GroMEt2SMTLib Encoding Examples

Dan Bryce
dbryce@sift.net
SIFT, LLC.

September 2, 2022

1 Direct Encoding

$$F.fn.x = F.fn.pof[i] \quad :\forall i \in range(|F.fn.pof|), x = F.fn.pof[i].name \tag{1}$$

$$F.fn.pof[j] = F.fn.bf[i].value.fn.opo[j'] \quad :\forall i \in range(|F.fn.bf|), \forall j \in range(|F.fn.pof|) \text{ where } F.fn.pof[j].box - 1 = i, j' \in range(F.fn.bf[i].value.fn.opo), \tag{2}$$

$$F.fn.opo[i] = F.fn.pof[j] \quad :\forall k \in range(F.fn.wfopo) \text{ where } i = F.fn.wfopo[k].src - 1, j = F.fn.wfopo[k].tgt - 1 \tag{3}$$

$$F.fn.pif[i] = F.fn.pof[j] \quad :\forall k \in range(F.fn.wff) \text{ where } i = F.fn.wfopo[k].src - 1, j = F.fn.wfopo[k].tgt - 1 \tag{4}$$

(5)

1.1 Examples

1.1.1 exp0

Listing 1: Python code for example exp0.

```
1 x = 2
```

Listing 2: GroMEt for example exp0.

```
1 {
2     "name": "exp0",
3     "fn": {
```

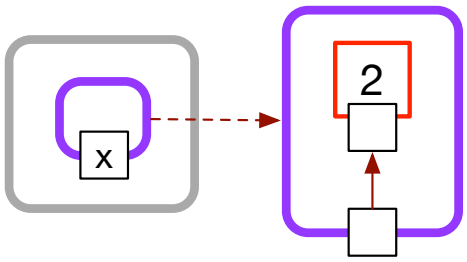

Figure 1: GroMEtFunction network for exp0.

```
4           "b": [{"function_type":"MODULE","name":"module"
                }],
5           "bf": [
6               {"function_type":"EXPRESSION","contents":1,"
                    name":""}
7           ],
8           "pof": [{"name":"x","box":1}]
9       },
10      "attributes": [
11          {
12              "type": "FN",
13              "value": {
14                  "b": [{"function_type":"EXPRESSION","name
                        ":""}],
15                  "opo": [{"name":"","box":1}],
16                  "bf": [
17                      {"function_type":"LITERAL","value":{"
                            value_type":"Integer","value":2},"
                            name":""}
18                  ],
19                  "pof": [{"name":"","box":1}],
20                  "wfopo": [{"src":1,"tgt":1}]
21              }
22          }
23      ],
24      "metadata": [
25
26      ]
27  }
```

The following equations encode the semantics of the GroMEt in Listing **??**.

$$
\begin{aligned}
exp0.fn.x &= exp0.fn.pof[0] \\
exp0.fn.pof[0] &= exp0.fn.bf[0].value.fn.opo[0] \\
exp0.fn.bf[0].value.fn.opo[0] &= exp0.fn.bf[0].value.fn.pof[0] \\
exp0.fn.bf[0].value.fn.pof[0] &= exp0.fn.bf[0].value.fn.bf[0].value \\
exp0.fn.bf[0].value.fn.bf[0].value &= 2
\end{aligned}
$$

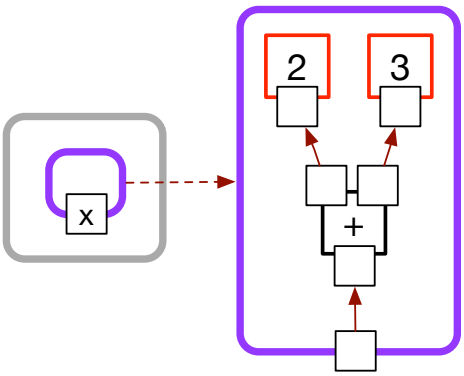

Figure 2: GroMEt Function network for exp1.

