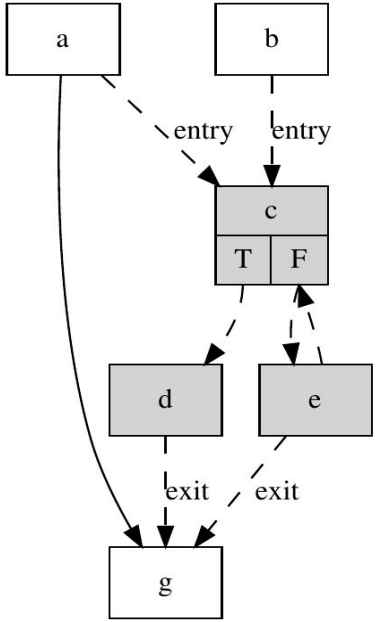
Reducing the Computational Complexity of RegionInfo

Nandini Singhal[1], Pratik Bhatu[1], Aditya Kumar[2], Tobias Grosser[3], Ramakrishna Upadrasta[1]
[1]Indian Institute of Technology Hyderabad
[2]Samsung R&D Austin
[3]ETH Zurich

Refined Region



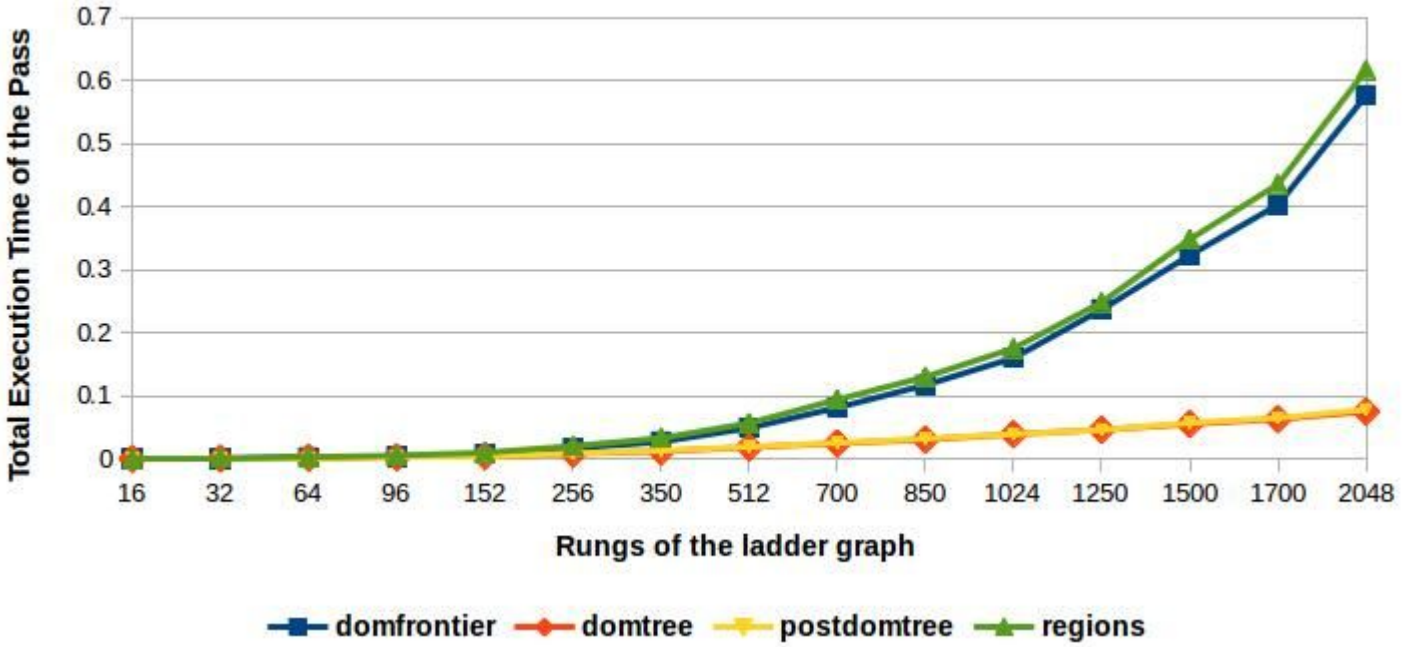
Why regions?

- Part of CFG which satisfies certain properties can be optimised without affecting rest of CFG
- Divide and conquer algorithms on region tree

Being used as SCoPs in Polly and also in AMD GPU backend

What is the problem?

