FORMULA & P Language

Xinran (Eva) Liu

Mentor: Dr. Daniel Balasubramanian

ISIS
Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu

VANDERBILT UNIVERSITY

FORMULA & P Language

- ## What is FORMULA?

FORMULA is a language and framework for specifying domain-specific languages (DSLs) and checking properties of those specifications. It aims to provide efficient reasoning and compilation of input programs, as well as symbolic model generation.

- ## How it works

FORMULA specifications are written as strongly-typed open-world logic programs. They are highly declarative and can express rich synthesis and verification problems. Model generation is enabled by efficient symbolic execution of logic programs into constraints.

- ## What is P?

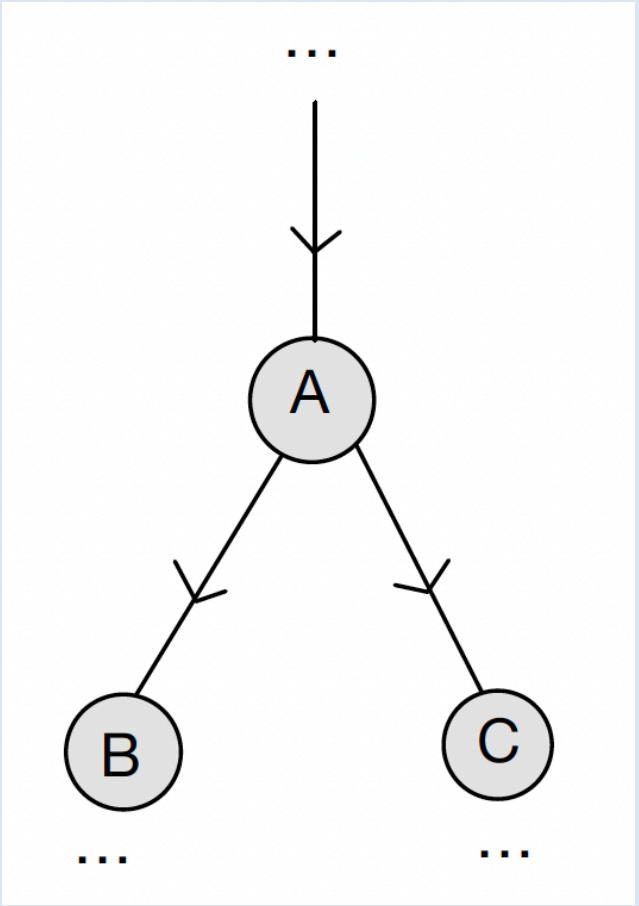
P is a domain-specific language for modeling, specifying, and testing distributed systems. The original P compiler was implemented using FORMULA.

- ## What I did

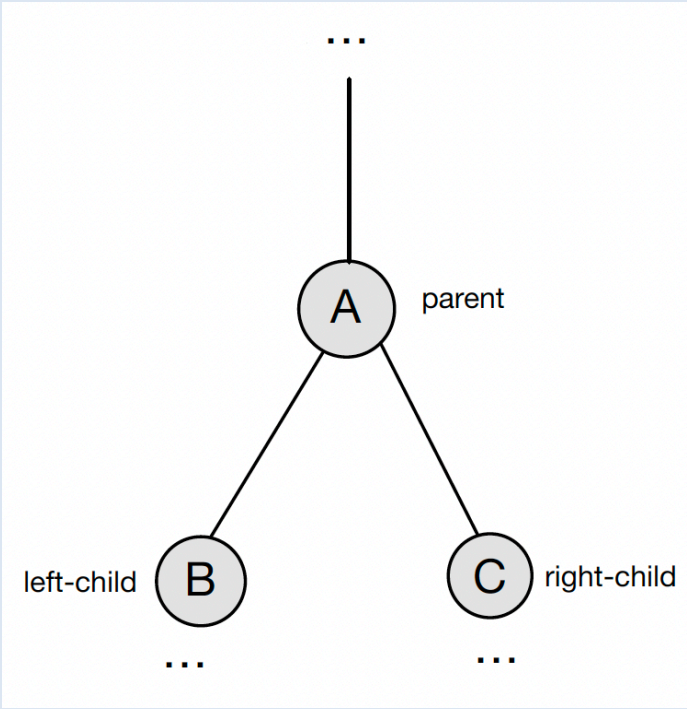
This summer I expanded the set of example FORMULA programs, created tutorials, tested new features, and developed a wiki page with documentation.
I also learned about the domain-specific language P and helped identify changes to the P language that were made since the initial P compiler was migrated from FORMULA to a standalone compiler.

