

Base Language - Bug #11977

A literal left behind when a constant conditional collapses is not wrapped.

10/12/2026 10:18 AM - Octavian Adrian Gavril

Status:	Internal Test	Start date:	
Priority:	Normal	Due date:	
Assignee:	Octavian Adrian Gavril	% Done:	100%
Category:		Estimated time:	0.00 hour
Target version:			
billable:	No	reviewer:	Ovidiu Maxiniuc
vendor_id:	GCD	production:	No
case_num:		env_name:	
version_reported:		topics:	Conversion
version_resolved:			
Description			

History

#1 - 10/12/2026 10:22 AM - Octavian Adrian Gavril

- topics Conversion added

Here is a test case:

```
define temp-table ttcust no-undo
  field code as character
  field qty as integer.

create ttcust.

assign
  ttcust.code = (if "abc" <> ? then "abc" else ttcust.code)
  ttcust.qty = (if 1 <> ? then 1 else ttcust.qty).

ttcust.code = ("ghi").

display ttcust.code ttcust.qty.
```

Text and NumberType wrapper types are not generated properly. As a result, the converted code ends up looking like this:

```
batch(() ->
{
  ttcust.setCode(("abc"));
  ttcust.setQty(1);
});
```

This results in the following compilation errors:

```
just-compile:
[javac] Compiling 28 source files to /home/aog/gcd/dataset/build/classes
[javac] Compiling 2 source files to /home/aog/gcd/dataset/build/classes
[javac] /home/aog/gcd/dataset/src/com/goldencode/dataset/A.java:38: error: incompatible types: String
```

```

cannot be converted to Text
[javac]          ttcust.setCode(("abc"));
[javac]          ^
[javac] /home/aog/gcd/dataset/src/com/goldencode/dataset/A.java:39: error: incompatible types: int
cannot be converted to NumberType
[javac]          ttcust.setQty((1));
[javac]          ^
[javac] Note: Some messages have been simplified; recompile with -Xdiags:verbose to get full output
[javac] 2 errors

```

BUILD FAILED

#3 - 10/13/2026 03:28 AM - Octavian Adrian Gavril

- Assignee set to Octavian Adrian Gavril
- Status changed from New to WIP
- reviewer Ovidiu Maxiniuc added

This is the original AST for `ttcust.code = (if "abc" <> ? then "abc" else ttcust.code)`:

```

<ast col="16" id="12884901927" line="8" text="=" type="ASSIGN">
  [...]
  <ast col="4" id="12884901930" line="8" text="ttcust.code" type="FIELD_CHAR">
    [...]
  </ast>
  <ast col="0" id="12884901931" line="0" text="expression" type="EXPRESSION">
    [...]
    <ast col="18" id="12884901932" line="8" text="(" type="LPARENS">
      [...]
      <ast col="38" id="12884901943" line="8" text="&quot;abc&quot;;" type="STRING">
        [...]
      </ast>
    </ast>
  </ast>
</ast>

```

In annotations/cleanup.rules, we reduce expressions to a single output when possible, but the LPARENS node remains. Later, when literal wrapping should be performed in convert/literals, the `isAssign = upPath(this, prog.assign, prog.expression)` check fails because of the extra layer added by LPARENS.

I created task branch **11977a** and committed a fix for this issue in **rev16800**. Now we drop the parenthesis node whenever a single leaf node is all that remains.

This is how the AST looks after the fix:

```

<ast col="16" id="12884901927" line="8" text="=" type="ASSIGN">
  [...]
  <ast col="4" id="12884901930" line="8" text="ttcust.code" type="FIELD_CHAR">
    [...]
  </ast>
  <ast col="0" id="12884901931" line="0" text="expression" type="EXPRESSION">
    [...]
    <ast col="38" id="12884901943" line="8" text="&quot;abc&quot;;" type="STRING">

```

```
[...]  
</ast>  
</ast>  
</ast>
```

And here is the generated output:

```
batch(() ->  
{  
    ttcust.setCode(new character("abc"));  
    ttcust.setQty(new integer(1));  
});
```

So, this testcase is resolved in **11977a/16800**.

I noticed that Ovidiu worked in this area as well.
Ovidiu, please review these changes.

#4 - 10/13/2026 03:28 AM - Octavian Adrian Gavril

- % Done changed from 0 to 100

- Status changed from WIP to Review

#5 - 10/13/2026 08:03 AM - Greg Shah

I don't have an issue with the concept. But I also don't understand why we are seeing this issue now. It seems like a regression.

#6 - 10/13/2026 08:21 AM - Octavian Adrian Gavril

Greg Shah wrote:

I don't have an issue with the concept. But I also don't understand why we are seeing this issue now. It seems like a regression.

I suspect that this is not a common way to write conditional assignments. None of our active customers use this syntax in their current codebase.

#7 - 10/13/2026 06:56 PM - Ovidiu Maxiniuc

- Status changed from Review to Internal Test

I understand that the trunk was expecting to reduce the assignment to something like:

```
ttcust.code = ("abc")
```

How will these expressions convert? Your changes are limited to reduced ternary cases. I am just thinking if we should move this code in a more generic place.

The changes are good. It should work as described.

To note: `ref.numChildren == 1` is always true (if `lprens` doesn't have a exactly one child then there is a problem) and can be dropped.

PS: maybe that strange condition was 'composed' using `includes`? Or it was simply written to comment out some code - I do sometime add the equivalent of `if false` in front of a statement to temporarily disable it.