Thinking About Programming Programmer & Machine

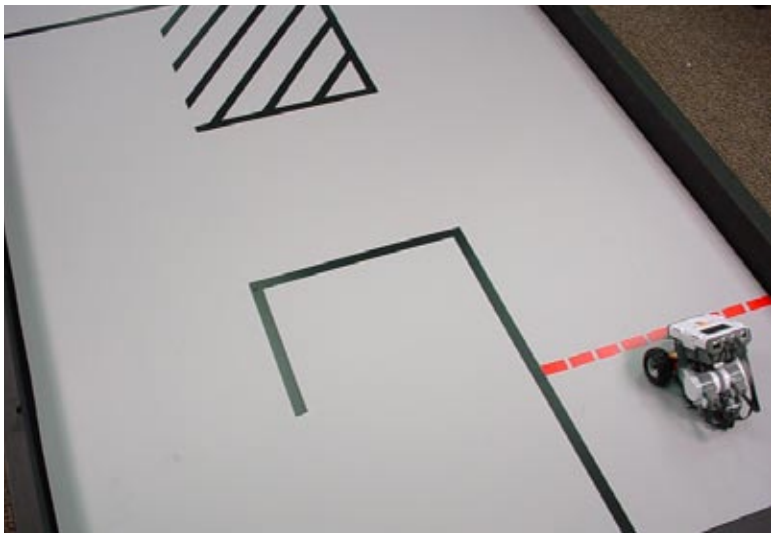
In this lesson, you will learn about the roles of the programmer and the robot, and how the two need to work together in order to accomplish their goals.

Robots are made to perform useful tasks. Each one is designed to solve a specific problem, in a specific way.

Robotic Tractor

Problem:
Drive safely through a field which may contain obstacles

Solution:
Move towards the destination, making small detours if any obstacles are detected



Labyrinth Robot

Problem:
Get through the maze

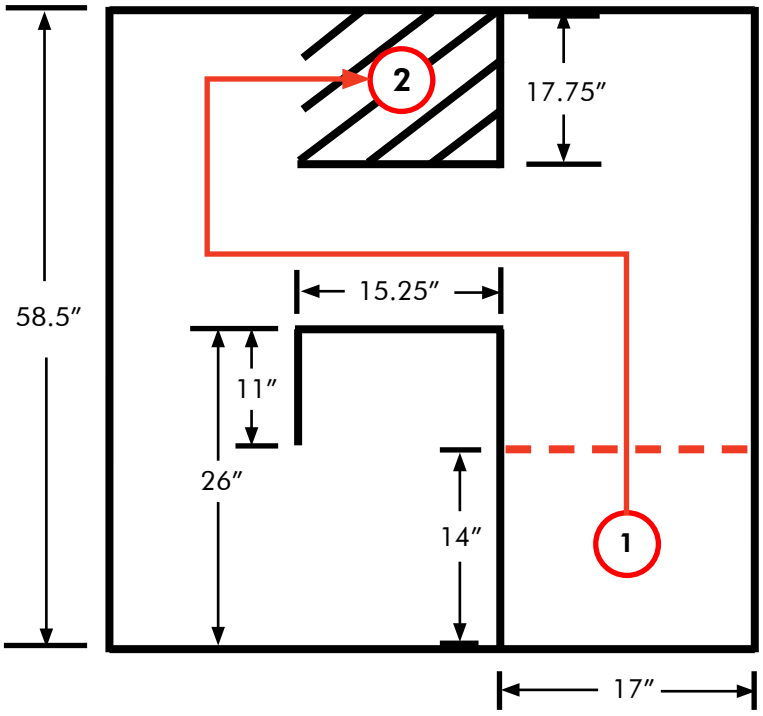
Solution:
Move along a predetermined path in timed segments

Thinking about Programming Programmer & Machine (cont.)

Let's take a closer look at this last robot. How does it do that? How does it know to do that?

