%analog

A definitive notation for modelling electronic analogue computation

Charlie Care

Novemember 2004

`%analog` is a notation written for tkeden. The notation comprises of two main layers. Firstly there is the notation language itself, this was created using the agent oriented parser and is designed to provide a high-level notation for describing the relationship between the various components available on an electronic analogue computer. Secondly, there is the is the *engine* that maintains the state of the computer. When `%analog` definitions are parsed, they are translated into eden definitions. This second layer consists of a number of procedures that work alongside the translated `%analog` to simulate the activity of the computer.

There is also a third layer which provides a basic grapical interface to controlling the computation. This allows `%analog` to be used for standalone simulation of an analogue computer. However, it is intended that the notation should provide a framework on which more elaborate models can be constructed and therefore this layer is not essential to the workings of the underlying notation.

1 Analogue Computers and %analog

An electronic analogue computer is essentially a large set of computing components. Most of these are based on the operational amplifyer and can be configured as *summers* or *integrators*. Also available are various potentiometers (variable resistors) which allow variables to be scaled. As well as these components there are often function generator and multiplyers, but since these are not provided in the `%analog` notation we will not go into them in great depth. For the purposes of the notation, summers, integrators and potentiometers are available as primitive components. The `%analog` definitions for these components are given in Table 2 along with their graphical and mathematical representations.

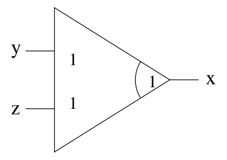
Note that the graphical representations of the summer and integrator in 2 have a the numeral 1 next to each input and output. On an analogue computer provision is made for inputs and outputs to have an amplification factor of 1 (no amplification) or 10. This amplification is represented on the diagram by placing either a 1 or a 10 next to the relevant connection. In `%analog` none of the connections have amplification, to represent amplified inputs and outputs you use an `amplify()` definition. An example is shown in Figure 1.

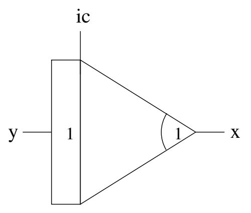
Sometimes, analogue integrators are shown in analogue computing diagrams with a number of inputs. Such a component is essentially a summer and an integrator in one package and is therefore reperesented in `%analog` as a summer connected to an integrator (see Figure 1).

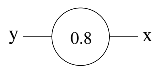
2 How to represent a differential equation

In my copy of [?] appears this example representation of a differential equation on an analogue computer.

Component	Semantics	Graphical Representation

`%analog`
`x=summer(y,z);`
	$x = y + z$	

`%analog`
`x=integrator(ic,y);`
	$x = \int y dt + ic$	

`%analog`
`x=scale(0.8,y);`
	$x = \frac{8}{10} y$	

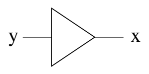
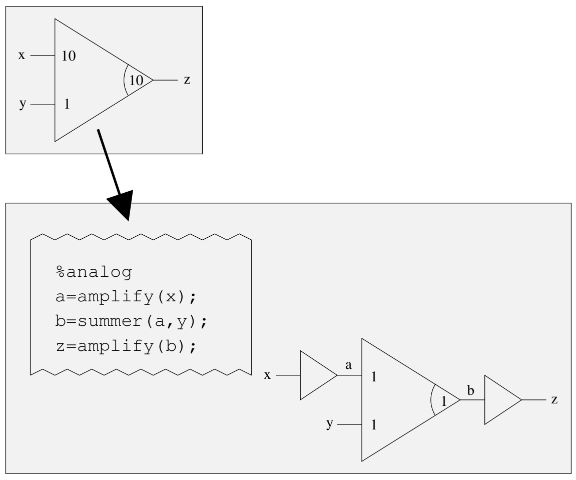
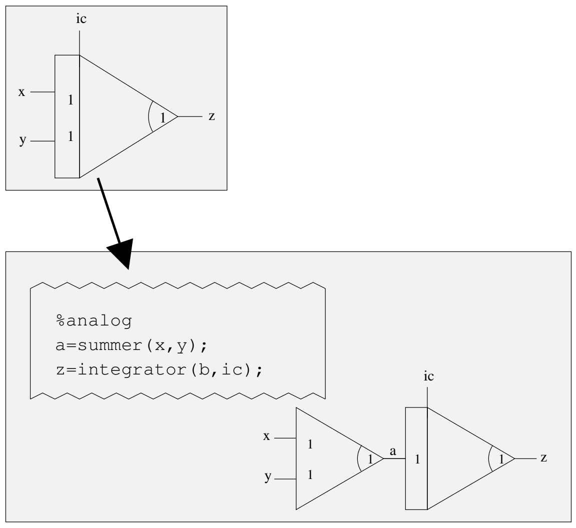
`%analog`
`x=amplify(y);`
	$x = 10y$	

Table 1: The basic `%analog` definitions



```
%analog
a=amplify(x);
b=summer(a,y);
z=amplify(b);
```

Figure 1: How to represent amplified inputs/outputs in `%analog`



```
%analog
a=summer(x,y);
z=integrator(b,ic);
```

Figure 2: How to reresent a integrator with summing inputs in `%analog`