OpenSMT 2 - Bug #3511

Assertion violation: const [with T = unsigned int; RegionAllocator<T>::Ref = unsigned int]: Assertion `r < sz'

15/09/2016 13:45 - Karine Even Mendoza

Status:	Closed	**Start date:**	15/09/2016
Priority:	Normal	**Due date:**	
Assignee:	Antti Hyvärinen	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour

Description

Code example: hi-bench/main-bench/funfrog_regression/09_refinement/havoc_all_1.c --init-mode 0

The exception happen when creating the following ptref:
(= |funfrog::c::dec::?retval_tmp#1| (not (<= 1 (* (- 1) |c::a#9|))))

The error: evolcheck: /usr/local/include/opensmt/Alloc.h:64: const T& RegionAllocator<T>::operator[](RegionAllocator<T>::Ref) const [with T = unsigned int; RegionAllocator<T>::Ref = unsigned int]: Assertion `r < sz' failed.

It happens on the second time (when does refinement), here is the trace:
Checking Claim #1 (100%) ...
Last assertion location: 29 / 32 (5)

- NONDET abstraction used for function: c::__CPROVER_initialize
- INLINING function: c::main
  Processing a deferred function: c::main
- NONDET abstraction used for function: c::dec
- NONDET abstraction used for function: c::stub
- NONDET abstraction used for function: c::stub2
  SYMEX TIME: 0
  All SSA steps: 21
  Ignored SSA steps after slice: 8
  SLICER TIME: 0
  XXX Partition: 5 (ass_in_subtree: 0) - c::stub2 (loc: 25, TRU)
  XXX Partition: 4 (ass_in_subtree: 0) - c::stub (loc: 20, TRU)
  XXX Partition: 3 (ass_in_subtree: 0) - c::dec (loc: 16, TRU)
  XXX Partition: 2 (ass_in_subtree: 1) - c::main (loc: 12, INL)
  XXX Partition: 1 (ass_in_subtree: 0) - c::__CPROVER_initialize (loc: 1, TRU)
  XXX Partition: 0 (ass_in_subtree: 1) - nil (loc: 0, INL)
  CONVERSION TIME: 0.001
  RESULT
  ; ============================[Search Statistics]==============================
  ; | Conflicts | ORIGINAL | LEARNT | Progress |
  ; | | Vars Clauses Literals | Limit Clauses Lit/Cl | |
  ; ===
  ;   0 | 0   0 | 0.092 s | 94.555 MB
  SOLVER TIME: 0.002
  SAT - doesn't hold
  ASSERTION IS VIOLATED
  HAVOCING (of 4 calls) AREN'T SUITABLE FOR CHECKING ASSERTION.
- REFINING function: c::__CPROVER_initialize
- REFINING function: c::dec
- REFINING function: c::stub
  Go to next iteration
  Invalidating partition: 1
  Invalidating partition: 3
  Invalidating partition: 4
  Processing a deferred function: c::__CPROVER_initialize
  Processing a deferred function: c::dec
  Processing a deferred function: c::stub
  SYMEX TIME: 0
  All SSA steps: 61

Ignored SSA steps after slice: 39
SLICER TIME: 0
XXX Partition: 8 (ass_in_subtree: 0) - c::stub (loc: 20, INL)
XXX Partition: 7 (ass_in_subtree: 0) - c::dec (loc: 16, INL)
evolcheck: /usr/local/include/opensmt/Alloc.h:64: const T& RegionAllocator<T>::operator[](RegionAllocator<T>::Ref) const [with T = unsigned int; RegionAllocator<T>::Ref = unsigned int]: Assertion `r < sz' failed.
Aborted (core dumped)

The code before the refinement: ================================
```
(set-logic QF_LRA)
(declare-fun |funfrog::?callstart_symbol#5| () Bool)
(declare-fun |funfrog::?callend_symbol#5| () Bool)
(declare-fun |funfrog::?callstart_symbol#4| () Bool)
(declare-fun |funfrog::?callend_symbol#4| () Bool)
(declare-fun |funfrog::?callstart_symbol#3| () Bool)
(declare-fun |funfrog::?callend_symbol#3| () Bool)
(declare-fun |funfrog::?callstart_symbol#2| () Bool)
(declare-fun |funfrog::?callend_symbol#2| () Bool)
(declare-fun |funfrog::?error_symbol#1| () Bool)
(declare-fun |goto_symex::guard#1| () Bool)
(declare-fun |c::main::$tmp::return_value_dec$1!0#2| () Real)
(declare-fun |funfrog::c::dec::?retval#1| () Real)
(declare-fun |c::a#6| () Real)
(declare-fun |c::a#5| () Real)
(declare-fun .oite0 () Real)
(declare-fun |funfrog::?callstart_symbol#1| () Bool)
(declare-fun |funfrog::?callend_symbol#1| () Bool)
(assert
(and
; XXX Partition: 0
(and (= |c::main::$tmp::return_value_dec$1!0#2| |funfrog::c::dec::?retval#1|) (= |goto_symex::guard#1| (= 0
|c::main::$tmp::return_value_dec$1!0#2|)) (= |c::a#6| .oite0) (= |funfrog::?callstart_symbol#2| |funfrog::?callstart_symbol#3|) (= (and
|goto_symex::guard#1| (and |funfrog::?callend_symbol#3| |funfrog::?callstart_symbol#2|)) |funfrog::?callstart_symbol#4|) (= (and (not
|goto_symex::guard#1|) (and (or |funfrog::?callend_symbol#4| (not |goto_symex::guard#1|)) (and |funfrog::?callend_symbol#3|
|funfrog::?callstart_symbol#2|))) |funfrog::?callstart_symbol#5|) (= (not (or (not (and (or |funfrog::?callend_symbol#5|
|goto_symex::guard#1|) (and (or |funfrog::?callend_symbol#4| (not |goto_symex::guard#1|)) (and |funfrog::?callend_symbol#3|
|funfrog::?callstart_symbol#2|)))) (not (<= 0 (* |c::a#6| (- 1)))))) |funfrog::?error_symbol#1|) (or (not |funfrog::?callend_symbol#2|) (and
(or |funfrog::?callend_symbol#5| |goto_symex::guard#1|) (and (or |funfrog::?callend_symbol#4| (not |goto_symex::guard#1|)) (and
|funfrog::?callend_symbol#3| |funfrog::?callstart_symbol#2|)))))
; XXX Partition: 1
(and |funfrog::?error_symbol#1| |funfrog::?callstart_symbol#1| (= |funfrog::?callend_symbol#1| |funfrog::?callstart_symbol#2|))
; XXX oite symbol: .oite0
(and (or (not |goto_symex::guard#1|) (= 0 .oite0)) (or |goto_symex::guard#1| (= |c::a#5| .oite0)))
))
(check-sat)
```

