Data Structures in SDL

SDL is designed to be generated by tools automatically. SDL is a control mechanism for any object that can be presented by a program. With SDL, total controllability can be achieved. SDL has been used for generating complicated stimulus, error injections, simulation controls, etc. It is virtually unlimited to what kinds of data structures that can be created in SDL.

1.1 Constant

This is the simplest data type in SDL. A property value that is a constant can represent a parameter, an on/off switch, and a multi-value knob.

1.2 String

String is actually the most generic data type of property values. It is used to represent a file name, a keyword (on, off, ++, --, etc.), an instance path, and a new data type, etc.

1.3 Random Constant

A random constant is generated from the random generator once and stays unchanged during the simulation. Random constants have the following forms,

1. **random:** any possible integer value
2. **[min, max]:** an integer between min and max.

1.4 SDL integer

An SDL integer is a combination of a constant and a random constant.

1.5 Random Variable

A random variable has the same form as a random constant. The difference is a random variable keeps changing its value by calling the random generator every time before the use. It is the underline layer of software to decide which one is proper. Most of the time, a random constant is sufficient.

1.6 List Variable

A list variable has the form like [3,5,7,9]. For example, a logical object consists of several physical objects. The logical object can have a property assignment that uses a list variable to define its physical object ids.

1.7 Linked List

To represent a linked list, two or more properties have to group together. For example, if we like to represent a list of memory addresses, the following properties can be defined,

```
startingAddress = 0x1000
nextOffsetAddress = [16, 32]      // This is a random variable
```

In this case, an address list starting from 0x1000 is formed. The next address value is between 0x1010 to 0x1020 in random, and so on. This is the simplest form of a linked list. Another property can be added to define the size of the memory block or the number of list elements to limit the length of the linked list.
Another example is the TLP traffic generator in PCIE-VR.

1.8 Array

The following property assignment defines an array with four elements.

```
numOfElements = 4
element(0) = [1,5]
element(1) = 6
element(2) = 3
element(3) = random
```

One usage of array data type is error injection. You can precisely control the number of errors to inject and when to inject them.

1.9 Tree

Grouping several objects and property assignments together can represent a tree data type. For example,

```
%soc
// define the port count
numOfPorts = 2

%port(0)
numOfNodes = 3    // port(0) has three nodes

%port(1)
numOfNodes = 1    // port(1) has one node

%node(0,0)        // node(0,0) is the first node that connects to port(0)
%node(0,1)        // node(0,1) is the second node that connects to port(0)
%node(0,2)        // node(0,2) is the third node that connects to port(0)
```

```
%node(1,0)          // node(1,0) is the only node that connects to port(1)
```

The above objects and properties represent the tree shown in Figure 0.1.

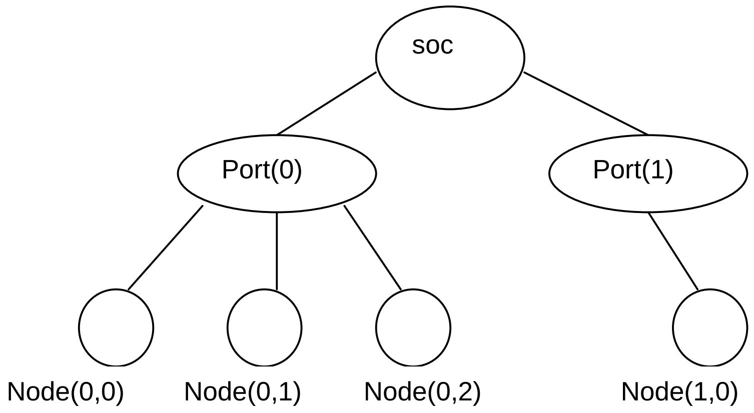

Figure 0.1 A Tree Data Structure Example

The tree data type is used extensively in the networking architecture simulator [1] to represent any kind of network topology that is connected to a networking ASIC. This approach is also applicable to represent USB and PCI Express connectivity.

1.10 Function

A property value can be an expression of other properties. The following example is from the Automatic Driver Generator (ADG).

Memory Configuration Example

A table is a logical entity that can be stored in multiple physical memories. For example, a logical memory is stored in four physical memories: phyMemA, phyMemB, phyMemC, and phyMemD. Base addresses may be different for each memory that can be expressed in an equation form.

row	offset	127 64	63 0
0	0	baseAddr=0x0000	baseAddr=0x1000
1	1		
2	2		
…	…	***phyMemA***	***phyMemB***
1023	1023		
1024	0	baseAddr=0x2000	baseAddr=0x3000
1025	1		
1026	2		
…	…	***phyMemC***	***phyMemD***
2047	1023		

```
path    = ((row < 1024) ?
            ((bit < 64) ? "phyMemA": "phyMemB"):
            ((bit < 64) ? "phyMemC": "phyMemD"))

address = ((row < 1024) ? ((bit < 64) ? 0x0000: 0x1000):
                          ((bit < 64) ? 0x2000: 0x3000))

offset  = ((row < 1024) ? row: (row-1024))
```

1.11 Object Embedding

Objects, such as Dynamic Simulation Control (DSC), can be embedded in SDL. DSC is a scripting tool. This gives the user the power of writing portable script in SDL.
Commonly used scripts, such as Perl or Tcl, can also be embedded.