Problem → ?????? → **Goal Reached**

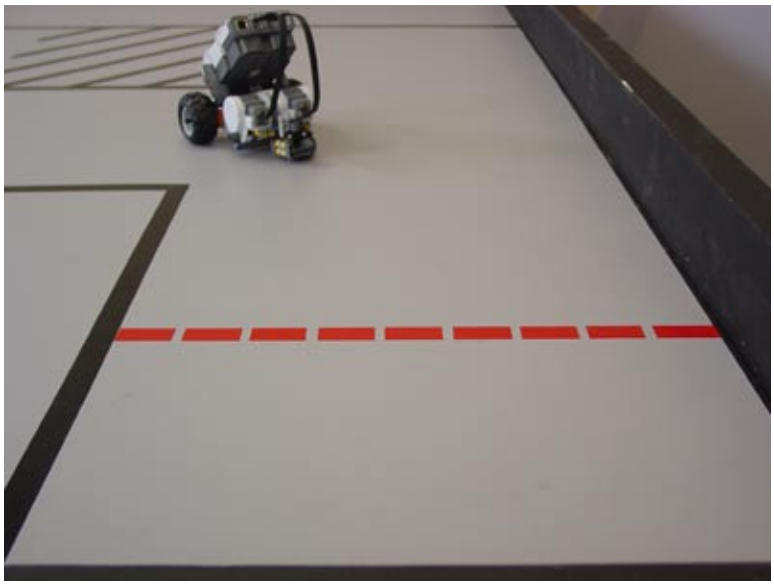
Creating a successful robot takes a team effort between humans and machines.



Role of the Programmer

The human is responsible for identifying the task, planning out a solution, and then explaining to the robot what it needs to do to reach the goal.

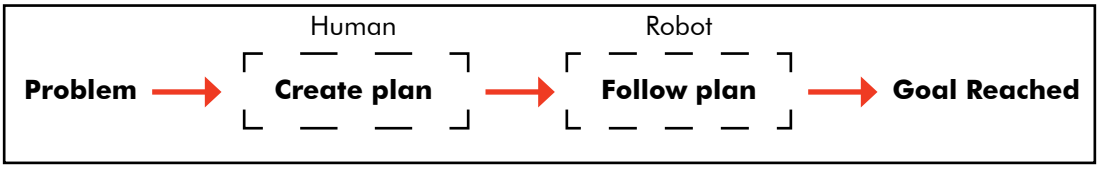
Thinking about Programming Programmer & Machine (cont.)



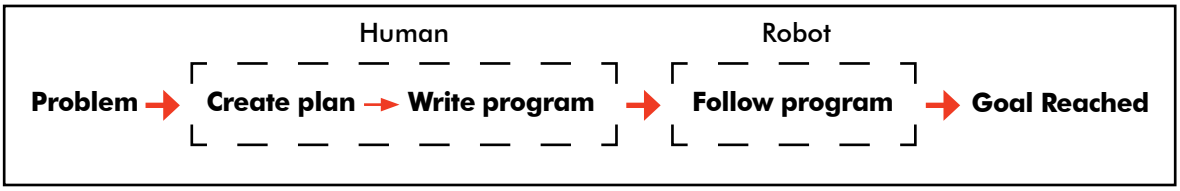
Role of the Robot

The machine is responsible for following the instructions it is given, and thereby carrying out the plan.

The human and the robot can accomplish the task together by dividing up the responsibilities. The human programmer must come up with the plan and communicate it to the robot, and the robot must follow the plan.



Because humans and machines don't normally speak the same language, a special language must be used to translate the necessary instructions from human to robot. There are many such languages, with ROBOTC being one of them. These human-to-robot languages are called "programming" languages, and instructions written in them are called "programs".



Thinking about Programming Programmer & Machine (cont.)

End of Section

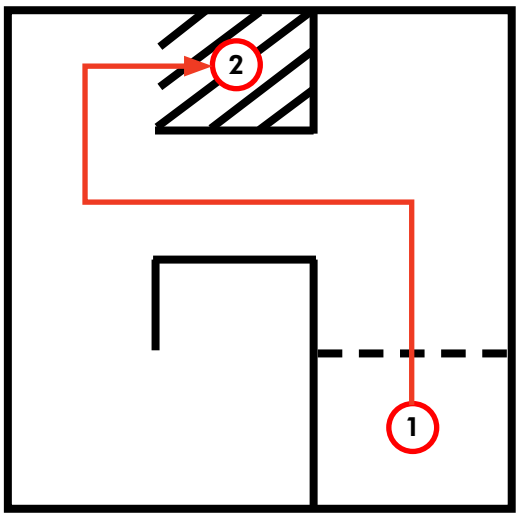
The human who writes the program is called the programmer. The programmer's job, therefore, is to identify the problem that the robot must solve, to create a plan to solve it, and to turn that plan into a program that the robot can understand. The robot will then run the program, and perform the task.

Finally, take note: the robot only follows the program, it does not think for itself. Just as it can be no *stronger* than it is built, the robot can be no *smarter* than the program that the human programmer gave it. You, as programmer, will be responsible for planning and describing to the robot exactly what it needs to do to accomplish its task.

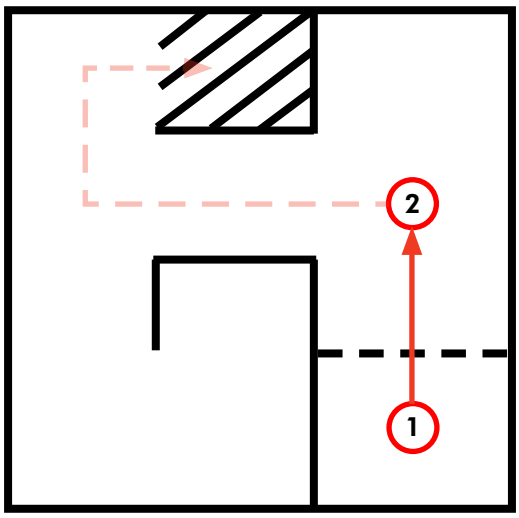
Thinking About Programming Planning & Behaviors

In this lesson, you will learn how thinking in terms of "behaviors" can help you to see the logic behind your robot's actions, and break a big plan down into practical parts.

"Behaviors" are a very convenient way to talk about what the robot is doing, and what it must do. Moving forward, stopping, turning, looking for an obstacle… these are all behaviors.



Complex Behavior
Some behaviors are big, like "solve the maze."



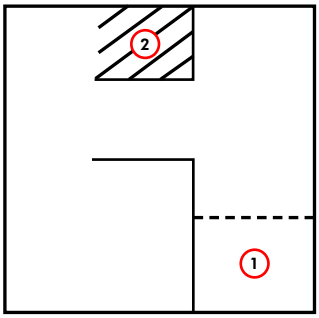
Basic or Simple Behavior
Some behaviors are small, like "go forward for 3 seconds." Big behaviors are actually made up of these smaller ones.

As you begin the task of programming, you should also begin thinking about the robot's actions in terms of behaviors. Recall that as programmer, your primary responsibilities are:

- **First**, to formulate a plan for the robot to reach the goal,
- And **then**, to translate that plan into a program that the robot can follow.

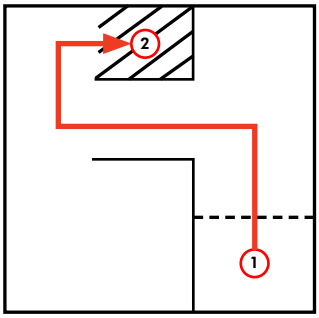
The plan will simply be *the sequence of behaviors that the robot needs to follow*, and the program will just be those behaviors translated into the programming language.

Thinking about Programming Planning & Behaviors (cont.)



1. Examine problem
To find a solution, start by examining the problem.

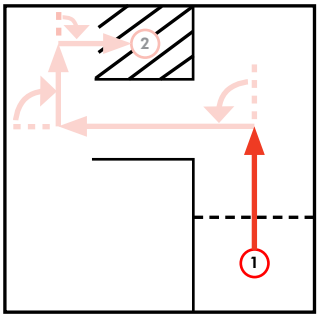
Here, the problem is to get from the starting point (1) to the goal (2).



2. Broad solution
Try to see what the robot needs to do, at a high level, to accomplish the goal.

Having the robot follow the path shown on the left, for example, would solve the problem.

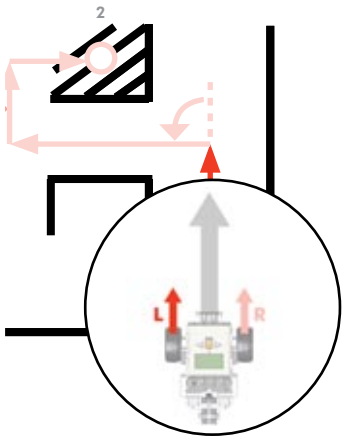
You've just identified the first behavior you need! Write it down.

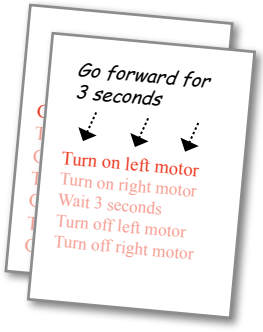


3. Break solution into smaller behaviors
Now, start trying to break that behavior down into smaller parts.

Following this path involves moving forward, then turning, then moving forward for a different distance, then turning the other way, and so on. Each of these smaller actions is also a behavior.

Write them down as well, taking care to keep them in the correct sequence.



