

1) The figure below shows the test feature 'is\_negative' at the call to 'compare\_functions'. In the call, I pass "agent n.is\_negative" as the argument, where 'n' is a JJ\_BIG\_NATURAL\_8.

The screenshot shows the Squeak IDE with the 'is\_negative' feature selected. The feature code is as follows:

```

is_negative
-- Tests the corresponding feature from JJ_BIG_NATURAL.
local
  fn: STRING 8
  n: like number_anchor
  i: INTEGER_32
do
  fn := "is_negative"
  from
    i := 1
  until
    i > test_limit
  loop
    n := new_random_number (word_limit)
    compare_functions (agent n.is_negative, True, fn, ARRAY [ANY] <<>>)
    i := i + 1
  end
end

```

The 'Objects' table shows the current state of the execution:

Name	Value	Type	Address
Exception raised	target_closed: PREC...		
Current object	<0x7FD06E380D48>	BIG_NATURAL_8_TESTS	0x7FD06E380D48
Locals			
fn	is_negative	STRING_8	0x7FD06E380D50
i	1	INTEGER_32	
n	-595901053	JJ_BIG_NATURAL_8	0x7FD06E380D88

The 'In Feature' table shows the call stack:

In Feature	In Class	From Class	@
compare_func...	BIG_NATUR...	BIG_NATUR...	1
is_negative	BIG_NATUR...	BIG_NATUR...	5
is_negative	BIG_NATUR...	BIG_NATUR...	1
run_all	BIG_NATUR...	BIG_NATUR...	2
make	BIG_NATUR...	BIG_NATUR...	8+1

However, inside feature 'compare\_functions', the argument is not a JJ\_BIG\_NATURAL\_8; it is a BIG\_NATURAL\_8\_TEST, which is the type of the enclosing class at the call to 'compare\_functions'. Hence it fails on the precondition. ?????

The screenshot shows the Squeak IDE with the 'compare\_functions' feature selected. The feature code is as follows:

```

compare_functions (a_routine: ROUTINE; a_expected: ANY; a_name: STRING 8; a_args: ARRAY [ANY])
-- Compare the result of 'a_routine' from {JJ_BIG NUMBER} to the one of:
-- 1) if 'a_expected' is a ROUTINE, the result of executing a
--    call to that routine, or
-- 2) if NOT a ROUTINE, the result of calling out on that argument.
-- Return the result of the comparison, possibly printing (depending
-- on is_terse) the result of executing 'a_routine' (and the expected
-- value if they don't match).
-- Arguments 'a_name' and 'a_args' are only used to output the signature
-- of 'a function'.
-- (from BIG_NATURAL_TESTS)
-- (export status {NONE})
require -- from BIG_NATURAL_TESTS
target_closed: attached {like number_anchor} a_routine.target
other target_closed: attached {ROUTINE} a_expected as r implies attached {like number_anchor} r.target
arguments_closed: a_routine.open_count = 0
other arguments_closed: attached {ROUTINE} a_expected as r implies r.open_count = 0
local
  s: STRING 8

```

The 'Objects' table shows the current state of the execution:

Name	Value	Type	Ad
Exception raised	target_closed: PREC...		
Current object	<0x7FD06E380D48>	BIG_NATURAL_8_TESTS	0x7
Arguments			
a_routine	<0x7FD06E380CF0>	PREDICATE [ITUPLE]	0x7
Agent	is_negative		
calc_rout...	0x0	POINTER	
closed_op...	<0x7FD06E380DF8>	TUPLE [BIG_NATURAL_8_TESTS, JJ_BIG_NATURAL_8]	0x7
encaps_ro...	0x10ED90FE0	POINTER	
is_basic	False	BOOLEAN	
is_target_c...	True	BOOLEAN	
last result	False	BOOLEAN	

The 'Watch' table shows the current state of the execution:

Expression	Value	Type	Ad
a_routine.target	<0x7F0...	BIG_NATURAL_8_TESTS	0x7

2) Feature 'set\_with\_array' was giving strange results in the debugger. In the debugger the three circled values in the picture below were all different; they should have been the same, as they are now. Argument 'a\_array' came in with certain values but the value assigned to 'd' was different; when that value was placed into Current, it was again a different value. Then, I noticed that the local variable 'place' was never used, and I had no idea why I had put that there, so I deleted that line. Now the procedure works! I then I put the unused local back in (i.e. undo edit) and, wow, the procedure STILL works as expected; I now cannot reproduce the error. ???

Flat view of feature 'set\_with\_array' of class JJ\_BIG\_NATURAL\_8

```

set_with_array (a_array: ARRAY [like word])
-- Set the words from 'a_array', where the array holds
-- intended words with high-order words first.
-- (from JJ_BIG_NATURAL)
require -- from JJ_BIG_NATURAL
array_exists: a_array /= Void
array_not_empty: not a_array.is_empty

local
i: INTEGER_32
d: like word
place: JJ_BIG_NATURAL_8

do
wipe_out
from
i := a_array.count
until
i < 1
loop
d := a_array [i]
extend (a_array [i])
i := i - 1
end
end

```

What is this?  
After deleting this unused local, this procedure seems to work fine. ???  
But after replacing the line, the error is no longer happening. ???

Before deleting 'place':  
on entry, a\_array [4] was 242.  
After assignment, 'd' became 168.  
Extending into Current added some other value [don't remember what that value was, because it works now.]

Name	Value	Type	Addr	Expression
count	1			a_array.index
capacity	10			...
0	242	NATURAL_8		
Once routi...				
Constants				
index	0	INTEGER_32		
is_negative	False	BOOLEAN		
is_unstable	False	BOOLEAN		
object_comp...	False	BOOLEAN		
Once routines				
Constants				
Arguments				
a_array	<0x7F846AB80D38>	ARRAY [NATURAL_8]	0x7F...	
area	count=4, capacity=4	SPECIAL [NATURAL_8]	0x7F...	
count	4			
capacity	4			
0	187	NATURAL_8	0x7F...	
1	133	NATURAL_8	0x7F...	
2	35	NATURAL_8	0x7F...	
3	242	NATURAL_8	0x7F...	
Once ro...				
Constants				
lower	1	INTEGER_32		
object_co...	False	BOOLEAN		
upper	4	INTEGER_32		
Once routi...				
Constants				
Locals				
d	242	NATURAL_8	0x7F...	
i	4	INTEGER_32		
place	Void	JJ_BIG_NATURAL	Void	

Watch » 2 Call Stack AutoTest F