The code while refine, so far is: ================================
```
(set-logic QF_LRA)
(declare-fun |funfrog::?callstart_symbol#5| () Bool)
(declare-fun |funfrog::?callend_symbol#5| () Bool)
(declare-fun |funfrog::?callstart_symbol#4| () Bool)
(declare-fun |funfrog::?callend_symbol#4| () Bool)
(declare-fun |funfrog::?callstart_symbol#3| () Bool)
(declare-fun |funfrog::?callend_symbol#3| () Bool)
(declare-fun |funfrog::?callstart_symbol#2| () Bool)
(declare-fun |funfrog::?callend_symbol#2| () Bool)
(declare-fun |funfrog::?error_symbol#1| () Bool)
(declare-fun |goto_symex::guard#1| () Bool)
(declare-fun |c::main::$tmp::return_value_dec$1!0#2| () Real)
(declare-fun |funfrog::c::dec::?retval#1| () Real)
(declare-fun |c::a#6| () Real)
(declare-fun |c::a#5| () Real)
(declare-fun .oite0 () Real)
(declare-fun |funfrog::?callstart_symbol#1| () Bool)
(declare-fun |funfrog::?callend_symbol#1| () Bool)
(declare-fun |c::a#9| () Real)
(declare-fun |c::a#4| () Real)
```

```
(declare-fun |funfrog::c::dec::?retval_tmp#1| () Real)
(assert
(and
; XXX Partition: 0
(and (= |c::main::$tmp::return_value_dec$1!0#2| |funfrog::c::dec::?retval#1|) (= |goto_symex::guard#1| (= 0
|c::main::$tmp::return_value_dec$1!0#2|)) (= |c::a#6| .oite0) (= |funfrog::?callstart_symbol#2| |funfrog::?callstart_symbol#3|) (= (and
|goto_symex::guard#1| (and |funfrog::?callend_symbol#3| |funfrog::?callstart_symbol#2|)) |funfrog::?callstart_symbol#4|) (= (and (not
|goto_symex::guard#1|) (and (or |funfrog::?callend_symbol#4| (not |goto_symex::guard#1|)) (and |funfrog::?callend_symbol#3|
|funfrog::?callstart_symbol#2|))) |funfrog::?callstart_symbol#5|) (= (not (or (not (and (or |funfrog::?callend_symbol#5|
|goto_symex::guard#1|) (and (or |funfrog::?callend_symbol#4| (not |goto_symex::guard#1|)) (and |funfrog::?callend_symbol#3|
|funfrog::?callstart_symbol#2|)))) (not (<= 0 (* |c::a#6| (- 1)))))) |funfrog::?error_symbol#1|) (or (not |funfrog::?callend_symbol#2|) (and
(or |funfrog::?callend_symbol#5| |goto_symex::guard#1|) (and (or |funfrog::?callend_symbol#4| (not |goto_symex::guard#1|)) (and
|funfrog::?callend_symbol#3| |funfrog::?callstart_symbol#2|)))))
; XXX Partition: 1
(and |funfrog::?error_symbol#1| |funfrog::?callstart_symbol#1| (= |funfrog::?callend_symbol#1| |funfrog::?callstart_symbol#2|))
; XXX Partition: 2
(or |funfrog::?callstart_symbol#4| (not |funfrog::?callend_symbol#4|))
; XXX oite symbol: .oite0
(and (or (not |goto_symex::guard#1|) (= 0 .oite0)) (or |goto_symex::guard#1| (= |c::a#5| .oite0)))
))
(check-sat)
evolcheck: /usr/local/include/opensmt/Alloc.h:64: const T& RegionAllocator<T>::operator[](RegionAllocator<T>::Ref) const [with T =
unsigned int; RegionAllocator<T>::Ref = unsigned int]: Assertion `r < sz' failed.
Aborted (core dumped)
```

History

#1 - 16/09/2016 12:24 - Antti Hyvärinen

It looks like the problem is that HiFrog wants to equate (not (<= 1 (* (- 1) |c::a#9|))) with |funfrog::c::dec::?retval_tmp#1|.
Since the former is Boolean and the latter is Real the result is undefined.

Looking at the benchmark this is exactly what happens there, and is of course perfectly ok in C. In my opinion this should be fixed in HiFrog. What do you say?

#2 - 12/10/2016 17:21 - Karine Even Mendoza

This case seems to be ok, I think we fixed this specific bug.

Maybe it is the same case with disk.c claim 1?

#3 - 12/10/2016 17:23 - Karine Even Mendoza

- Status changed from New to Closed

- % Done changed from 0 to 100