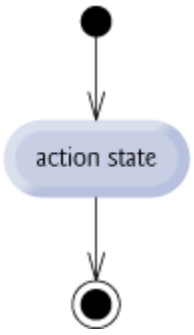
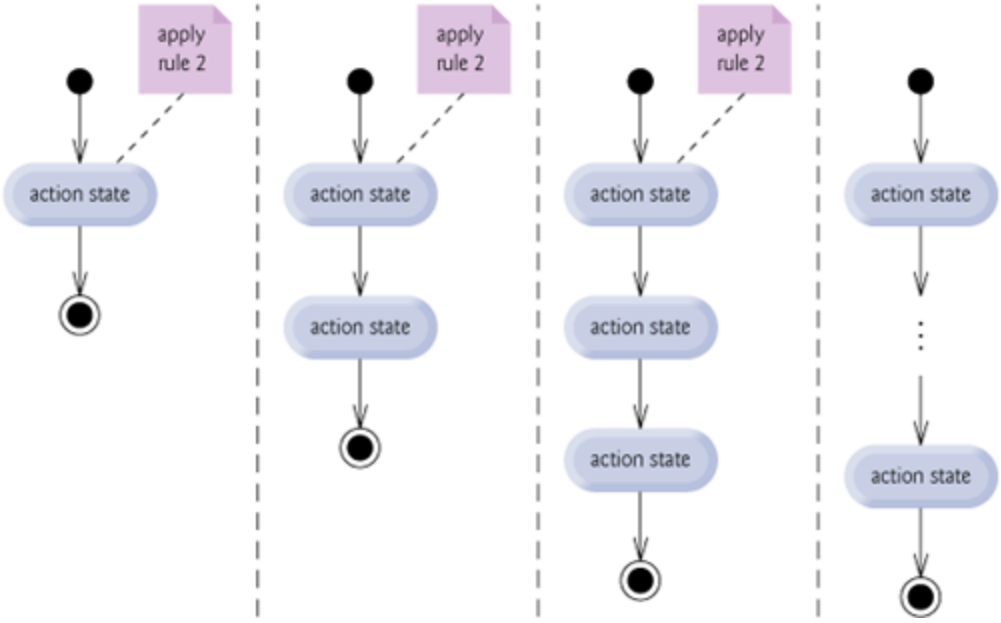
Rules for Forming Structured Programs	
1	Begin with the simplest activity diagram (Fig. 5.22).
2	Any action state can be replaced by two action states in sequence.
3	Any action state can be replaced by any control statement (sequence of action states, `if`, `if...else`, `switch`, `while`, `do...while` or `for`).
4	Rules 2 and 3 can be applied as often as you like and in any order.

Figure 5.22. Simplest activity diagram.



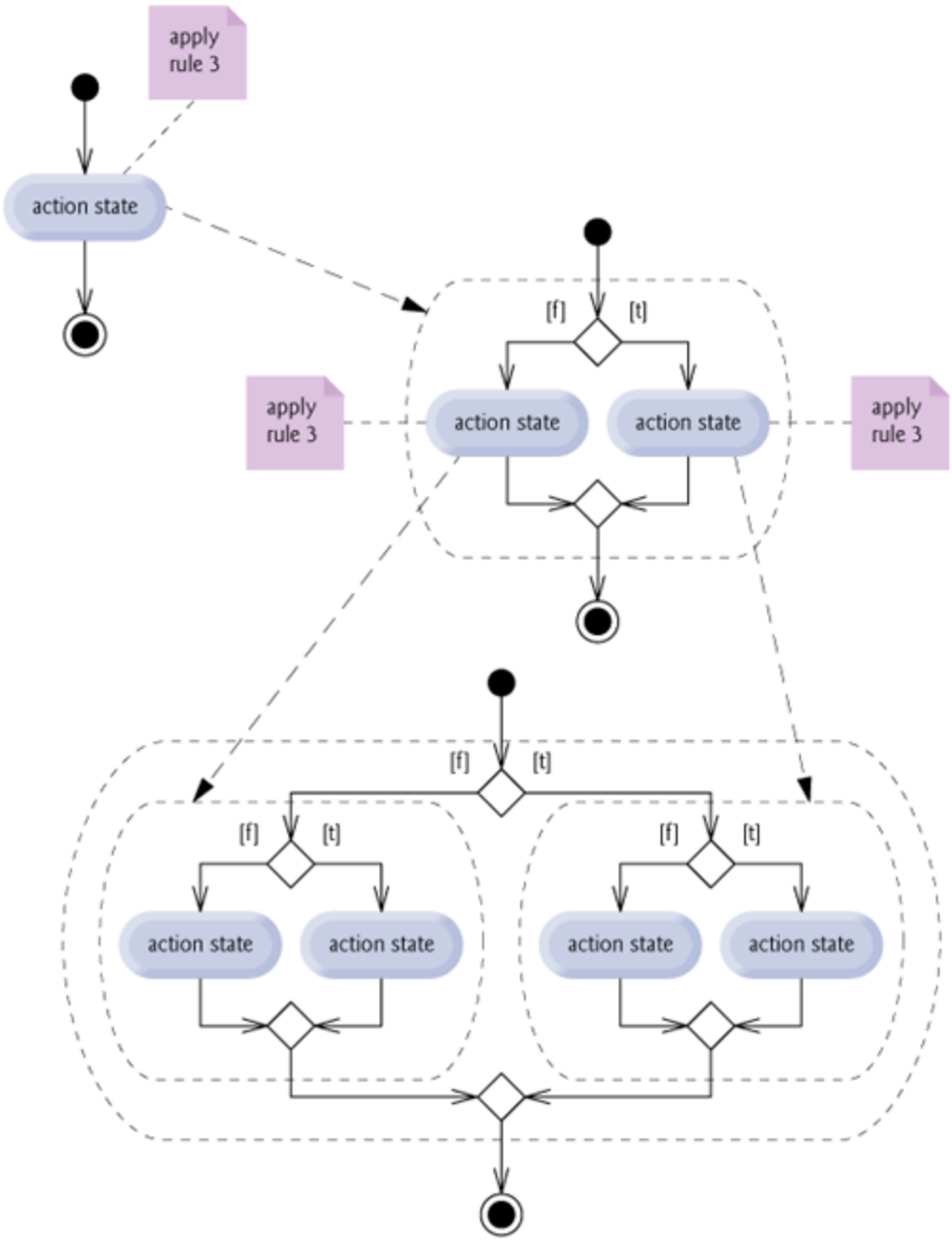
Applying the rules in Fig. 5.21 always results in a properly structured activity diagram with a neat, building-block appearance. For example, repeatedly applying rule 2 to the simplest activity diagram results in an activity diagram containing many action states in sequence (Fig. 5.23). Rule 2 generates a stack of control statements, so let us call rule 2 the **stacking rule**. [Note: The vertical dashed lines in Fig. 5.23 are not part of the UML. We use them to separate the four activity diagrams that demonstrate rule 2 of Fig. 5.21 being applied.]

Figure 5.23. Repeatedly applying the stacking rule (rule 2) of Fig. 5.21 to the simplest activity diagram.



Rule 3 is called the **nesting rule**. Repeatedly applying rule 3 to the simplest activity diagram results in an activity diagram with neatly nested control statements. For example, in Fig. 5.24, the action state in the simplest activity diagram is replaced with a double-selection (`if`...`else`) statement. Then rule 3 is applied again to the action states in the double-selection statement, replacing each with a double-selection statement. The dashed action-state symbol around each double-selection statement represents the action state that was replaced. [Note: The dashed arrows and dashed action-state symbols shown in Fig. 5.24 are not part of the UML. They are used here to illustrate that any action state can be replaced with a control statement.]

Figure 5.24. Repeatedly applying the nesting rule (rule 3) of Fig. 5.21 to the simplest activity diagram.

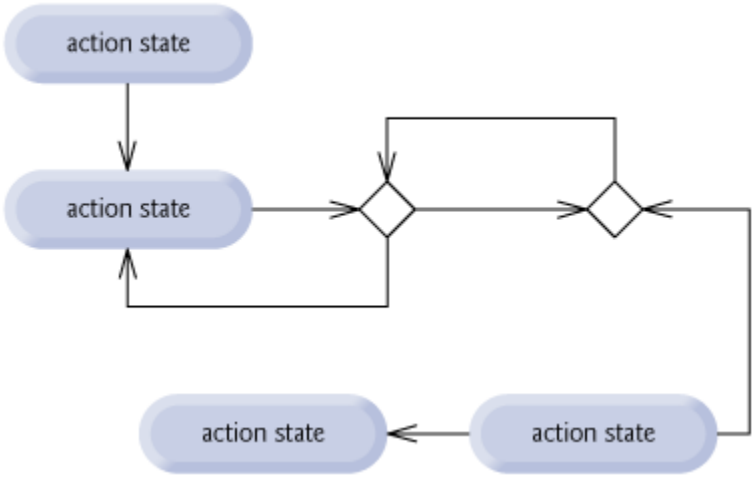


Rule 4 generates larger, more involved and more deeply nested statements. The diagrams that emerge from applying the rules in Fig. 5.21 constitute the set of all possible structured activity diagrams and hence the set of all possible structured programs. The beauty of the structured approach is that we use only seven simple single-entry/single-exit control statements and assemble them in only two simple ways.

If the rules in Fig. 5.21 are followed, an "unstructured' activity diagram (like the one in Fig. 5.25) cannot be created. If you are uncertain about whether a particular diagram is structured,

apply the rules of Fig. 5.21 in reverse to reduce the diagram to the simplest activity diagram. If you can reduce it, the original diagram is structured; otherwise, it is not.

Figure 5.25. "Unstructured" activity diagram.



Structured programming promotes simplicity. Bohm and Jacopini have given us the result that only three forms of control are needed to implement an algorithm:

- Sequence
- Selection
- Repetition

The sequence structure is trivial. Simply list the statements to execute in the order in which they should execute. Selection is implemented in one of three ways:

- `if` statement (single selection)
- `if`...`else` statement (double selection)
- `switch` statement (multiple selection)

In fact, it is straightforward to prove that the simple `if` statement is sufficient to provide any form of selection—everything that can be done with the `if`...`else` statement and the `switch` statement can be implemented by combining `if` statements (although perhaps not as clearly and efficiently).

Repetition is implemented in one of three ways:

- `while` statement
- `do`...`while` statement
- `for` statement

It is straightforward to prove that the `while` statement is sufficient to provide any form of repetition. Everything that can be done with the `do`...`while` statement and the `for` statement can be done with the `while` statement (although perhaps not as conveniently).

Combining these results illustrates that any form of control ever needed in a Java program can be expressed in terms of

- sequence
- `if` statement (selection)
- `while` statement (repetition)

and that these can be combined in only two ways—stacking and nesting. Indeed, structured programming is the essence of simplicity.