From Diagrams To Programs

Teacher Notes to support Tenderfoot Unit 5: Theoretical Computers – Fun with finite-state machines

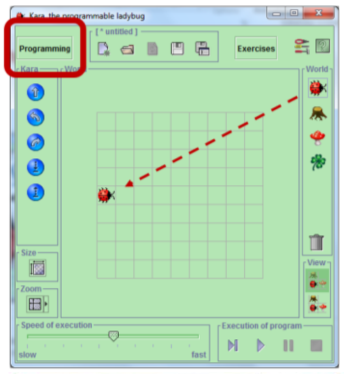
Kara is a programmable ladybird. This activity uses Kara as a vehicle for introducing programming and alternative ways of thinking about algorithms.

Preparation required:
Kara available on class computers. Check to ensure security settings allow Java based applications.
Kara manual for each student as reference.

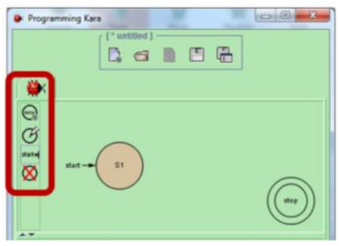
Finite-State Machines

This is a practical session looking at introducing programming through Finite State Machines. The application we'll use, Kara, was born in the research done by Raimond Reichert at ETH, Zurich (a leading STEM University) in the years 1999-2003. He was looking at ways that theoretical computers could be used to introduce the fundamental concepts of Computer Science and was an early advocate of making computer science a part of general school education. He wrote, "Programming practiced as an educational exercise … is best learned in a toy environment, designed to illustrate selected concepts in the simplest possible setting. The fundamental concepts of programming may be intellectually demanding, but they are not complex in the sense of requiring mastery of lots of details. Instead of using a programming language, we use a simpler model … finite-state machines." This approach has certain advantages. Reichert goes on to say, "It is easy to represent them in a graphical manner. Paths of execution are defined statically as paths in a directed graph; no other control structures are needed." A program can be defined by creating a state diagram, and children can quickly produce diagrams that can do fairly powerful things.

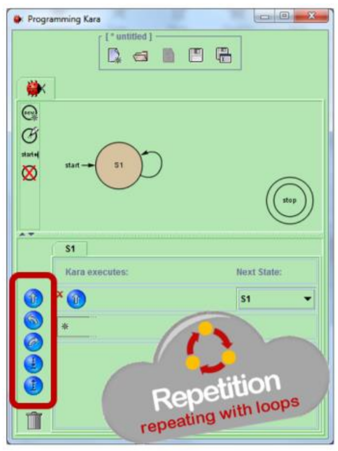
Introducing Kara



Kara, a Java application is free to download (goo.gl/OPZKor) and requires no installation. It comes with a handy 7 page manual which you can distribute to children. Both are included in the resources. The presentation is a practical introduction designed to be followed by the class. When you run the application, Kara's world opens. It is a very simple world, consisting of a grid of squares. Each square can have leaves, mushrooms or trees in them, but we start by just placing Kara in her world (by dragging).



The Programming button opens a second window where pupils build their Finite State Machine. It already has an end state in the diagram. Down the left hand side are controls to let you create, edit and delete states. Create a new state, named S1. The new state appears in your diagram. If you hover the mouse over the state, with the MIDDLE highlighted, you can move it into the centre of the diagram and make it the Start State, by selecting that option on the left.

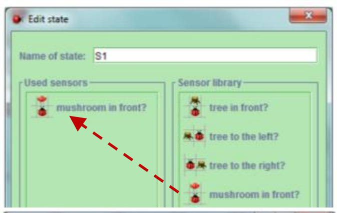


With the mouse over the EDGE of the state, we can draw our transitions. Initially, draw a line, back to itself, as shown. In the lower half of the window, the transition has been recorded. Note the tab, which indicates which state we have selected. Notice also the 'Next State', which indicates we remain in S1.

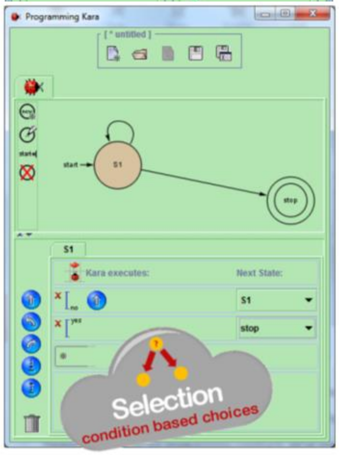
Look at the options on the left. These allow Kara to move forward, turn left and right, pick up or put down an object. Add a move forward command to the transition definition (simply drag it across). When executed Kara will move forward one square, return to S1, move forward again, return to S1 and so on, moving forward continually. We've written our first program, expressed as a Finite State Machine, which implements a continuous loop.

There are only 3 constructs in programming: Sequence, Selection and Repetition.

We can make Kara's world more exciting by adding a mushroom! Kara should continue moving forward until she bumps into a mushroom. Then she should stop. How might we express this in the FSM?



The discussion should draw out the need for a transition to the Stop state, but also the need to test for a mushroom at S1. We need to add a sensor, the input from which will determine which transition to take. With S1 selected, activate the Edit State option and add a mushroom sensor.

