FLOW CHART LANGUAGE B

The flow chart language used in Part I is based on a small subset of the CCITT Specification and Description Language SDL, CCITT [1988], Rockstrom and Saracco [1982], SDL [1987], Saracco, Smith and Reed [1989]. There are a few deviations that bring its semantics closer to that of the PROMELA language discussed in Chapters 5, 6 and Appendix C.

Each self-contained flow chart defines a process that, at least conceptually, is executed concurrently with all other similarly defined processes. Each flow chart has one entry point that is labeled either with a process name or with the symbol *start*.

As in a traditional flow chart, the actions of a process are specified with symbols of various shapes linked by directed arcs. Six different types of symbols are used, as illustrated in Figure B.1.

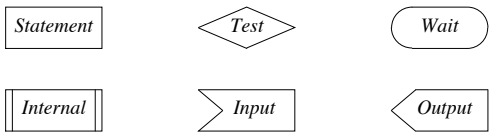

Figure B.1 — Flow Chart Symbols

These symbols represent:

- Statements, e.g, assignments
- Boolean tests, e.g., expressions
- Wait conditions, e.g., receives
- Internal events, e.g., timeouts
- Message inputs and outputs

The boolean tests are evaluated without delay. Wait conditions, however, are used to model process synchronizations. They specify that the executing process does not proceed beyond that point in the program unless a specific condition holds. The two remaining flow chart elements, used for connecting the symbols from Figure B.1, are:

- Directed arcs
- Connectors

This gives us a total of eight basic building blocks to construct charts.

The directed arcs indicating the control flow can only converge in connectors, as illustrated in Figure B.2. They can diverge, without connectors, at wait conditions and at boolean tests.

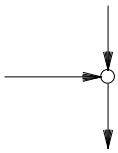

Figure B.2 — Connector and Arcs

Each flow chart process has associated with it an implicit message queue, theoretically of infinite capacity, that is used to store the incoming messages. Messages are appended to the queues in output statements and they are retrieved from the queues in input statements. Message names must uniquely identify the receiving process. Note that a message name can always be extended with the name of a process to guarantee this.

Outputs, statements, wait conditions, internal events, and boolean tests may appear anywhere in a flow chart. Inputs may only follow a wait symbol labeled *receive*. More than one input may appear.

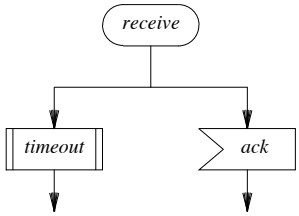

Figure B.3 — Inputs and Timeouts

A wait condition labeled *receive* will delay the executing process until the implicit message queue of that process contains, in its first slot, a message of a type specified in one of the inputs that follow the wait symbol in the flow chart. It is a protocol error if the message in the first slot of the queue is of another type.

A timeout is an internal synchronizing condition that is represented as an internal event. The corresponding condition will always eventually become true. If a timeout event is specified following a wait symbol labeled *receive*, the executing process can abort the wait for an incoming message and continue with the execution of the statements following the timeout.

The wait symbol can also be labeled with an expression. In this case the executing process will be delayed until the expression, when evaluated, yields the boolean value *true* (or any non-zero integer value).

A boolean test must be labeled with an expression, but in this case the expression is evaluated once and the resulting value is used to select an outgoing link with the corresponding label. The process is not delayed. It is an error if the evaluation of the expression yields a value for which there is no matching label on any of the outgoing arcs. The effect of such an error is undefined.

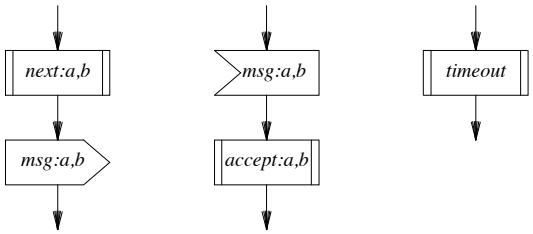

Figure B.4 — Internal Events

Two special internal actions modeling file access are predefined: *next* and *accept*. The notation *next:a,b* indicates the internal retrieval of data items *a* and *b* from an internal data base. Similarly, *accept:a,b* indicates the storage of the data items in an internal data base. The two actions *next* and *accept* include all background processing that is associated with the retrieval and storage of data items, respectively. Their usage is illustrated in Figure B.4.

The use of variables and abstract data types is not restricted by the flow chart language. Similarly, the contents of a statement box can be anything that does not involve wait conditions, receiving or sending messages, timeouts and boolean tests.

For examples, refer to the flow charts in Chapters 2 and 4.