1.1.2 exp1

Listing 3: Python code for example exp1.

```
x = 2 + 3
```

Listing 4: GroMEt for example exp1.

```
{
    "name": "exp1",
    "fn": {
        "b": [{"function_type":"MODULE","name":"module"
            }],
        "bf": [
            {"function_type":"EXPRESSION","contents":1,"
                name":""}
        ],
        "pof": [{"name":"x","box":1}]
    },
    "attributes": [
        {
            "type": "FN",
            "value": {
                "b": [{"function_type":"EXPRESSION","name
                    ":""}],
                "opo": [{"name":"","box":1}],
                "bf": [
                    {"function_type":"LITERAL","value":{"
                        value_type":"Integer","value":2},"
                        name":""},
                    {"function_type":"LITERAL","value":{"
```

```
                                value_type":"Integer","value":3},"
                                name":""},
19                          {"function_type":"PRIMITIVE","name":"
                                +"}
20                      ],
21                      "pif": [{"name":"","box":2},{"name":"","
                            box":2}],
22                      "pof": [{"name":"","box":1},{"name":"","
                            box":2},{"name":"","box":2}],
23                      "wff": [{"src":1,"tgt":1},{"src":2,"tgt"
                            :2}],
24                      "wfopo": [{"src":1,"tgt":3}]
25                  }
26              }
27      ],
28      "metadata": [
29
30      ]
31  }
```

$$
\begin{aligned}
exp1.fn.x &= exp1.fn.pof[0] \\
exp1.fn.pof[0] &= exp1.fn.bf[0].value.fn.opo[0] \\
exp1.fn.bf[0].value.fn.opo[0] &= exp1.fn.bf[0].value.fn.pof[2] \\
exp1.fn.bf[0].value.fn.pif[0] &= exp1.fn.bf[0].value.fn.pof[0] \\
exp1.fn.bf[0].value.fn.pif[1] &= exp1.fn.bf[0].value.fn.pof[1] \\
exp1.fn.bf[0].value.fn.pof[2] &= exp1.fn.bf[0].value.fn.pif[0] + exp1.fn.bf[0].value.fn.pif[1] \\
exp1.fn.bf[0].value.fn.pof[0] &= exp1.fn.bf[0].value.fn.bf[0].value \\
exp1.fn.bf[0].value.fn.bf[0].value &= 2 \\
exp1.fn.bf[0].value.fn.pof[1] &= exp1.fn.bf[0].value.fn.bf[1].value \\
exp1.fn.bf[0].value.fn.bf[1].value &= 3
\end{aligned}
$$

2 Symbolic Execution Encoding

2.1 Examples

2.1.1 exp0

- Recurse the expo0 tree to identify the leaf node: $exp0.attr[0].bf[0].value = (2, \phi_0)$, where 2 is the concrete value and ϕ_0 is the symbolic value.
- Pass the box function value to its output port: $exp0.attr[0].pof[0] = exp0.attr[0].bf[0].value = (2, \phi_0)$.

- Pass the output port value to the outer output port: $exp0.attr[0].opo[0] = exp0.attr[0].pof[0] = (2, \phi_0)$
- Pass the outer output port to the containing box function output port: $exp0.pof[0] = exp0.attr[0].opo[0] = (2, \phi_0)$
- Associate the name of the output port with the value of the output port: $exp0.x = exp0.pof[0] = (2, \phi_0)$

2.1.2 Conditional

3 Parameter Synthesis

Let ps(b1, b2, b3) = [([0, 0.5], [0.1, 0.2], [0.8, 0.95]), ([0.51, 0.52], [0.0, 0.09], [0.8, 0.95])] Let ps(b) = [([0.3, 0.51]), ([0.6, 0.9])]

A parameter space is represented (here) as a list of hypercubes, where a hypercube is a tuple of intervals (one for each dimension).

If I take the project onto b1, I get $\text{ps}_b2, b3(b1) = [([0, 0.5]), ([0.51, 0.52])]$

Intersecting this with ps(b) I get $\text{Intersect}(\text{ps}_b2, b3(b1), ps(b)) = [([0.5, 0.51])]$

This is saying that there exists a value of b2 and a value of b3 where for all values of b1 in the intersection (above) the models agree when b1 = b. This is one of many ways of comparing the parameter spaces, but is probably the most simple.

Another way to do it is to define a ps(b') that intersects ps(b1, b2, b3) with a special parameter space ps=(b1, b2, b3) that is defined as the points (degenerate hypercubes) where b1=b2=b3. For example, Intersect(ps(b1, b2, b3), ps=(b1, b2, b3)) = [] because there are no hypercubes in ps(b1, b2, b3) where b1=b2=b3. If we had some ps = [([0.0, 0.3], [0.2, 0.4])], then intersecting it with a similar 2-D parameter space where the parameters are equal would result in a range [0.2, 0.3] (there are infinitely many point hypercubes, so I represent it as a range).

4 Model Comparison

Coming soon ...