RegionInfo uses dominance frontier



Proposed Algorithm

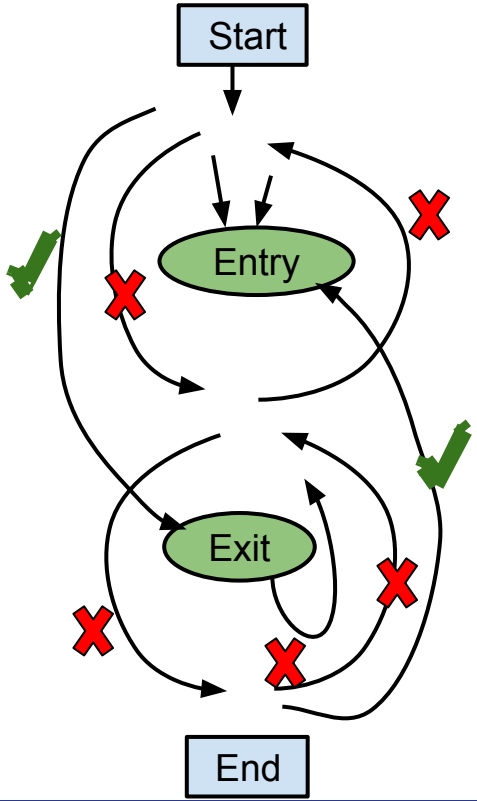
Forward edges

Back edges

Entry dominates Exit

Traverse nodes from entry until exit while skipping detected regions and already traversed regions which failed other checks.

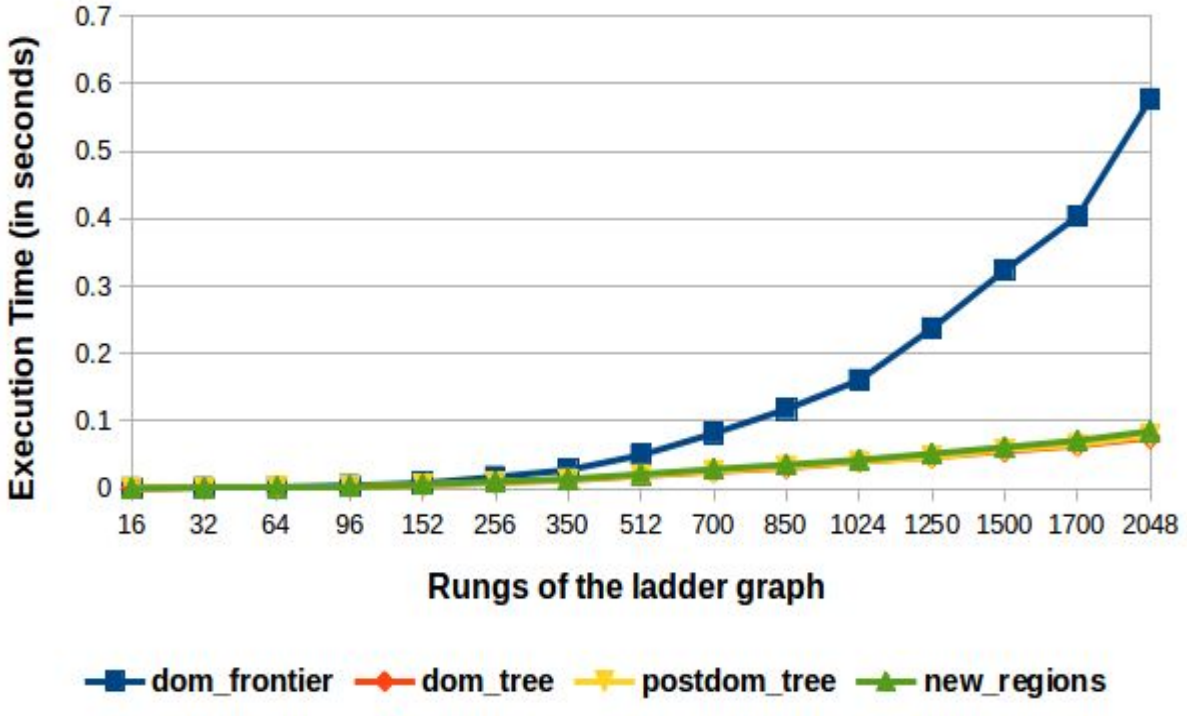
Exit post-dominates Entry



- **Identify innermost loops in which all edges to entry reside.**
- **If none, no back edge i.e. passes check**
- **If more than 2 different innermost loops, then fails region check**
- **Else store the loop and compare with exit later**

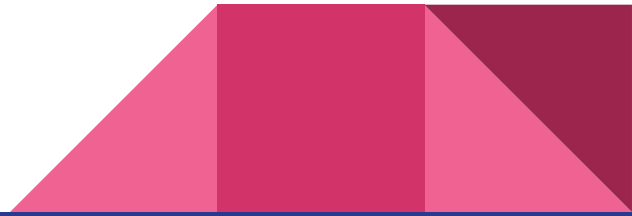
- **Get LoopForEdge for preds of Exit**
- **Compare Entry and Exit's InnermostLoopForEdge**
- **If different, not a region**

RegionInfo using Dom, PDom, LoopInfo



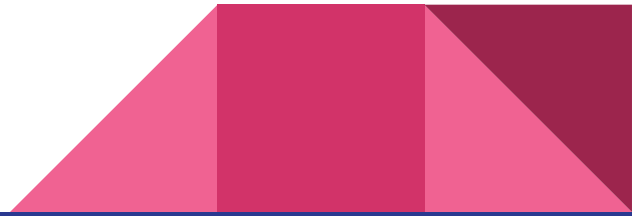
Conclusion

- Reduced complexity of RegionInfo pass
- No of refined regions detected remain same



To be explored..

- How to connect Regions and Loops better?
- On-demand analysis of only the required regions
- Remove the redundant RegionPass?



Thank You!