Topic support guide

Cambridge International AS & A Level

Cambridge International AS and A Level Computer Science

9608

For examination from 2017

Topic 4.2.3 State-transition diagrams

Cambridge Advanced

CAMBRIDGE International Examinations

Version 1

Contents

Introduction

How to use this guide

The aim of this guide is to facilitate your teaching of Cambridge International AS and A Level Computer Science topic 4.2.3 State-transition diagrams, part of topic 4.2 Algorithm design methods. The guidance and activities in this resource are designed to help teachers devise programmes of study which provide teaching time devoted to theory work as well as activities that consolidate learning.

Section 1 lists some key terms used in this topic and their definitions. Section 2 introduces state-transition diagrams and provides example diagrams. Section 3 lists some online resources that you or your learners may find useful, including software packages for creating state-transition diagrams. Section 4 gives ideas for class and homework activities.

Learning objectives

Using this document should help you guide learners in the following syllabus learning objectives:

- use state-transition diagrams to document an algorithm
- use state-transition diagrams to show the behaviour of an object

Prior knowledge

Before you begin teaching this topic you should:

- be familiar with state-transition diagrams and their conventions
- understand how to construct a state-transition diagram from the description of the behaviour of an object
- understand how to construct a state-transition diagram from a state-transition table
- understand how to construct a state-transition diagram from an algorithm

1. Key terms

Word/phrase	Meaning
accepting state	A state the system reaches when the input string is valid
event	Something that can happen within a system, such as a timer event, or an input to the system, that may trigger a transition to another state
finite state machine (FSM)	A system that consists of a fixed set of possible states with a set of allowable inputs that may change the state and a set of possible outputs
guard condition	A condition which must be met for a transition to occur from one state to another
state	The value or the position in which a system is at a given point
state transition diagram	A graphical representation of a finite state machine
state transition table	A table that shows all the states of an FSM, all possible inputs and the state resulting from each input
transition	The change from one state to another state

2. Theory and worked examples

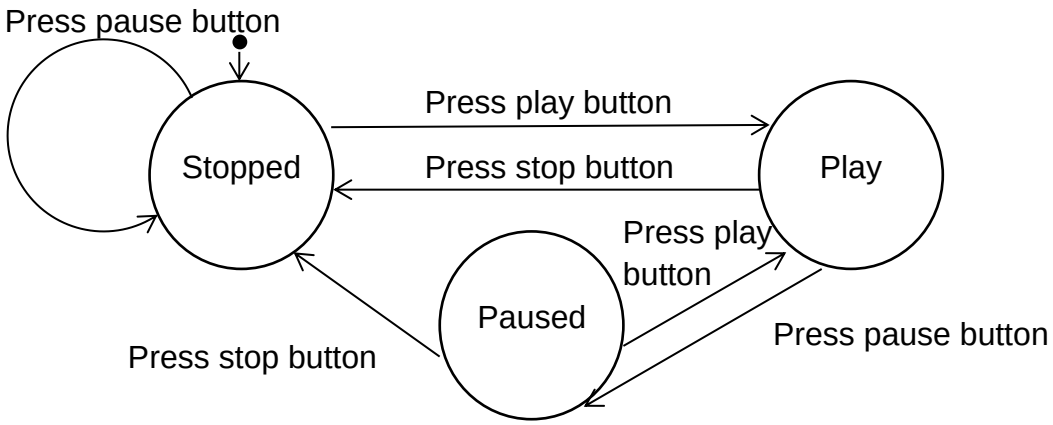
2.1 What are state-transition diagrams?

State-transition diagrams are suitable for systems that operate as finite-state machines – these are systems that have a fixed number of different states that may change on an event or input. State-transition diagrams give a visual representation of all the states that a system can have, the events such as inputs or timers that may result in transition between states, and the transitions between states. They may also show the conditions needed for an event(s) to cause a transition to occur (the guard condition), and the outputs or actions carried out as the result of a transition.

There are different conventions for state-transition diagrams, but states are normally represented as nodes, transitions as interconnecting arrows, and events as labels on the arrows. Conditions are normally specified in square brackets after the event label (see Section 4.2, homework solution 1). The initial state is indicated by an arrow with a black dot.

2.2 Worked example – media player

The example below shows a simple state-transition diagram for a media player with three buttons: stop, play and pause. The initial state of the player is stopped. In each state, only the buttons for the other states can be pressed (e.g. in play, only the stop and pause buttons can be pressed). Pressing the pause button when the player is stopped does not result in any change to the player.



The event (press pause when state is Stopped) that does not cause any change in state is indicated by the circular arrow.

A finite-state machine can also be represented by a state-transition table, which lists all the states, all possible events, and the resulting state. The following is the state-transition table for the diagram above:

Current State	Event	Next State
Stopped	Press play button	Play
Stopped	Press pause button	Stopped
Play	Press stop button	Stopped
Play	Press pause button	Paused
Paused	Press play button	Play
Paused	Press stop button	Stopped

