Progess Report 7

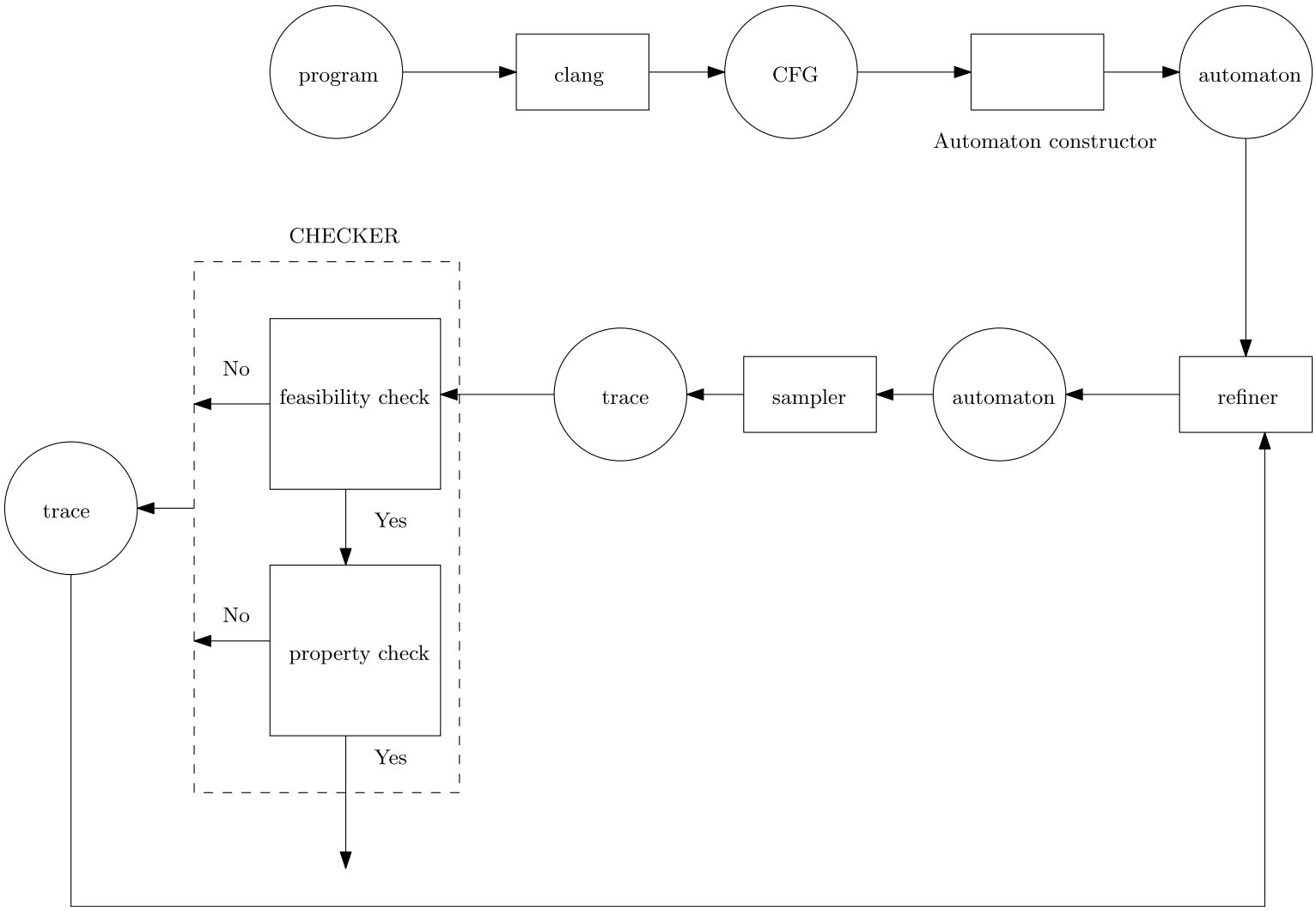
Xie Li

October 20, 2020

Overview of the Progress

- Continue the development with Weizhi.
- Currently implemented the checking of **path feasibility**, LTLf: $\mathbf{G}p$ and LTLf: $\mathbf{F}p$.

Current Core Algorithm



Test case

```
int main()
{
    int x, y;
    while(1)
    {
        x = 1;
        x = 6;
        y = 1;

        while(x > 0)
        {
            x--;
            if ( x <= 1)
                y = 0;
        }
    }
    return 0;
}
```

