COMS W4115

Project Proposal

Vence Stanev (UIN: vs2226)

Turn based simulation language TBSL

Abstract

The turn based simulation language (TBSL) is a language that enables programmers to describe a current state of a system comprised of objects and relations. The goal of TBSL is to run that simulation for a number of turns in order to examine the effects of particular phenomena on the system.

Applications

Among other things, TBSL can be used to describe a group of people as well as how they are connected and attempt to predict infection rates among the group given a particular type of infectious disease.

Identifiers

An identifier is a sequence of letters and digits only. Upper and lower case letters are considered different. Identifiers should not be longer than 32 characters.

Comments

The characters /* introduce a comment, which terminates with the first occurrence of the characters */.

Keywords

Keywords are identifiers that are reserved words in TBSL. They have specific function and cannot be used as regular identifiers.

Init – initialize an object

Relation - define a relation

Func – define a function

List – define a list of “Objects”

Call – execute an operator

Next – makes the simulation go to the next turn

Basic types

TBSL has only one basic type, which is called “Object”. No notion of type conversion is defined. When declaring a variable, type is not specified but the variable needs to be initialized. A variable is initialized by providing a list of attributes, which is a list of tuples, each tuple being a name\value pair. The name is always a string and the value can be an int, float or string.

Syntax example:

```
Init a ((“status”,”active”), (“cost”, 5.7), (“ValueAddPerTurn”,10));
```

TBSL also supports lists of “Objects”.

Syntax example:

```
List ObjList;  List ObjList.Append(a); List ObjList.Prepend(a); List ObjList.Remove(a);
```

Operators

Operators in TBSL work on a list of Objects. Some operators may require that the objects have the same set of attributes while others will not.

Syntax example:

```
Init a ((“status”,”active”), (“cost”, 5.7), (“ValueAddPerTurn”,10));

Init b ((“status”,”inactive”), (“cost”, 4.0), (“ValueAddPerTurn”,12));

Call Compare (a,b);

Call IncrementCost (a);
```

Relations

Relations are a special kind of operators. They represent a connection between the object in the input list and in effect create an undirected graph.

Syntax example:

```
Init a ((“status”,”active”), (“cost”, 5.7), (“ValueAddPerTurn”,10));

Init b ((“status”,”inactive”), (“cost”, 4.0), (“ValueAddPerTurn”,12));
```

Relation NextTo (a,b);

Functions

TBSL also supports functions in order to promote modularity. Functions have only one parameter which is a list of objects.

Syntax example:

```
Func MyFunciton (ListOfObjects)
{
Init a ((“status”,”active”), (“cost”, 5.7), (“ValueAddPerTurn”,10));
Init b ((“status”,”inactive”), (“cost”, 4.0), (“ValueAddPerTurn”,12));
Relation NextTo (a,b);
ListOfObjects.Append(a);
ListOfObjects.Append(b);
}
```