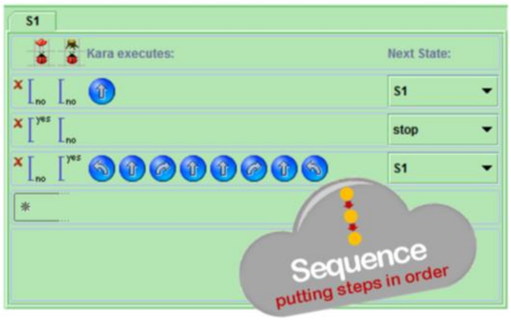


When you return to the Programming window, notice how the sensor has been added to the state definition. We can now set different actions in answer to the condition by selecting the Yes or No option, and adding a second statement moving Kara to the Stop state.

Thinking again about our programming constructs, we've now implemented an IF statement, or selection. The state diagram will update and when the program is executed Kara should stop at the mushroom.

The final part of the presentation adds a tree into Kara's path. When Kara detects a tree, we want her to go round it, before continuing on her way. Check the students can recount the steps involved:

- Edit S1
- Add a Tree sensor
- Assign the correct actions to each condition



With two sensors there are four possible combinations. In this case, we don't need to consider what would happen if Kara sensed both a tree and a mushroom in front because only one element can be on one square. However, students will need to consider all combinations when trees are detected to the side, whilst mushrooms are in front.

There are similar considerations with leaves, which are detected when they are underneath Kara. These are encountered in the tasks, and can be linked to work with students developing truth tables. Notice here how we have a sequence of actions triggered by a condition, the third of our key programming constructs.

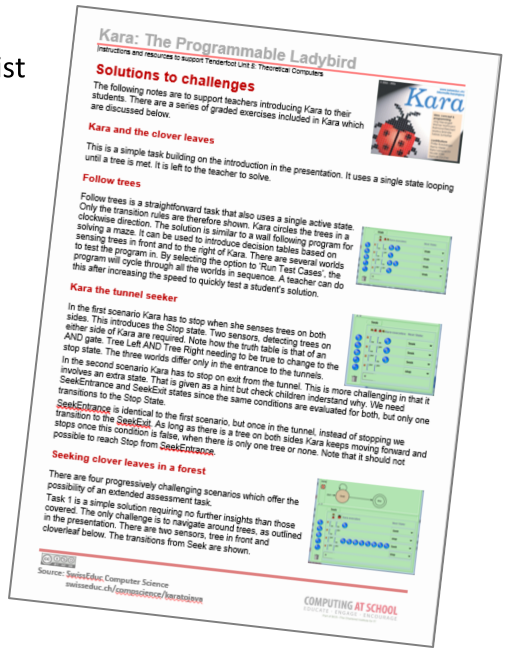
Using the usual icon saves a finite-state machine as a file with a .kara extension. Note that it will not save the world you have configured. To do that, you will need to use the same save icon in Kara's world. These are saved as files with a .world extension

Kara comes with a series of graded exercises built in. The drop down list in Kara's world reveals what is available.

The easy challenges should be accessible to KS3 children, whilst the harder challenges provide plenty of scope for differentiation and extension work.

As well as the explanation of the task, each challenge comes with several preconfigured worlds to try your solution. Notice that a world can be constrained to just a few tiles, rather than the whole grid.

Solutions and hints to the easy challenges are provided in the resources but it is worth teachers trying to solve as many as possible for themselves so they can appreciate any issues students will encounter.


Kara: The Programmable Ladybird

Instructions and resources to support Tenderfoot Unit 5: Theoretical Computers

Solutions to challenges

The following notes are to support teachers introducing Kara to their students. There are a series of graded exercises included in Kara which are discussed below.

Kara and the clover leaves

This is a simple task building on the introduction in the presentation. It uses a single state looping until a tree is met. It is left to the teacher to solve.

Follow trees

Follow trees is a straightforward task that also uses a single active state. Only the transition rules are therefore shown. Kara circles the trees in a clockwise direction. The solution is similar to a wall following program for solving a maze. It can be used to introduce decision tables based on sensing trees in front and to the right of Kara. There are several worlds to test the program in. By selecting the option to 'Run Test Cases', the program will cycle through all the worlds in sequence. A teacher can do this after increasing the speed to quickly test a student's solution.

Kara the tunnel seeker

In the first scenario Kara has to stop when she senses trees on both sides. This introduces the Stop state. Two sensors, detecting trees on either side of Kara are required. Note how the truth table is that of an AND gate. Tree Left AND Tree Right needing to be true to change to the stop state. The three worlds differ only in the entrance to the tunnels.

In the second scenario Kara has to stop on exit from the tunnel. This is more challenging in that it involves an extra state. That is given as a hint but check children inderstand why. We need SeekEntrance and SeekExit states since the same conditions are evaluated for both, but only one transitions to the Stop State.

SeekEntrance is identical to the first scenario, but once in the tunnel, instead of stopping we transition to the SeekExit. As long as there is a tree on both sides Kara keeps moving forward and stops once this condition is false, when there is only one tree or none. Note that it should not possible to reach Stop from SeekEntrance.

Seeking clover leaves in a forest

There are four progressively challenging scenarios which offer the possibility of an extended assessment task.

Task 1 is a simple solution requiring no further insights than those covered. The only challenge is to navigate around trees, as outlined in the presentation. There are two sensors, tree in front and cloverleaf below. The transitions from Seek are shown.

Source: SwissEduc Computer Science swisseduc.ch/compscience/karatojava

COMPUTING AT SCHOOL
EDUCATE · ENGAGE · ENCOURAGE