Base Language - Bug #3216

second operand of logical AND and OR operators must be deferred

12/05/2016 08:45 AM - Ovidiu Maxiniuc

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Ovidiu Maxiniuc	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#2 - 12/05/2016 09:20 AM - Ovidiu Maxiniuc

The second (right) operand of AND and OR operators must always be deferred. In very simple cases, it may be that this is not necessary (E.g. when the right operand is a logical literal, variable or field). However, it is a usual habit for programmers to test for a certain condition that is guarded by another one. One example is in C/C++/Java when testing the value referred by a pointer: if (p != null and p.boolVal). The second operand is not evaluated unless p is a valid pointer, otherwise a NPE or similar would be thrown.

4GL does the same. Recently, I discovered in client code, constructs like this:

```
define variable empty-string as character init "".
IF NUM-ENTRIES(empty-string) GE 2 AND ENTRY(2, empty-string) = "ab"
then message "ok".
```

Normally, this is a correct code, that does not display anything. Not deferring the evaluation of the second operand until the left operand is evaluated will cause the following condition:

```
** Entry 2 is outside the range of list. (560)
```

We cannot predict at conversion time the expressions that do not throw such conditions except for the very simple cases. So we must evaluate the right operand conditioned by the first one, not based on whether or not it has visible side-effects.
On the bright side, deferring the evaluation of such expression might be a performance improvement if complex right operands are not evaluated any more because the operator was short-cut at the moment when the left operand value is known.

#3 - 12/05/2016 10:03 AM - Greg Shah

We usually handle 2nd operand deferral properly. The case you have cited, should work. See the next-child-rules section of convert/operators.rules. Presumably the defer_evaluation annotation is not set in this case, but it should be set. I suspect the bug is in annotations/deferred_logical_evaluation.rules where we calculate that value.

#4 - 12/05/2016 10:43 AM - Ovidiu Maxiniuc

Greg Shah wrote:

> We usually handle 2nd operand deferral properly. The case you have cited, should work. See the next-child-rules section of convert/operators.rules. Presumably the defer_evaluation annotation is not set in this case, but it should be set. I suspect the bug is in annotations/deferred_logical_evaluation.rules where we calculate that value.

Indeed, the annotation was missing. The deferred_logical_evaluation added it only for special cases (when a the function in right operand has a side effect). I tried to improve those conditions but they can depend on business logic of the 4GL application and cannot be predicted at conversion time. I have chosen to go with deferring all right operands. The only downside would be if the evaluation of the extra lambdas are really slow. You worked in this area and probably done some tests here. Are they a reason of performance concern?

#5 - 12/05/2016 11:44 AM - Greg Shah

> I tried to improve those conditions but they can depend on business logic of the 4GL application and cannot be predicted at conversion time.

The change that will fix this specific case would be to add kw_entry to the section under here:

```
            <!-- unsafe when the accessed resource is uninitialized or can
                 otherwise raise an ERROR condition -->
```

That won't solve other possible cases. But we have already marked the known dangerous cases. We need to review this after we add new built-in support, we haven't done that recently.

> The only downside would be if the evaluation of the extra lambdas are really slow. You worked in this area and probably done some tests here. Are they a reason of performance concern?

Not really. There can be a performance hit for capturing lambdas, but it is so small that the natural variability of Java code will overwhelm it. In other words, any negative performance cannot be reliably measured at the application level.

The original idea behind this was less about performance and more about emitting simpler code. After the move to lambdas, the impact of deferred evaluation is much less, though it is still present a little.

I guess I'm OK with making things default to deferred mode. It is a safer default approach.

But even if we go down the path of enabling deferred evaluation by default, we would still want to defer evaluation of simple (non-array) variable references.

#6 - 12/05/2016 11:45 AM - Greg Shah

- Project changed from Liberty to Base Language

#7 - 12/07/2016 08:40 AM - Greg Shah

3201b was merged to trunk as revision 11132. It changed all right operands to be deferred.

> But even if we go down the path of enabling deferred evaluation by default, we would still want to defer evaluation of simple (non-array) variable references.

I still want to exclude these. It is a small improvement, but it will reduce the amount of code emitted in some non-trivial number of cases.

#8 - 12/07/2016 09:13 AM - Greg Shah

- % Done changed from 0 to 90

#9 - 12/07/2016 09:27 AM - Ovidiu Maxiniuc

- Status changed from New to WIP

- % Done changed from 90 to 0

- Start date deleted (12/05/2016)

Greg Shah wrote:

> I still want to exclude these. It is a small improvement, but it will reduce the amount of code emitted in some non-trivial number of cases.

I understand, I will exclude lambas for simple (non-array) variable references in the a branch. Pushing the change to already stable 3201b was a little risky so I preferred to defer the change to new 3201c.

#10 - 12/07/2016 09:27 AM - Ovidiu Maxiniuc

- % Done changed from 0 to 90

#11 - 12/07/2016 09:31 AM - Greg Shah

Yes, that makes good sense.

#12 - 01/12/2017 07:40 AM - Ovidiu Maxiniuc

The final patch for this task is part of 11135 revision, committed from 3201c. This issue can be closed.

#13 - 01/12/2017 07:45 AM - Greg Shah

- % Done changed from 90 to 100

- Status changed from WIP to Closed

- Assignee set to Ovidiu Maxiniuc