Example FORMULA program: Data Graphs

```
domain Digraphs
{
 V : : = fun (lbl: Integer -> dat: Integer).
 E : : = new (src: V, dst: V).
}
```




```
domain DAGs extends Digraphs
{
 path : : = (V, V).
 path(u, w) : - E(u, w); E(u, v), path(v, w).
 conforms no path(u, u).
}
```

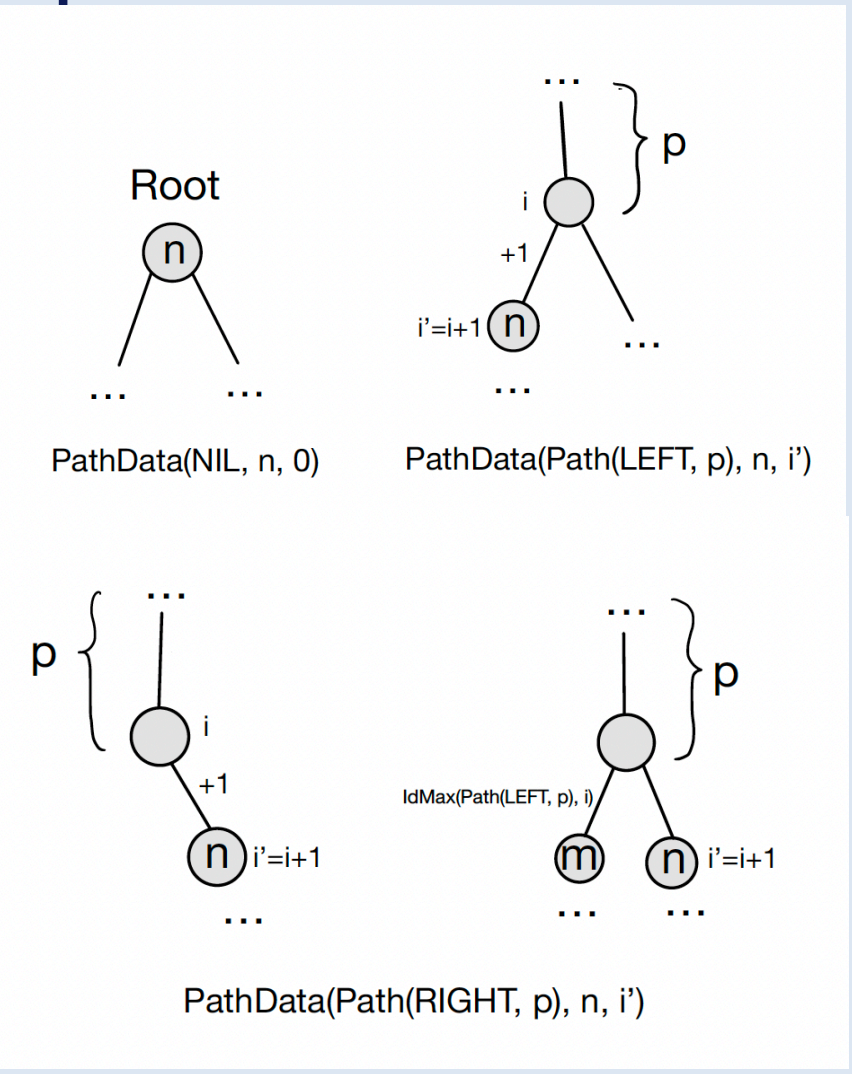



```
domain Trees extends DAGs
{
 conforms no { w | E(u, w), E(v, w), u != v }.
}
```

```
domain AlgTrees
{
 Node : : = new (dat: Integer, left: any Node + { NIL }, right: any Node + { NIL }).
 Tree : : = new (any Node).
 conforms count({ t | t is Tree}) <= 1.
}
```

