. (2009). An explicit representational focus for teaching and learning about animals in the environment. *Teaching Science: The Journal of the Australian Science Teachers Association, 55* (4), 21-27.

Tytler, R., & Prain, V. (2010). A framework for re- thinking learning in science from recent cognitive science perspectives. *International Journal of Science Education, 32* (15), 2055-2078.

Waldrip, B., Prain, V., & Carolan, J. (2010). Using multi-modal representations to improve learning in junior secondary science. *Research in Science Education, 40* (1), 65-80.

Wang, H., Johnson, T. R., Sun, Y., & Zhang, J. (2005). Object location memory: The interplay of multiple representations. *Memory & Cognition, 33* (7), 1147-1159.

Wilensky, U. (1997). Models of genetic drift. http://ccl.northwestern.edu/netlogo/models/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wilensky, U. (2005). NetLogo Vants model. http://ccl.northwestern.edu/netlogo/models/Vants. Center for Connected Learning, Northwestern University, Evanston, IL.

Wilensky, U., & Reisman, K. (2006). Thinking like a wolf, a sheep, or a firefly: Learning biology through constructing and testing computational theories—An embodied modeling approach. *Cognition and Instruction, 24* (2), 171-209.

Wilensky, U., & Novak, M. (2010). Teaching and learning evolution as an emergent process: The BEAGLE project. In R. S. Taylor, & M. Ferrari (Eds.), *Epistemology and science education: Understanding the evolution vs. intelligent design controversy* (pp. 213-243). New York: Routledge.

Xu, L., Tytler, R., Clarke, D., & Rodriguez, C. (2012). The value of multi-theoretic analyses: Representational & distributed cognition perspectives on classroom sequence about matter. Manuscript submitted for publication.

Zhang, J., & Norman, D. A. (1994). Representations in distributed cognitive tasks. *Cognitive Science, 18* (1), 87-122.

Zhang, J. (1997a). The nature of external representations in problem solving. *Cognitive Science, 21* (2), 179-217.

Zhang, J. (1997b). Distributed representation as a principle for the analysis of cockpit information displays. *International Journal of Aviation Psychology, 7* (2), 105-121.

Zhang, J. (1998). A distributed representation approach to group problem solving. *Journal of the American Society for Information Science, 49* (9), 801-809.

Zhang, J., & Patel, V. L. (2006). Distributed cognition, representation, and affordance. *Pragmatics & Cognition, 14* (2), 333-341.

Zhang, J., & Wang, H. (2009). An exploration of the relations between external representations and working memory. *PLoS One, 4* (8), 1-10.