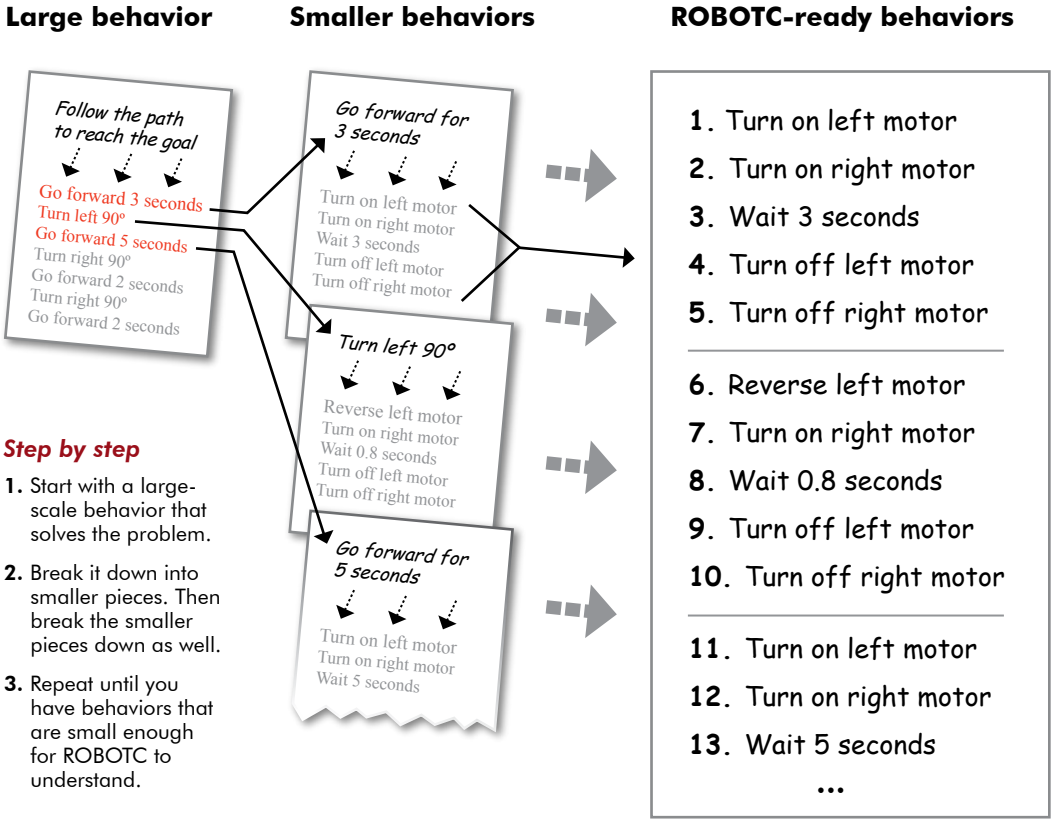


4. Break into even smaller pieces
If you then break down these behaviors into even smaller pieces, you'll get smaller and smaller behaviors, with more and more detail. Keep track of them as you go.

Eventually, you'll reach commands that you can express directly in the programming language.

For example, ROBOTC has a command to turn on one motor. When you reach a behavior that says to turn on one motor, you can stop breaking it down, because it's now ready to translate.

Thinking about Programming Planning & Behaviors (cont.)



Step by step

1. Start with a large-scale behavior that solves the problem.
2. Break it down into smaller pieces. Then break the smaller pieces down as well.
3. Repeat until you have behaviors that are small enough for ROBOTC to understand.

When all the pieces have reached a level of detail that ROBOTC can work with – like the ones in the **"ROBOTC-ready behaviors"** list above – take a look at the list you've made. These behaviors, in the order and way that you've specified them, represent the plan that the robot needs to follow in order to accomplish the goal.

Because the steps are still written in English, they should be relatively easy to understand for the human programmer.

As the programmer becomes more experienced, the organization of the behaviors in English will start to include important techniques from the programming language itself, like if-else statements and loops. This hybrid language, halfway between English and the programming language, is called ***pseudocode***, and is an important tool in helping to keep larger programs understandable.

```
1. Turn on left motor
2. Turn on right motor
3. Wait 3 seconds
4. Turn off left motor
5. Turn off right motor
```

Simple pseudocode
Your list of behaviors to perform in a specific order are a simple form of pseudocode.

```
if (the light sensor sees light)
{
   turn on left motor
   hold right motor still
}
```

Later pseudocode
As your programming skills grow, your pseudocode will include more complex logic, but will still serve the same purpose: to help you find and express the necessary robot behaviors in simple English.

Thinking about Programming Planning & Behaviors (cont.)

End of Section

By starting with a very large solution behavior, and breaking it down into smaller and smaller sub-behaviors, you have a logical way to figure out what the robot needs to do in order to accomplish its task.

By recording the behaviors in English, you have taken the first steps toward good pseudocoding practice, allowing you to easily review the behaviors and their organization as you prepare to translate them to program code.

The only step remaining is to translate your behaviors from English pseudocode to ROBOTC programming language.