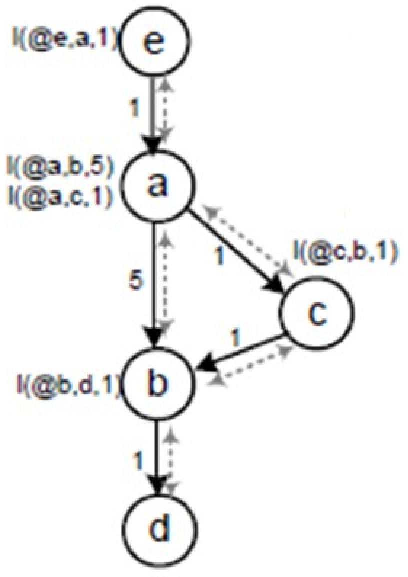
Hi Sumanth and Drew,

In this example, we simulate the calculating shortest paths from each node to all other possible nodes. Since we do not have the real network (we can have it when we deploy the testbed if funded), so the execution is a simulation, more specifically, a discrete, pseudo-synchronized simulation of running the shortest path algorithm (protocol) on all nodes.

At the very beginning, we initialize the execution environment, with following code

```
from pyDatalog import pyDatalog
pyDatalog.create_terms('Src, Dest, Cost, Path, link, f_init, path')
```

There are two sets of input. One is the directed graph as shown in Figure 1 left side and below. Another input is the rule set, i.e. sp1 – sp4.



As shown in the Figure 1 of the paper "Declarative Networking" paper, at the very beginning, every node has as input the links that goes out of this nodes. So at node e, it is the link tuple l (@e, a, 1), or mathematically <e, a, 1>. In Python and pyDatalog, simply write it as

t (e, a, 1)

There two ways to interpret sp1. The difference lays in how we treat link predicate/relation.

(1) Treat a link predicate/relation as a fact, or a literal. This way, we can feed input at every node if it is really a distributed environment. In simulated distribution, this can be done with single reading. Similar to examples [42-44] in the pyDatalog tutorial, it can be done with following statements,

```
    + link(e, a, 1)
    + link(a, b, 5)
    + link(a, c, 1)
    + link(e, d, 1)
    + link(c, b, 1)
```

In the real distributed environment, another slightly different way would be let each node process only the link(s) that goes out from this node. In this variety, the node E processes

```
    + link(e, a, 1)
```

The node A processes

```
    + link(a, b, 5)
    + link(a, c, 1)
```

The node B processes

```
    + link(b, d, 1)
```

And node C processes

```
    + link(c, b, 1)
```

With this approach, link has to be rewritten into a function to be able to utilize the facts above. For example, at Node e

```
    Path (Src, Dest) <= link (Src, Dest, Cost)
    Path (Src, Dest, Path, Cost) <= link (Src, Dest, Cost) & path (Src, Dest)
```

(2) Treat link as a predicate. This approach can only applies to the distributed environment. At node E, the following code should be executed

```
    Src = e
    Dst = a
    Cost = 1
```

Below is code of sp1 on Node e,

```
    Path (Src, Dest) <= link (Src, Dest, Cost)
    Path (Src, Dest, Path, Cost) <= link(Src, Dest, Cost) & path (Src, Dest)
```