1.3 Formalizing ADTs

If we wish to prove either

- that an algorithm that uses an ADT is correct, or
- that a data structure correctly implements an ADT,

then it is useful to have a formal specification of the correct behaviour of the ADT. Let's consider one method of describing an abstract data type (ADT) formally.

A *deterministic* ADT can be specified by stating the following 5 items.

- A set Q of states.
- A set $Q_0 \subseteq Q$ of possible initial states.
- A set OPS of possible operations (including their arguments).
- A set RES of possible responses.
- A transition function $\delta : Q \times OPS \to RES \times Q$.

If $\delta(q, op) = (res, q')$, then when the ADT is in state q and the operation op is applied to it, the operation returns the response res and the ADT moves to state q'.

This specification completely determines the set of legal histories of the ADT as follows. A finite sequence

$$q_0 \xrightarrow{op_1, r_1} q_1 \xrightarrow{op_2, r_2} q_2 \xrightarrow{op_3, r_3} \cdots \xrightarrow{op_n, r_n} q_n$$

or an infinite sequence

$$q_0 \xrightarrow{op_1, r_1} q_1 \xrightarrow{op_2, r_2} q_2 \xrightarrow{op_3, r_3} \cdots$$

is called a *legal history* if

- $q_0 \in Q_0$, and
- for all i, $\delta(q_{i-1}, op_i) = (res_i, q_i)$.

These legal histories are intended to capture all possible behaviours of the ADT (suitably initialized); i.e., what responses should be returned when the ADT is accessed by a sequence of operations.

A *non-deterministic* ADT can be specified in a similar way, except that the transition function is of the form $\delta : Q \times OPS \to \mathcal{P}(RES \times Q)$. Instead of giving a single outcome of operation *op* being applied to the ADT in state q, $\delta(q, op)$ now gives the set of all possible outcomes. Thus, $(res, q') \in \delta(q, op)$ indicates that if the ADT is in state Q and the operation *op* is applied to it, one possible outcome is that the operation gets the response *res* and the ADT moves to state q'. The definition of legal histories is similar, except the second condition changes to

- for all i, $(res_i, q_i) \in \delta(q_{i-1}, op_i)$.

We should strive for simplicity when writing a formal specification of an ADT. For example, we should not put any unnecessary structure on the states. In other words, we should try to make the set Q as small as possible. If every sequence of operations and responses starting from a state q_1 is also legal starting from state q_2 and vice versa, then q_1 and q_2 are indistinguishable, so there is no reason not to collapse the two states into a single state.

Example 1. The familiar (and deterministic) *stack* ADT can be specified as follows. Let D be a set. Items to be placed on the stack are from this domain. D might be finite or infinite.

- $Q = D^*$ (i.e., the set of all finite strings of elements of D).
- $Q_0 = \{\varepsilon\}$, where ε denotes the empty string. (Here, we are assuming that the stack is initially empty.)
- $OPS = \{\textsc{Push}(d)\} \cup \{\textsc{Pop}\}$.
- $RES = D \cup \{\textsc{Ack}, \textsc{Empty}\}$.
- In the following definition of the transition function, we use $\cdot$ to denote string concatenation.

$$\begin{aligned} \delta(s, \textsc{Push}(d) &= (\textsc{Ack}, s \cdot d), && \forall s \in Q, \forall d \in D \\ \delta(\varepsilon, \textsc{Pop}) &= (\textsc{Empty}, \varepsilon) \\ \delta(s \cdot d, \textsc{Pop}) &= (d, s), && \forall s \in Q, \forall d \in D \end{aligned}$$

Intuitively, the state represents the contents of the stack as a string, with the top of the stack represented by the right end of the string.

Example 2. A *bag* ADT stores a finite set of values and provides two operations: INSERT adds an element to the set and REMOVE removes and returns an *arbitrary* element of the set. Here, we assume the bag cannot contain duplicates. The REMOVE operation is inherently non-deterministic: making it deterministic would put unnecessary restrictions on implementations of the bag. So we model it as a non-deterministic ADT as follows.

Let D be a set. Elements stored in the set are drawn from D.

- Q is the set of all finite subsets of D.
- $Q_0 = \{\{\}\}$. (Here, we assume that the bag must be empty initially.)
- $OPS = \{\textsc{Insert}(d) : d \in D\} \cup \{\textsc{Remove}\}$.
- $RES = D \cup \{\textsc{Success},\ \textsc{Fail}\}$.
- The transition function is defined as follows.

$$\begin{array}{rcll}
\delta(S, \textsc{Insert}(d)) & = & \{(\textsc{Success}, S \cup \{d\})\}, & \forall S \in Q, \forall d \in D - S \\
\delta(S, \textsc{Insert}(d)) & = & \{(\textsc{Fail}, S)\}, & \forall S \in Q, \forall d \in S \\
\delta(\{\}, \textsc{Remove}) & = & \{(\textsc{Empty}, \{\})\} & \\
\delta(S, \textsc{Remove}) & = & \{(d, S - \{d\}) : d \in S\}, & \forall S \in Q \text{ with } S \neq \{\}
\end{array}$$