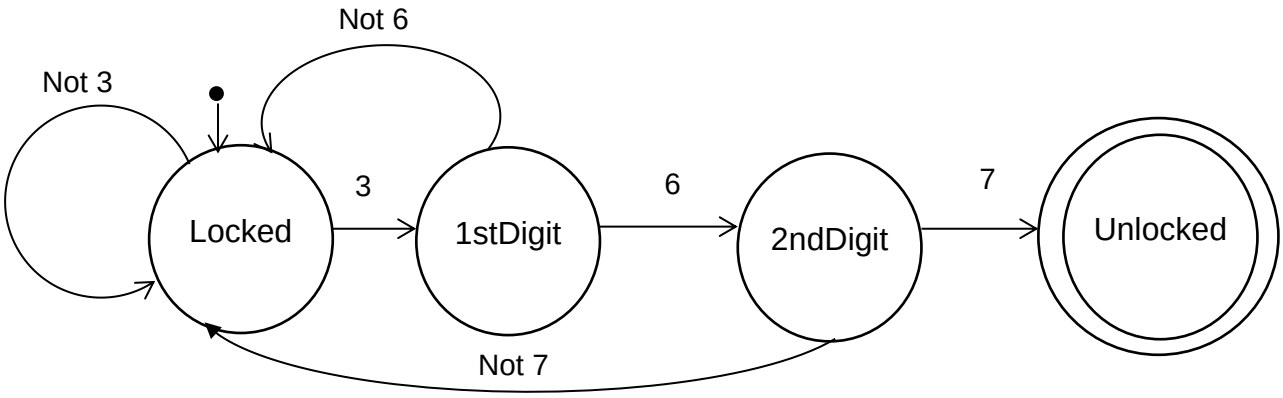
2.2 Worked example – combination lock

State-transition diagrams are also useful for showing the working of algorithms that involve a finite number of states. The following algorithm is for a three-digit combination lock where the correct combination to unlock is '367'. The initial state is Locked, each correct digit changes the state, until the combination unlocks the lock. An incorrect digit returns the lock to the original locked state.

```
DECLARE State : String
DECLARE Number : Integer

State ← Locked
INPUT Number
CASE OF Number
  3 : IF State = Locked
        THEN State ← 1stDigit
      ENDIF
  6 : IF State = 1stDigit
        THEN State ← 2ndDigit
        ELSE State ← Locked
      ENDIF
  7 : IF State = 2ndDigit
        THEN State ← Unlocked
        ELSE State ← Locked
      ENDIF
ENDCASE
```

The state-transition diagram for the algorithm is shown below:



The double line around the Unlocked state indicates that lock halts in this state – this is also known as the 'accepting state'.

3. Online resources

The following are useful resources for understanding state-transition diagrams.

The content of websites is dynamic and constantly changing. Schools are strongly advised to check each site for content and accessibility prior to using it with learners. Cambridge International Examinations is not responsible for the accuracy or content of information contained in these sites. The inclusion of a link to an external website should not be understood to be an endorsement of that website or the site's owners (or their products/services).

3.1 Websites

The first part of this tutorial is a useful introduction to the elements of state transition diagrams

http://www.sparxsystems.com/resources/uml2_tutorial/uml2_statediagram.html

Slide presentation of finite state machines and state transition diagrams for the AQA A Level syllabus, includes a simple turnstile example.

http://community.computingatschool.org.uk/resources/1357

Harel statecharts, their advantages and use in object-oriented programming

http://www.cs.unc.edu/~stotts/145/CRC/state.html

3.2 Videos

State-transition diagrams and testing. This video explains how a state-transition diagram can be used to generate the tests required to fully test all states of a system. It also gives an easy to understand overview of state-transition diagrams.

https://www.youtube.com/watch?v=iQwn_jBQ39k

Visual Paradigm software tutorial for drawing a simple state machine diagram. This tutorial goes through the principles of the bank account state-transition diagram covered in the theory notes in this resource and it recaps the main principles.

https://www.youtube.com/watch?v=UzUUZRK_Q6Y

3.3 Software

Interactive state transition programming environment. The Kara educational programming environment will allow you to create and execute state machines.

http://www.swisseduc.ch/compscience/karatojava/

A free 30-day trial of the Visual Paradigm software is available from the link below. This software will enable learners to try out the drawing of simple state-transition diagrams.

http://www.visual-paradigm.com/download/.

4. Class and homework activities

4.1 Homework questions

1. Create a state transition diagram for the following system.

   A door can be in one of the following three states:
   - Opened
   - Closed
   - Locked

   The door is initially open.
   An open door can be closed if the condition that the doorway is empty is met.
   A closed door can be opened.
   A closed door can be locked.
   A locked door can be unlocked.

2. Create a state transition diagram for the following system.

   A burglar alarm has three states:

   - Off
   - Alarmed (the alarm is switched on)
   - Sounding

The possible inputs are: turning the alarm on or off with a key, and sensors on (movement detected which sounds the alarm) or off. The state-transition table that describes the burglar alarm is a follows:

Current State	Key	Sensors	Next State
Off	Off	On	Off
Off	On	Off	Alarmed
Off	On	On	Sounding
Alarmed	Off	Off	Off
Alarmed	Off	On	Off
Alarmed	On	On	Sounding
Sounding	Off	Off	Off
Sounding	Off	On	Off
Sounding	On	Off	Alarmed

3. Draw a state-transition diagram for the following algorithm for a bank account. The account has two states, one for zero balance, and one for a positive balance. Withdrawals are not allowed if this would cause a negative balance.

```
DECLARE State : String
DECLARE Transaction : Integer
DECLARE Balance : Integer

State ← ZeroBalance
Balance = 0

INPUT Transaction
// Calculate the new balance, first saving the current balance in case the
transaction cannot be made
OldBalance ← Balance
Balance ← Balance + Transaction

IF Transaction > 0 THEN // This is a deposit
  State ← PositiveBalance
ELSE IF Transaction < 0 THEN // This is a withdrawal
  CASE OF Balance
    = 0 : State ← ZeroBalance
    > 0 : State ← PositiveBalance
    < 0 : Balance ← OldBalance  // As the balance would be negative, no
transaction can be made
```