Example FORMULA program: Data Gra

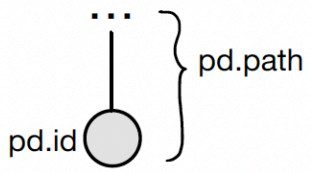
```
transform AlgToRelTree (tree:: AlgTrees) returns (graph:: RelTrees)
{
  Path     ::= ({ LEFT, RIGHT }, Path + { NIL }).
  PathData ::= (path: Path + { NIL }, node: Node, id: Integer).
  IdMax    ::= (path: Path + { NIL }, id: Integer).

  PathData(NIL, n, 0) :- Tree(n).
  PathData(Path(LEFT, p),  n, i')  :- PathData(p, Node(_, n, _), i), i' = i + 1, n : Node.
  PathData(Path(RIGHT, p), n, i')  :- PathData(p, Node(_, NIL, n), i), i' = i + 1, n : Node;
                                      PathData(p, Node(_, m, n), _), IdMax(Path(LEFT, p), i), i' = i + 1, n : Node.

  IdMax(pd.path, pd.id) :- pd is PathData, pd.node = Node(_, NIL, NIL).
  IdMax(pd.path, i)     :- pd is PathData, pd.node = Node(_, n, NIL), IdMax(Path(LEFT, pd.path), i);
                           pd is PathData, pd.node = Node(_, _, n), IdMax(Path(RIGHT, pd.path), i).

  V(id, n.dat) :- PathData(_, n, id).

  E(V(pid, prnt.dat), V(cid, chld.dat)) :- PathData(p, chld,  cid), PathData(p', prnt, pid), p = Path(_, p').
}
```



Example FORMULA program: Data Graphs

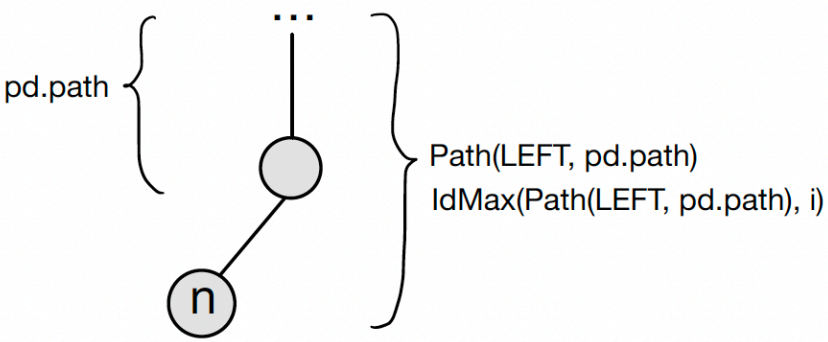
IdMax:

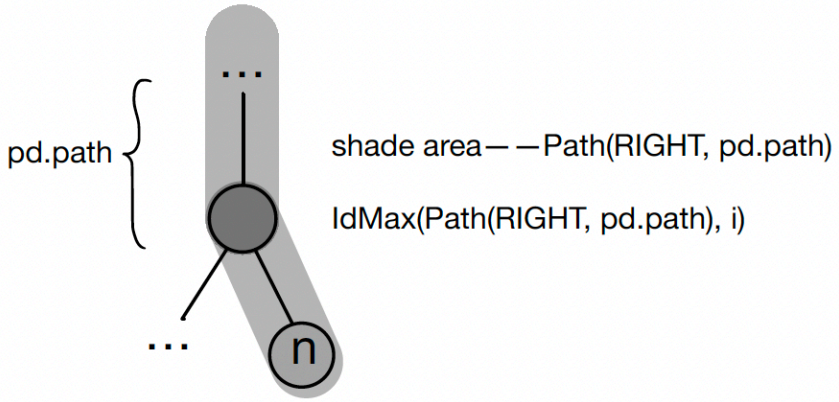
case 1: IdMax(pd.path, pd.id)




```
IdMax(pd.path, pd.id) :- pd is PathData, pd.node = Node(_, NIL, NIL).

IdMax(pd.path, i)  :- pd is PathData, pd.node = Node(_, n, NIL),
 IdMax(Path(LEFT, pd.path), i); pd is PathData, pd.node = Node(_, _,
```

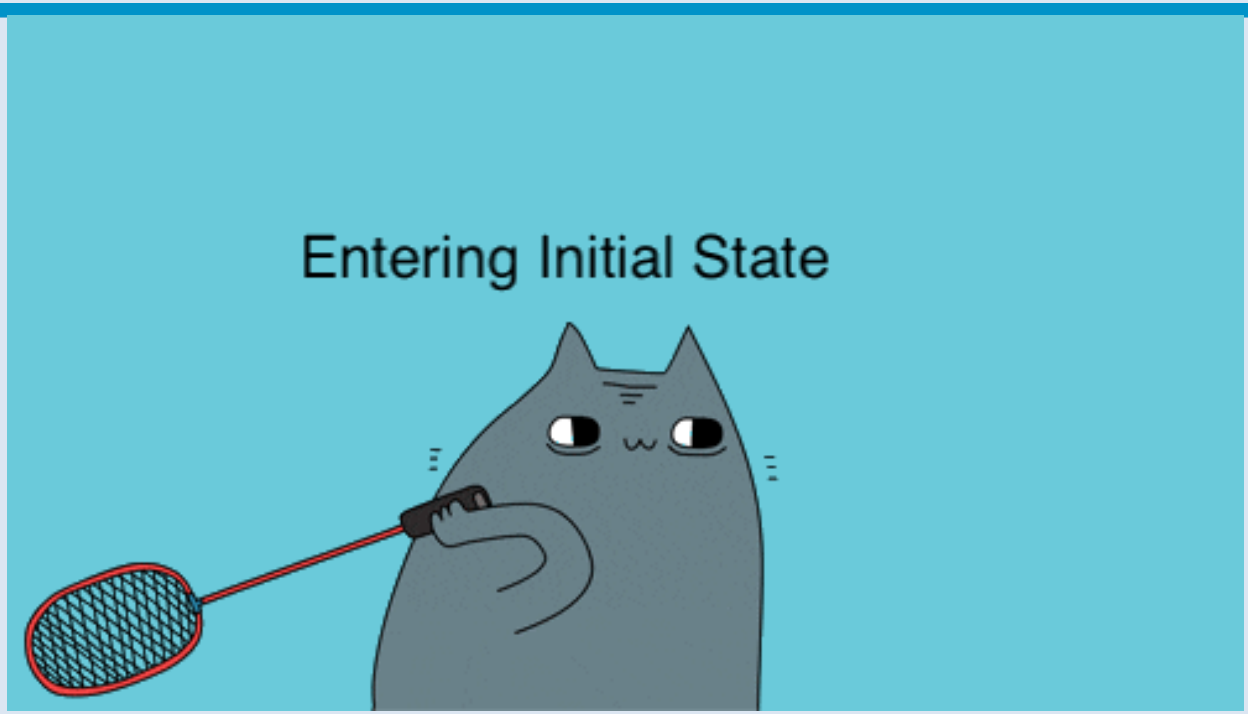
case 2: IdMax(pd.path, i)





P Language: Modular and Safe Programming for Distributed Systems

```
// PingPong.p
event PING assert 1: machine;
event PONG assert 1;
event SUCCESS;

main machine Client {
  var server: machine;

  start state Init {
    entry {
      server = new Server();
      raise SUCCESS;
    }
    on SUCCESS goto SendPing;
  }

  state SendPing {
    entry {
      send server, PING, this;
      raise SUCCESS;
  }
    on SUCCESS goto
WaitPong;
  }

  state WaitPong {
    on PONG goto SendPing;
  }
}
```




```
machine Server {
  start state WaitPing {
    on PING goto SendPong;
  }

  state SendPong {
    entry (payload: machine) {
      send payload, PONG;
      raise SUCCESS;

    }
    on SUCCESS goto
WaitPing;
  }
}
```