hifrog - Bug #7795

Type constraints option for LA theories

29/08/2018 10:08 - Karine Even Mendoza

Status:	New	**Start date:**	29/08/2018
Priority:	High	**Due date:**	
Assignee:	Martin Blicha	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour

Description

Following Grigory answer, this code is a test for the assertion ordering than for the type constraints: assuming a >= 0 and b >= 0 we can easily derive that c >= 0.

The original code, suppose to run with type_constraints 1 and results in UNSAT. We can run the loop till 10 (no need for so many iterations). To prove claim 1 and 2 we need type-constraints 1, and using claim 1+2 we can prove claim 3 (I think via claims-opt option).

Older version of type-constraints use to work well. I think not all the added type-constraints are in the end added to the SMT query (can be the remove of incrementality).

==========
The code:

```
int sum ()
{
    int s=0;
    unsigned n;
    for (int i = 0; i <10; i++)
    {
        n = nondetUInt();
        s=s+n;
    }
    return s;
}

int main()
{
    int a,b,c;

    a=sum();
    assert(a>=0);

    b=sum();
    assert(b>=0);

    c=a+b;
    assert(c>=0);
}
```

Please add this test in the end to the regression test with reference to TACAS17 paper.

History

#1 - 29/08/2018 10:18 - Karine Even Mendoza

type-constraints 0: does nothing
type-constraints 1: adds bounds to non-det declaration, that is

char a;
char b = a + 1;

(and (=< 128 a#0) (< a#0 128))

type-constraints 2: adds bounds to non-det declaration and add assert to other vars (to check overflow), that is

```
char a;
char b = a + 1;
assert (a < b);
> (and (and (=< 128 a#0) (< a#0 128))  // assume
(not (and
(< a b)  // original assert
(and (=< 128 b#0) (< b#0 128)) // additional assert from type_constraints 2
))
// KE: I hope I am not wrong with this encoding, but the idea is that we add the constraints to the assert encoding and check its not (as if it were part of the assert :
assert(a < b && b.is_in_bounds);)
```

We can also run older version to see how it use to work.

#2 - 29/08/2018 15:29 - Martin Blicha

The example is missing function declaration.

```
unsigned int nondetUInt();
```

With that, the assertions are verified with type-constraints 1 and overflow is detected with type-constraints 2, as expected.