

# ABSTRACT DATA TYPES

TODD D. VANCE

## 1. LOCAL GRAPH

A local graph (modeling a directed graph, loops and multiple edges allowed, from which only a node and its immediate neighborhood are visible at any one time) is actually a system of ADTs. It consists of a single Cursor, and a finite number of Nodes and outgoing Edges indexed by Directions from a node. In addition, there is a concept of Items. An Item belongs to something, either a Node, another Item, the Cursor, or it belongs to nothing (the Nil object). The motivation for Items is Zork-like text adventure games, in which a Node is a room, the Cursor is the player, and Directions are connections to other rooms. Items can then be in a room, on the player, in or on another item, or nowhere (for a period of time).

The axioms are as follows:

- (1) There exists a unique Cursor object  $c$ .
- (2) There exists a unique Nil object  $\epsilon$ .
- (3) There exists a unique Node object  $n$  such that  $c \in n$ .
- (4) For each Direction  $d$  and Node object  $n$ , either  $n.d = \epsilon$  or there exists a unique Node object  $m$  such that  $n.d = m$ .
- (5) For each Item object  $i$ , exactly one of the following hold:
  - (a)  $i \in \epsilon$ .
  - (b)  $i \in c$ .
  - (c) There exists a unique Node object  $n$  such that  $i \in n$
  - (d) There exists a unique Item object  $j$  such that  $i \in j$
- (6) There are no circular inclusions among items; that is for no sequence of Item objects  $i_1, i_2, \dots, i_n$  does it happen that  $i_1 \in i_2 \in \dots \in i_n \in i_1$ .

For practical purposes, some query operations are permitted:

- (1) If  $n$  is a Node object, then  $\text{dir}(n)$  is a set containing all directions  $d$  such that  $n.d \neq \epsilon$ .
- (2)  $\text{loc}(c)$  is the unique Node object  $n$  such that  $n \in c$ .
- (3)  $\text{loc}(i) = o$  is equal to the object  $o$  which must be exactly one of the following:
  - (a)  $o = \epsilon$  if  $i \in \epsilon$ .
  - (b)  $o = n$  if  $n$  is a Node object and  $i \in n$ .
  - (c)  $o = j$  if  $j$  is an Item object and  $i \in j$ .
  - (d)  $o = c$  if  $i \in c$ .

Optional operations provide more functionality:

- (1) If  $n$  is a Node object, then  $\text{itm}(n)$  is the set of all Item objects  $i$  such that  $i \in n$ .
- (2) If  $i$  is an Item object, then  $\text{itm}(i)$  is the set of all Item objects  $j$  such that  $j \in i$

- (3)  $itm(c)$  is the set of Item objects  $i$  such that  $i \in c$ .
- (4)  $itm(\epsilon)$  is the set of Item objects  $i$  such that  $i \in \epsilon$ .
- (5)  $itm()$  is the set of all Item objects.
- (6)  $nod()$  is the set of all Node objects.