Path Feasibility

```
states: 0 z3:expr: true
states: 1 z3:expr: true
states: 2 z3:expr: true
states: 3 z3:expr: (= retval0 0)
states: 4 z3:expr: true
states: 5 z3:expr: (= x0 1)
states: 6 z3:expr: (= x1 6)
states: 7 z3:expr: (= y0 1)
states: 8 z3:expr: true
states: 9 z3:expr: (= tmp0 x1)
states: 10 z3:expr: (= cmp0 (> tmp0 0))
states: 11 z3:expr: cmp0
states: 12 z3:expr: (= tmp10 x1)
states: 13 z3:expr: (= dec0 (+ tmp10 (- 1)))
states: 14 z3:expr: (= x2 dec0)
states: 15 z3:expr: (= tmp20 x2)
states: 16 z3:expr: (= cmp30 (<= tmp20 1))
states: 17 z3:expr: cmp30
states: 18 z3:expr: (= y1 0)
states: 19 z3:expr: true
states: 20 z3:expr: true
```

```
states: 9 z3:expr: (= tmp1 x2)
states: 10 z3:expr: (= cmp1 (> tmp1 0))
states: 11 z3:expr: (not cmp1)
states: 21 z3:expr: true
states: 5 z3:expr: (= x3 1)
states: 6 z3:expr: (= x4 6)
states: 7 z3:expr: (= y2 1)
states: 8 z3:expr: true
states: 9 z3:expr: (= tmp2 x4)
states: 10 z3:expr: (= cmp2 (> tmp2 0))
states: 11 z3:expr: cmp2
states: 12 z3:expr: (= tmp11 x4)
states: 13 z3:expr: (= dec1 (+ tmp11 (- 1)))
states: 14 z3:expr: (= x5 dec1)
states: 15 z3:expr: (= tmp21 x5)
states: 16 z3:expr: (= cmp31 (<= tmp21 1))
states: 17 z3:expr: (not cmp31)
states: 20 z3:expr: true
```

Not feasible.

Path Feasibility

```
states: 0 z3:expr: true
states: 1 z3:expr: true
states: 2 z3:expr: true
states: 3 z3:expr: (= retval0 0)
states: 4 z3:expr: true
states: 5 z3:expr: (= x0 1)
states: 6 z3:expr: (= x1 6)
states: 7 z3:expr: (= y0 1)
states: 8 z3:expr: true
states: 9 z3:expr: (= tmp0 x1)
states: 10 z3:expr: (= cmp0 (> tmp0 0))
states: 11 z3:expr: cmp0
states: 12 z3:expr: (= tmp10 x1)
states: 13 z3:expr: (= dec0 (+ tmp10 (- 1)))
states: 14 z3:expr: (= x2 dec0)
states: 15 z3:expr: (= tmp20 x2)
states: 16 z3:expr: (= cmp30 (<= tmp20 1))
states: 17 z3:expr: (not cmp30)
states: 20 z3:expr: true
```

```
states: 9 z3:expr: (= tmp1 x2)
states: 10 z3:expr: (= cmp1 (> tmp1 0))
states: 11 z3:expr: cmp1
states: 12 z3:expr: (= tmp11 x2)
states: 13 z3:expr: (= dec1 (+ tmp11 (- 1)))
states: 14 z3:expr: (= x3 dec1)
states: 15 z3:expr: (= tmp21 x3)
states: 16 z3:expr: (= cmp31 (<= tmp21 1))
states: 17 z3:expr: (not cmp31)
states: 20 z3:expr: true
states: 9 z3:expr: (= tmp2 x3)
states: 10 z3:expr: (= cmp2 (> tmp2 0))
states: 11 z3:expr: cmp2
states: 12 z3:expr: (= tmp12 x3)
states: 13 z3:expr: (= dec2 (+ tmp12 (- 1)))
states: 14 z3:expr: (= x4 dec2)
states: 15 z3:expr: (= tmp22 x4)
states: 16 z3:expr: (= cmp32 (<= tmp22 1))
states: 17 z3:expr: (not cmp32)
states: 20 z3:expr: true
..............Path checker.............
satisfied
```

LTLf: $\mathbf{F}(x > 1)$

- Use a map to store pair $\langle a, x > 0 \rangle$
- The formula is given by $\mathbf{F}a$ and will be parsed by spot into an AST.
- Then we can extract the name of a and use a map to restore $x > 0$.
- Substitute the occurence of x with $x_1, x_2, \ldots$ along the path and do the checking.

LTLf: $\mathbf{F}(x > 1)$

```
tempFormula: true
origin: (> x 1)
subs: (not (> x0 1))
letter formula: true
origin: (> x 1)
subs: (not (> x0 1))
letter formula: true
origin: (> x 1)
subs: (not (> x0 1))
letter formula: true
origin: (> x 1)
subs: (not (> x0 1))
letter formula: (= retval0 0)
origin: (> x 1)
subs: (not (> x0 1))
letter formula: true
origin: (> x 1)
subs: (not (> x0 1))
letter formula: (= x0 1)
origin: (> x 1)
subs: (not (> x1 1))
letter formula: (= x1 6)
```

Sample Based Checker

Repeat the sampling and the checking several times.

Later Work

- Current Implementation is ugly:
- 1. Not a very decent combination of LTL and z3 checking.
- 2. Not simple for refactoring and later extension.
- 3. Mem leak may exists..
- Later add trace abstraction and extend the expressiveness of LTL logic.