Item 341
git_comments:

1. Recursively enter all enclosing nodes, as appropriate.
2. This node is a child of a node that has been passed over via CompositeBehavior, and should also be skipped. All child nodes of a skipped composite should always be skipped.
3. These checks occur after visiting the enclosing node to ensure that if this node has been visited while visiting the enclosing node the node is not revisited, or, if an enclosing Node is skipped, this node is also skipped.

git_commits:

1. **summary:** This closes #3329
   message: This closes #3329

github_issues:

github_issues_comments:

github_pulls:

1. **title:** Visit a Transform Hierarchy in Topological Order
   body: This reverts commit 6ad6433ec0c02aec8656e9e3b27f6e0f974f8ece. Be sure to do all of the following to help us incorporate your contribution quickly and easily: - [x] Make sure the PR title is formatted like: `[BEAM-<Jira issue #>] Description of pull request` - [x] Make sure tests pass via `mvn clean verify`. - [x] Replace `<Jira issue #>` in the title with the actual Jira issue number, if there is one. - [x] If this contribution is large, please file an Apache [Individual Contributor License Agreement](https://www.apache.org/licenses/icla.pdf). ---

github_pulls_comments:

1. R: @ssisk
2. [![Coverage Status](https://coveralls.io/builds/11891964/badge)](https://coveralls.io/builds/11891964) Coverage increased (+0.01%) to 70.592% when pulling **8b436faa4c844e3aa1a98f5c9db356f7c0f4ba2d on tgroh:topo_order_only** into **b53e6f0dc91948040364d1e742b023d5a2be8433 on apache:master**.
3. [![Coverage Status](https://coveralls.io/builds/11897185/badge)](https://coveralls.io/builds/11897185) Coverage increased (+0.005%) to 70.6% when pulling **7c3c32b8c5d88cb63f9379f27751da4ef41bdb41 on tgroh:topo_order_only** into **e980ae921f48b180ffb7fd7dba7eea10ef0f304a on apache:master**.
4. [![Coverage Status](https://coveralls.io/builds/11911595/badge)](https://coveralls.io/builds/11911595) Coverage increased (+0.009%) to 70.6% when pulling **97e2a5c96e568ac62c70e0153693496347f30098 on tgroh:topo_order_only** into **d2c4093ed2fd47e28b31898a17e28d0575aa9ff9 on apache:master**.
5. [![Coverage Status](https://coveralls.io/builds/11911718/badge)](https://coveralls.io/builds/11911718) Coverage decreased (-0.002%) to 70.589% when pulling **97e2a5c96e568ac62c70e0153693496347f30098 on tgroh:topo_order_only** into **d2c4093ed2fd47e28b31898a17e28d0575aa9ff9 on apache:master**.
6. [![Coverage Status](https://coveralls.io/builds/11912121/badge)](https://coveralls.io/builds/11912121) Coverage increased (+0.01%) to 70.603% when pulling **97e2a5c96e568ac62c70e0153693496347f30098 on tgroh:topo_order_only** into **d2c4093ed2fd47e28b31898a17e28d0575aa9ff9 on apache:master**.

github_pulls_reviews:

1. given the if statement that encloses this add, it seems like by definition this node is a child node. So why is it getting added to a set whose name implies that the contents are all composites?
2. you have your "am I allowed to visit this node?" logic handled differently: you're doing some checks at the beginning of this method, while this check is happening before the call. Might be clearer to move all the checks to the beginning of visit?
3. it seems odd that you're visit()ing other nodes before checking to see if this has been visited? Seems worth explaining why this is.

4. I keep wondering if we really need a difference between visitedNodes and composites - if you replaced this check with childNodeOf(visitedNodes), is there really any difference? if not, can you get rid of the passedComposites concept?
5. looking more, I realize you first visit the enclosing nodes, so this check would always fail with my suggestion. But I'm very suspicious of that first visitEnclosing call, so I'll leave this comment.
6. per our offline discussion: please explain how graph translation can affect visit()
7. if I were changing this code, I'd prefer to have a set of small tests structured like "doesn't revisit composite when arriving at child".
8. perhaps beefier comment explaining expected traversal behavior? the logic here seems sufficiently complex that it would help explain exactly why we do what we do.
9. I've added notes about the ordering, and the additional invariants we enforce.
10. This is the "Always have entered an enclosing node before visiting any enclosed nodes" invariant
11. Assuming our other invariants hold, we'll walk up the stack of nodes, and then immediately return from all of them. We could short-circuit this check in the case we have already been visited, but doing so requires duplicating rather than moving the code (as we may/will visit this node as part of the process of visiting its parent)
12. visited nodes includes nodes which we have reached; skippedComposites are all of the composites which returned `DO_NOT_ENTER_TRANSFORM` from `PipelineVisitor.enterCompositeTransform(Node)`, and as a result all children should also be skipped.
13. The child may or may not be a composite. This is minor performance improvement (as we won't walk up until we reach the skipped composite), but isn't necessary
14. I like having it here rather than earlier because it's the only invariant that relates to values, so having it local to visiting the values makes sense to me.
15. I've added some assertions to ensure that nodes are not visited more than once, and a test to demonstrate the children of skipped nodes are not visited.
16. should we assert that visited/exited/values all contain the right contents at the end of the test? Right now this test would pass if none of the nodes were visited (which would presumably fail a different test)
17. maybe assert that we did visit the enclosing node?

jira_issues:

jira_issues_comments: